

UNIVERSITY OF
BIRMINGHAM

Enhancing the Bees Algorithm using the Traplining Metaphor

by

Asrul Harun Ismail

A thesis submitted to the University of Birmingham for the degree of

DOCTOR OF PHILOSOPHY

Department of Mechanical Engineering

School of Engineering

College of Engineering and Physical Sciences

University of Birmingham

2021

UNIVERSITY OF
BIRMINGHAM

University of Birmingham Research Archive

e-theses repository

This unpublished thesis/dissertation is copyright of the author and/or third parties. The intellectual property rights of the author or third parties in respect of this work are as defined by The Copyright Designs and Patents Act 1988 or as modified by any successor legislation.

Any use made of information contained in this thesis/dissertation must be in accordance with that legislation and must be properly acknowledged. Further distribution or reproduction in any format is prohibited without the permission of the copyright holder.

For my beloved Grahita, Aaron, and Rafa.

For my parents, parents in law and grand parents.

For my brother (in Heaven), sisters, cousins, uncles, aunts, nephews, and nieces.

For all my friends.

"Allah (God) does not intend to make difficulty for you, but He intends to purify you and complete His favour upon that you may be grateful."

- Quran 5:6

"Indeed, I have rewarded them this Day for their patient endurance – that they are the attainers (of success)."

- Al Mu'minun 111

"The stone which the builders rejected has become the chief cornerstone."

- Psalm 118:22

"It takes time for a fruit to mature and acquire sweetness and become eatable; time is a prime factor for most good fortunes."

- The Vedas

"Better it is to live one day seeing the rise and fall of things than to live a hundred years without ever seeing the rise and fall of things."

- Buddha

Abstract

This work aims to improve the performance of the Bees Algorithm (BA), particularly in term of simplicity, accuracy, and convergence. Three improvements were made in this study as a result of bees' traplining behaviour.

The first improvement was the parameter reduction of the Bees Algorithm. This strategy recruits and assigns worker bees to exploit and explore all patches. Both searching processes are assigned using the Triangular Distribution Random Number Generator. The most promising patches have more workers and are subject to more exploitation than the less productive patches. This technique reduced the original parameters into two parameters. The results show that the Bi-BA is just as efficient as the basic BA, although it has fewer parameters.

Following that, another improvement was proposed to increase the diversification performance of the Combinatorial Bees Algorithm (CBA). The technique employs a novel constructive heuristic that considers the distance and the turning angle of the bees' flight. When foraging for honey, bees generally avoid making a sharp turn. By including this turning angle as the second consideration, it can control CBA's initial solution diversity.

Third, the CBA is strengthened to enable an intensification strategy that avoids falling into a local optima trap. The approach is based on the behaviour of bees when confronted with threats. They will keep away from re-visiting those flowers during the next bout for reasons like predators, rivals, or honey run out. The approach will remove temporarily threatened flowers from the whole tour, eliminating the sharp turn, and reintroduces them again to the habitual tour's nearest edge. The technique could effectively achieve an equilibrium between exploration and exploitation mechanisms. The results show that the strategy is very competitive compared to other population-based nature-inspired algorithms.

Finally, the enhanced Bees Algorithms are demonstrated on two real-world engineering problems, namely, Printed Circuit Board insertion sequencing and vehicles routing problem.

Acknowledgements

First of all, I am thankful to Allah S.W.T. for providing me with the courage to complete this report. My sincere gratitude to my supervisor, Professor Duc Truong Pham, for all of his encouragement and patience in guiding me on this PhD. To the Newton Fund, the British Council, the Indonesian Ministry of Higher Education, and Universitas Pancasila, none of this will be feasible without your patronage. Many thanks. To Prof Panos, Dr Mozafar Saadat and Dr Marco Castellani. I would like to express my heartfelt appreciation for providing me with helpful suggestions.

To my research colleagues Turki, Al Antoni, Ali, Ruiya, Natalia, Sultan, Kaiwen, Mario, Natinee, Mo, Joey, Mairi, Andy, Marcel, Murat, Feiying and Feifan, as well as to all my all Indonesian friends- Fahmi, Bela/Yane, Jonri, Hargyo, Riza/Syawal, Hermawan/Ria, Kris/Firda, Febi, Gumilar, Yusuf/Kurnia, Wishnu/Silvi, Ari, Bimo/Ebi, Any Math, Adi, Hijrah, Ari/Dewi, Boby, Adnan/Rini, Ria/Mansyur, Dea/Yudha, Fadli, Agung/Ana, Tkuz, Cipok, Tinja, Tius, Yansi, Miko, Pea, Aan, Hendra, Franky, Hastria, Ngacir, Dimz, Metz, Benjol, Tri, Adrian, Bobi, all PL'97 n 98, Robet, Buce, Widi, all MGM'94, Buyung, Kelik, Aji, Retno, Woro, Mira, Ika, Nana, Gama, all TI UGM, Sevi/Moki, Moko, Dayu, Iyus, Hans, Irly, all kontrakan gang Legi, Sari, Rani, Elty, Syntia, Om Andi, Rida, Yus, Gilang, Aldes, Doyok, Ganesha, Sigit, Ary, Burhan, Imron, all TMI ITB'06, Donna, Ai, Rio, all SDM gank, all TIUP gank, and all wow gamers gank- I would like to express my gratitude for their constant support.

My sincere appreciation to my parents, mother in-laws, Astri, Astari, Astrid, Lintang, Fikar, Kifli, Mbah Jafar, Om Rustam fams, Tres fams, Om Jun fams, Om Gatot fams, Pejompongan fams, and other families for their faith in me and for their prayers.

Finally, the most important, this is for my beloved wife Grahita Swarawisnu and my two sons, Aaron Alif Akasha and Rafa Arsa Ishmael. I appreciate your patience and empathy whenever I was unable to prioritise my responsibilities before this study concluded.

Abbreviations, Acronyms, and Symbols

List of Abbreviations and Acronyms

ABC Artificial Bee Colony

ACO Ant Colony Optimisation

ACOR ACO for real-valued continuous optimisation

ACS Ant Colony System

AI Artificial Intelligence

ANFIS Adaptive Neuro-Fuzzy Inference System

ANN Artificial Neural Networks

AS Ant System

BA Bees Algorithm

BCO Bee Colony Optimisation

Bi-BA Bi-Parameters Bees Algorithm

Bi-SBA Bi-Parameters Standard Bees Algorithm

BNSN Bees Nearest Straight Neighbour

BRO Bees Routing Optimiser

C.I. Confidence Interval

CA Cultural Algorithm

CBA Combinatorial Bees Algorithm

Cont Continuous Optimisation Problem

COP Combinatorial Optimisation Problem

CS Cuckoo Search

CVRP Capacitated Vehicle Routing Problem

DE Differential Evolution

DSH Domino Sequence Heuristic

EA Evolutionary Algorithms

EBA Enhanced Bees Algorithm

EED Environmental/Economic Dispatch

EP Evolutionary Programming

ES Evolutionary Strategy

FA Firefly Algorithm

FBPS Frequency-Based Pruning Strategy

FNN Feedforward Neural Network

FRNN Fixed-Radius Near Neighbour

GA Genetic Algorithm

GRASP Greedy Randomised Search Procedure

GSTM Greedy Sub Tour Mutation

HBMO Honey Bees Mating Optimisation

HS Harmony Search

KF Kalman Filter

LK Lin Kernighan

MBO Marriage in Honey Bees Optimisation

MBTD Moving Board with Time Delay

ML Machine Learning

MLP Multi-layer Perceptron

MOBA Multi-Objectives Bees Algorithm

MOCBA Multi-Objectives Combinatorial Bees Algorithm

NFE Number of Function Evaluation

NN Neural Networks

NNH Nearest Neighbour Heuristic

NP Non-deterministic Polynomial-time

OPF Optimal Power Flow

PCB Printed Circuit Board

PID Proportional-Integral-Derivative

PSO Particle Swarm Optimisation

qABC Quick Artificial Bee Colony

RBF Radial Basis Function

RNG Random Number Generator

SA Simulated Annealing

SS Scatter Search

Succ. Success rate

SVM Support Vector Machine

TRIZ Teoriya Resheniya Izobretatelskikh Zadatch

TS Tabu Search

TSP Travelling Salesman Problem

VNS Variable Neighbourhood Search

VPT Vantage Point Tree

VRP Vehicle Routing Problem

List of Symbols

Δr The additional range of bees' vision

AT The assembly time

Avg The average solution obtained from several running experiments

$Best$ The best solution obtained from several running experiments

BKS The best known solution

$C_{i,j}$ The cost from node i to j

CV The total violation of vehicle capacity

$Dims$ Number of problem dimensions

E Edge

e Number of elite sites in the Bees Algorithm

$Err.$ The error

$EvalRat$ The ratio of evaluations to reach best solution

F Function(s)

f Objective function(s)

FL The maximum number of flowers inside the bees vision

fn The minimum number of forgotten flowers

G Graph

K Total components of PCB

L The total length of a tour

m Number of best sites in the Bees Algorithm

$MaxEv$ The maximum function evaluation number (stopping criteria)

N Number of decision variables

n Number of scout bees in the Bees Algorithm

nep Number of bees recruited for e sites in the Bees Algorithm

ngh Size of patches including site and its neighbourhood in the Bees Algorithm

nsp Number of bees recruited for $(m - e)$ sites in the Bees Algorithm

pl The number of players in Domino Sequence

q_v The capacity of vehicle k

R^N Solution space in real value

s The obtained solution

s^* The global optimum solution

$StdDev$ Standard Deviation

V The total number of vehicles

Vx Vertex

x Decision variable(s), can be continuous, discrete or a mixture of both

x_{max} The upper bound of the solution space

x_{min} The lower bound of the solution space

Contents

Abstract **vii**

Acknowledgements **ix**

Abbreviations, Acronyms, and Symbols **xvi**

List of Figures **xxvi**

List of Tables **xxx**

1 Introduction **1**

1.1 Background 1

1.2 Aim and objectives 3

1.2.1 Aim 3

1.2.2 Objectives 3

1.3 Methodology 4

1.4 Thesis outline 4

2 Literature Review **7**

2.1 Background 7

2.2 Optimisation and its classification 7

2.2.1 Heuristics 9

2.2.2 Metaheuristic algorithms 14

2.2.3 Bees Algorithm 25

2.2.4 Applications of the Bees Algorithm 35
2.3 Behaviour of bees in nature 36
2.3.1 Bees in nature: the behaviours of food foraging and nest site selection . 36
2.3.2 Bees traplining foraging behaviour in nature 39
2.4 Population-based metaheuristic development 41
2.5 Measuring metaheuristic performance 42
2.6 Summary 44

3 Reduction in the number of Bees Algorithm parameters using triangular distribution by integrating exploration and exploitation 47
3.1 Preliminaries 47
3.2 Benchmark versions of Bees Algorithm 49
3.2.1 Enhanced Bees Algorithm (2008) 49
3.2.2 Standard Bees Algorithm (2009) 49
3.3 Searching mechanism of Bees Algorithm 50
3.4 Bees do exploration and exploitation in the same foraging trip 52
3.5 The triangular distribution searching mechanism 53
3.6 The proposed algorithm 55
3.7 Experiments, results and discussion 58
3.7.1 Experiment design 58
3.7.2 Experiment result 61
3.8 Summary 66

4 Novel exploration strategy of Combinatorial Bees Algorithm 69
4.1 Preliminaries 69
4.2 Bees' initial foraging tour behaviour in nature 71
4.3 A Combinatorial Bees Algorithm with the bee's strategy for tour construction . 73
4.4 Experiments, results and discussion 76
4.4.1 Preliminary experiment 77

4.4.2 Main experiment 83
4.4.3 Discussion 86
4.5 Summary 87

5 Novel exploitation strategy of Combinatorial Bees Algorithm 89
5.1 Preliminaries 89
5.2 Bees' behaviour of avoiding threats in nature 90
5.3 Proposed exploitation search or intensification strategy 91
5.4 Experiments, results and discussion 95
5.4.1 Performance studies of exploration vs exploitation strategy 96
5.4.2 Comparative analysis of algorithms 99
5.5 Summary 108

6 Two engineering applications 111
6.1 Printed circuit board assembly sequence optimisation 111
6.1.1 Preliminaries 111
6.1.2 Problem definition 113
6.1.3 The Combinatorial Bees Algorithm for PCB insertion sequence optimisation 115
6.1.4 The experiment results 116
6.2 Vehicle Routing Problem 123
6.2.1 Preliminaries 123
6.2.2 Problem definition 123
6.2.3 The Combinatorial Bees Algorithm for VRP 124
6.2.4 The experiment results 128
6.3 Summary 135

7 Conclusion, contributions and future work 137
7.1 Conclusion 137
7.2 Contributions 139

7.3 Future work 140

A The data of Chapter 2 **141**

B The results of Chapter 3 **153**

C The results of Chapter 4 **163**

D The results of Chapter 5 **179**

E The results of Chapter 6 **191**

F MATLAB code of Bi-Parameters Bees Algorithm (Bi-BA) for numerical function **203**

G MATLAB code of Combinatorial Bees Algorithm with Bees Nearest Straight Neighbour (BNSN) and/or Bees Routing Optimiser (BRO) for travelling salesman problem **223**

H MATLAB code of Combinatorial Bees Algorithm with BNSN using seed (Domino Sequence Heuristic) for PCB insertion sequence optimisation **275**

I MATLAB code of Combinatorial Bees Algorithm with BNSN without seed for PCB insertion sequence optimisation **289**

J MATLAB Code of Combinatorial Bees Algorithm with Bi-BA (Clustering) and BRO for capacitated vehicle routing problem **303**

References **327**

List of Figures

2.1 Taxonomy of optimisation methods 8
2.2 Procedure of nearest neighbour 10
2.3 Procedure of Domino sequence heuristic 11
2.4 Four neighbour solutions of A-B-C-D-E-F with 3 edge exchange 13
2.5 The developments of the Bees Algorithm 29
2.6 Developments of the Combinatorial Bees Algorithm 31
2.7 Percentages of applications using the Bees Algorithm by specialised area . . . 35
2.9 The Bees Algorithm with other metaheuristics (Hybrid) 36
2.10 Top 20 of the Bees Algorithm users by Country 37
2.11 Exploration (diversification) versus exploitation (intensification) 42
2.12 The performance assessment of a solution (minimisation problem) 42
2.13 The selection procedure of a statistical test 43
2.14 The visualisation of hypothesis statistical test 44

3.1 The bee's foraging behaviour (Woodgate *et al.*, 2017) 53
3.2 The ratio of exploration and exploitation of 5 different patches 54
3.3 The triangular distribution on the interval $[a, c]$ with mode b 54
3.4 The searching of the basic Bees Algorithm using the uniform distribution . . . 55
3.5 The triangular distribution searching assignment 55
3.6 The flowchart of the Basic and Standard(*) Bees Algorithm with a uniform distribution searching mechanism 56
3.7 The flowchart of the Enhanced Bees Algorithm 57

3.8 The flowchart of the Bi-Parameter (Standard=*) Bees Algorithm (Bi-(S)BA) . . 58
3.9 The linear interpolation of the recruitment mechanism of Bi-(S)BA 59
3.10 Statistical test (95% confidence level) on the accuracy of Bi-BA 62
3.11 Statistical test (95% confidence level) on the number of function evaluations (NFE) of Bi-BA 62
3.12 Statistical test (95% confidence level) on the accuracy of Bi-SBA 66
3.13 Statistical test (95% confidence level) on the NFE of Bi-SBA 67

4.1 Bees consider the distance and angle 72
4.2 The experiment arrays (Ohashi *et al.*, 2007) 72
4.3 Constructive procedure with a distance and angle consideration, the Bee's Nearest Straight Neighbour (BNSN) 73
4.4 Measuring the bees turning angle and penalty 75
4.5 Bee thinks distance and angle 76
4.6 Circuits of the difference constructive heuristics and random number generator (RNG) 79
4.7 The diversity of the constructive heuristics and RNG 82
4.8 Kruskal Wallis ANOVA test on the heuristic's diversity (1=RNG; 2=BNSN(1); 3=BNSN(2); 4=BNSN(3)) 82
4.9 Accuracy performance of all the Combinatorial Bees Algorithm versions 84
4.10 Kruskal Wallis ANOVA test on the accuracy of all versions of CBA for 15 datasets (1=RNG; 2=BNSN(1); 3=BNSN(2); 4=BNSN(3)) 84
4.11 Kruskal Wallis ANOVA test on the NFE of all versions of CBA for 15 datasets (1=RNG; 2=BNSN(1); 3=BNSN(2); 4=BNSN(3)) 84
4.12 The best solution tours of CBA+BNSN for [51,200] TSPs' instances 88

5.1 (a): The completed (paths) distance information; (b)-(f): the optimisation behaviour of bees 92

5.2 (a) Trap in local optima, (b) a forgotten node and sharp turning path inside the remaining nodes, (c) swap nodes e and g, (d) swap nodes f and g (e) a habitual tour and the re-introduction of the forgotten node, (f) inserting the forgotten node to the habitual tour. 92

5.3 Insertion procedures (a) 2 minimum distance of forgotten nodes and habitual path (DFH) with long and short edges (b) inserting vertex-1 to edge ab generates optimal tour (c) inserting vertex-1 to edge cd generates near-optimal tour. 94

5.4 The flowchart of The Bees Algorithm with Bee's Nearest Straight Neighbour (BNSN) and Bees Routing Optimiser (BRO) 95

5.5 Kruskal Wallis ANOVA test on the accuracy of Exploration Vs Exploitation strategies of CBA for 15 datasets [50, 200] (1 = CBA; 2 = CBA+NNH; 3 = CBA+BNSN(2); 4 = CBA+BNSN(3); 5 = CBA+BRO) 99

5.6 Kruskal Wallis ANOVA test on the NFE of Exploration Vs Exploitation Strategy of CBA for 15 datasets [50,200] (1=CBA; 2=CBA+NNH; 3=CBA+BNSN(2); 4=CBA+BNSN(3); 5=CBA+BRO) 99

5.7 Statistical test (95% confidence level) on the accuracy performance of CABC Vs qCABc Vs CBA+BNSN+BRO 105

5.8 The best solution tours of CBA+BRO for [51,200] TSPs' instances 109

6.1 PCB assembly machine of the MBTD type (with 2 Rotary Turret Heads, 10 Feeder Slots and a Move-able Assembly Table) (Castellani *et al.*, 2019) 112

6.2 The positions of all ten components on the board 114

6.3 Flowchart of CBA with seed for PCB assembly optimisation 117

6.4 Flowchart of CBA without seed for PCB assembly optimisation 118

6.5 The best solution of Castellani *et al.* (2019) (23.46 s assembly time) 119

6.6 The best solution of CBA+BNSN using seeds (23.00 seconds assembly time) . 120

6.7 The best solution of CBA+BNSN (24.4167 seconds assembly time) 120

6.8 Solution representation of routing problems 123

6.9 CBA for Vehicle Routing Problem 125
6.10 Combinatorial Bees Algorithm using Bi Parameter BA Clustering and BRO for Vehicle Routing Problem 126
6.11 Clustering 51 Customer (Eil51) into 5 clusters with total demand below or equal to 160 for each cluster 128
6.12 Result of the best solution of CBA+Bi-BA+BRO for [33,51] VRPs' instances . 133
6.13 Result of the best solution of CBA+Bi-BA+BRO for 75 dimension VRPs' instances 134
6.14 Result of the best solution of CBA+Bi-BA+BRO for 100 dimension VRPs' instances 135

B.1 Global view of Statistical test (95% confidence level) on the accuracy of Bi-BA 153
B.2 Global view of Statistical test (95% confidence level) on the number of function evaluations (NFE) of Bi-BA 154
B.3 Comparison of best performance of Bi-SBA Vs SBA (confidence interval (C.I.) = 95%) 155
B.4 The Comparison of Robust Performance - Global view (Bi-SBA Vs SBA) . . . 156
B.5 Global view of the comparison of robust accuracy performance with C.I. 95% (Bi-SBA Vs SBA) 157
B.6 Global view of the NFE robust performance with C.I. 95% (Bi-SBA Vs SBA) . 158
B.7 Global view of the statistical test (95% confidence level) on the accuracy of Bi-SBA 159
B.8 Global View of the statistical test (95% confidence level) on the NFE of Bi-SBA 160
B.9 Comparison of robust accuracy performance with C.I. 95% (Bi-SBA Vs SBA) . 161
B.10 The NFE robust performance with C.I. 95% (Bi-SBA Vs SBA) 161

C.1 Statistic test Constructive Heuristic (1= RNG; 2= BNSN(1); 3= BNSN(2); 4= BNSN(3)) 171

C.2 Statistic test TSPLIB [50,76] dimensions, (1= RNG; 2= BNSN(1); 3= BNSN(2); 4= BNSN(3)) 172

C.3 Statistic test TSPLIB [99,100] dimensions, (1= RNG; 2= BNSN(1); 3= BNSN(2); 4= BNSN(3)) 173

C.4 Statistic test TSPLIB [150,200] dimensions, (1= RNG; 2= BNSN(1); 3= BNSN(2); 4= BNSN(3)) 174

C.5 NFE Statistic test TSPLIB [50,76] dimensions, (1= RNG; 2= BNSN(1); 3= BNSN(2); 4= BNSN(3)) 175

C.6 NFE Statistic test TSPLIB [99,100] dimensions, (1= RNG; 2= BNSN(1); 3= BNSN(2); 4= BNSN(3)) 176

C.7 NFE Statistic test TSPLIB [150,200] dimensions, (1= RNG; 2= BNSN(1); 3= BNSN(2); 4= BNSN(3)) 177

D.1 Result of the best solution of CBA+BRO for selected [144,1577] TSPLIB's datasets 184

D.2 Statistic Kruskal Wallis test of Exploration Vs Exploitation strategy (Accuracy) [50,76] datasets, (1 = CBA; 2 = CBA+NNH; 3 = CBA+BNSN(2); 4 = CBA+BNSN(3); 5 = CBA+BRO) 185

D.3 Statistic Kruskal Wallis test of Exploration Vs Exploitation strategy (Accuracy) [99,100] datasets, (1 = CBA; 2 = CBA+NNH; 3 = CBA+BNSN(2); 4 = CBA+BNSN(3); 5 = CBA+BRO) 186

D.4 Statistic Kruskal Wallis test of Exploration Vs Exploitation strategy (Accuracy) [150,200] datasets, (1 = CBA; 2 = CBA+NNH; 3 = CBA+BNSN(2); 4 = CBA+BNSN(3); 5 = CBA+BRO) 187

D.5 The Kruskal Wallis test of Exploration Vs Exploitation strategy (NFE) [50,76] datasets, (1 = CBA; 2 = CBA+NNH; 3 = CBA+BNSN(2); 4 = CBA+BNSN(3); 5 = CBA+BRO) 188

D.6 The Kruskal Wallis test of Exploration Vs Exploitation strategy (NFE) [99,100] datasets, (1 = CBA; 2 = CBA+NNH; 3 = CBA+BNSN(2); 4 = CBA+BNSN(3); 5 = CBA+BRO) 189

D.7 The Kruskal Wallis test of Exploration Vs Exploitation strategy (NFE) [150,200] datasets, (1 = CBA; 2 = CBA+NNH; 3 = CBA+BNSN(2); 4 = CBA+BNSN(3); 5 = CBA+BRO) 190

E.1 The result of CBA (seed) with Domino Sequence Heuristic under 23.4 sec . . . 192

E.2 The 50 placement locations 194

List of Tables

2.1 The Benchmark Functions 32

2.2 The Benchmark Functions' Interval and optimum solutions 32

3.1 The scenarios of the comparative studies 60

3.2 The summary of accuracy results of the first comparative study of Bi-BA 63

3.3 The summary of evaluation results of the first comparative study of Bi-BA . . . 63

3.4 The best performance of Bi-SBA Vs SBA 65

3.5 Comparison of robustness (Bi-SBA Vs SBA) 65

4.1 The Initial solution of Random Number Generator (RNG), BNSN-1 (FL=1; Δr=[0,1]), BNSN-2 (FL=2; Δr=[0,1]), and BNSN-3 (FL=3; Δr=[0,1]) 80

4.2 The Initial solution's diversity level of RNG, BNSN-1 (FL=1; Δr=[0,1]), BNSN-2 (FL=2; Δr=[0,1]), and BNSN-3 (FL=3; Δr=[0,1]) 81

4.3 Parameters setting of CBA+BNSN 83

4.4 The comparison of 3 scenarios of CBA+BNSN 85

5.1 Parameter setting of the CBA+BNSN+BRO for comparative study 96

5.2 Accuracy results of 10 runs of CBA+BRO using ten times dimension of colony size 98

5.3 A comparison of BCO, ABC, and CBA 101

5.4 Accuracy results of 10 runs of CBA+BNSN(1,2,3)+BRO using a colony size of 40 bees 102

5.5 The NFE results of 10 runs of CBA+BNSN(1,2,3)+BRO using a colony size of 40 bees 103
5.6 Accuracy comparison of CBA Vs BCO for [1173,1379] datasets 106
5.7 Accuracy comparison of CBA Vs other nature-inspired algorithms 106
5.8 The convergence comparison of CBA Vs ABC 107

6.1 MBTD benchmark: the comparison of the results obtained between the CBAs . 119
6.2 The simulation data for the PCB assembly of 50 components obtained using the CBA+BNSN using seeds with a total assembly time is 23.00 seconds 121
6.3 The simulation data for the PCB assembly of 50 components obtained using the CBA+BNSN without seed with a total assembly time is 24.4167 seconds 122
6.4 Parameter setting of CBA for Vehicle Routing Problem 127
6.5 Parameter setting of Bi-BA and BRO for Vehicle Routing Problem 127
6.6 The best result of CBA + Bi-BA + BRO for VRP (Eil33) 129
6.7 The best result of CBA + Bi-BA + BRO for VRP (Eil51) 129
6.8 The best result of CBA+ Bi-BA + BRO for VRP (EilA76) 129
6.9 The best result of CBA+ Bi-BA + BRO for VRP (EilB76) 130
6.10 The best result of CBA + Bi-BA + BRO for VRP (EilC76) 130
6.11 The best result of CBA + Bi-BA + BRO for VRP (EilD76) 130
6.12 The best result of CBA + Bi-BA + BRO for VRP (A101) 131
6.13 The best result of CBA + Bi-BA + BRO for VRP (B101) 131
6.14 Comparison results of VRP instances 132

A.1 The Applications of Bees Algorithm 142

B.1 The result of Bi-BA with Scenario 1-1 (Bi-BA-1) 154
B.2 The result of Bi-BA with Scenario 1-2 (Bi-BA-2) 154
B.3 The result of Bi-BA with Scenario 1-3 (Bi-BA-3) 155
B.4 The result of Bi-SBA with Scenario 2-1 (Bi-SBA-1) 156
B.5 The result of Bi-SBA with Scenario 2-2 (Bi-SBA-2) 157

B.6 The result of Bi-SBA with Scenario 2-3 (Bi-SBA-3) 158
B.7 The result of Bi-SBA with Scenario 2-4 (Bi-SBA-4) 159
B.8 The result of Bi-SBA with Scenario 2-5 (Bi-SBA-5) 160

C.1 The diversity of RNG 164
C.2 The diversity of BNSN(F=1;Δr = [0,1]) 164
C.3 The diversity of BNSN(F=2;Δr = [0,1]) 164
C.4 The diversity of BNSN(F=3;Δr = [0,1]) 165
C.5 The error result (%) of RNG, BNSN(1), BNSN(2), BNSN(3) on 15 datasets TSPLIB for 10 independent runs 165
C.6 The result of basic BA on 15 datasets TSPLIB for 10 independent runs 165
C.7 the error results of 10 runs of basic CBA 166
C.8 The result of CBA+BNSN(1) on 15 datasets TSPLIB for 10 independent runs . 166
C.9 the error results of 10 runs of CBA+BNSN(1) 167
C.10 The result of CBA+BNSN(2) on 15 datasets TSPLIB for 10 independent runs . 167
C.11 the error results of 10 runs of CBA+BNSN(2) 168
C.12 The result of CBA+BNSN(3) on 15 datasets TSPLIB for 10 independent runs . 168
C.13 the error results of 10 runs of CBA+BNSN(3) 169
C.14 The evaluation (NFE) result of basic CBA on 15 datasets TSPLIB for 10 independent runs 169
C.15 The evaluation (NFE) result of CBA+BNSN(1) on 15 datasets TSPLIB for 10 independent runs 170
C.16 The evaluation (NFE) result of CBA+BNSN(2) on 15 datasets TSPLIB for 10 independent runs 170
C.17 The evaluation (NFE) result of CBA+BNSN(3) on 15 datasets TSPLIB for 10 independent runs 171

D.1 The results of 10 runs of CBA+BRO 180
D.2 The error results of 10 runs of CBA+BRO 180

D.3 The NFE results of 10 runs of CBA+BRO 181
D.4 the results of 10 runs of CBA+BNSN(1)+BRO using colony size of 40 bees . . 181
D.5 The results of 10 runs of CBA+BNSN(2)+BRO using colony size of 40 bees . . 181
D.6 The results of 10 runs of CBA+BNSN(3)+BRO using colony size of 40 bees . . 182
D.7 The NFE results of 10 runs of CBA+BNSN(1)+BRO using a colony size of 40 bees 182
D.8 The NFE results of 10 runs of CBA+BNSN(2)+BRO using a colony size of 40 bees 182
D.9 The NFE results of 10 runs of CBA+BNSN(3)+BRO using colony size of 40 bees 183

E.1 The simulation result of the PCB assembly of 50 components obtained using the Bees algorithm with Domino operators for a CS machine with a twin assembly head turret and 10 component feeders where the total assembly time is 23.33s. . 192
E.2 Bees Algorithm + BNSN + Domino (1-15 running experiments) 194
E.3 Bees Algorithm + BNSN + Domino (16-30 running experiments) 196
E.4 Bees Algorithm + BNSN (1-15 running experiments) 197
E.5 Bees Algorithm + BNSN (16-30 running experiments) 199
E.6 CBA + Bi-BA + BRO for VRP (10 running experiments) 201

Chapter 1

Introduction

This chapter discusses the research's motivations, objectives, and methods. It also presented the thesis's structure.

1.1 Background

Optimisation plays a critical role in human life, from recipe mixing to genome sequence generation. As human civilisation progresses into modernity, especially in the industrial world, problems arise and become more complicated. The increasing sophistication of problems drives the optimisation approaches away from exact conservative methods and more towards nature-inspired or metaheuristic algorithms. It is inevitable that this nature-inspired algorithm would be used to solve optimisation problems using technology such as computers or robots. It inevitably leads to the current development of artificial intelligence (AI).

The rapid advancement of AI has increased the importance of approximation algorithms and diminished the need for exact methods within the optimisation culture. The advancement of computer science technologies enchanted and strengthened this new intention. Furthermore, the complexity of real-world system problems diminishes the effectiveness of the exact algorithm as the primary method for solving the problems. It shows that exact algorithms are not often the most efficient approach for solving complex problems with multiple dimensions, a large degree of nonlinearity, and extreme constraints, even though they guarantee the optimal value solution.

The new age required increased productivity from manufacturing organisations. A second's delay could lead to a million pounds' loss. This consideration elevates the speed as a success factor for metaheuristics to a higher degree than previously. It prefers the faster (calculation) of an algorithm with a near-optimal value over the slower one, even though the slower one produces a global optimum. However, some scholars disregard the speed performance because computer technology is getting faster in computing.

The metaheuristic, or nature-inspired algorithm, was first introduced to the public in the 1960s and 1970s as Evolutionary Strategy (ES) and Genetic Algorithm (GA). The swarm class of metaheuristics was pioneered in the 1990s by Ant System (AS) and Particle Swarm Optimisation (PSO). Following that, a flood of metaheuristics algorithms were released and quickly gained attention.

Pham *et al.* (2005) added the Bees Algorithm (BA) to the community. The algorithm is classified as a population-based and nature-inspired metaheuristic algorithm. The algorithm serves as a metaphor for bees' foraging actions. BA first demonstrated the capacity to optimise numerical functions and then expanded to optimise combinatorial functions. Due to the BA's ability to solve problems in both realms is classified as a metaheuristics algorithm.

After 15 years, BA has developed into a successful algorithm. Moreover, developments in the continuous and combinatorial domains have improved the efficiency and effectiveness of BA. BA has authored over 500 publications and accumulated thousands of citations in order to address real-world problems. This algorithm is widely used in industries ranging from manufacturing to bioinformatics, making it one of the most common population-based intelligence algorithms. However, the developments were primarily based on hybridisation with established algorithms rather than on adopting novel ideas from nature, especially bee behaviour.

In this work, firstly, the author inspired himself to develop the BA by drawing inspiration from nature, specifically bee traplining foraging behaviours. This behaviour would encompass the ideas, methods, and principles necessary to construct the algorithm to solve complex computational problems. The development direction is toward increasing the algorithm's simplicity, precision, and time. The modified version was evaluated using 17 standard continuous do-

main benchmark functions and 25 TSPLIB datasets as the standard combinatorial test. Finally, new BA versions were implemented for real-world manufacturing-engineering applications or complicated industrial problems. These were the capacitated vehicle routing problem and the optimisation of the PCB assembly sequence.

1.2 Aim and objectives

1.2.1 Aim

This thesis aims to improve the effectiveness and efficiency of the BA by incorporating insights from bees' traplining behaviour. The results of the proposed BA versions will be compared to those of previous versions of BA and other metaheuristic algorithms in the field. The proposed versions would be implemented on example engineering applications.

1.2.2 Objectives

The objectives of this research are:

- To simplify the BA by reducing its parameters.
- To increase the likelihood of meeting global optimum conditions through diversification of the (combinatorial) BA's initial solution.
- To increase the accuracy and convergence speed of the (combinatorial) BA by avoiding local optimum traps.
- - To apply the new BA variants to established manufacturing problems, such as assembly sequence planning and vehicle route planning.

1.3 Methodology

The methodology used in this research is as follows:

- Review previous studies: a comprehensive survey of the state of the art intelligent optimisation strategies, with a particular emphasis on BA, was conducted in order to map the progress of BA developments.
- Investigate the bees traplining foraging behaviour: a review of bees behaviour was performed to identify potential solutions.
- Identify the research gaps: the development topic has to be identified to conclude a question.
- - Develop improved BA variants: the basic BA was modified by mimicking the bees traplining foraging behaviour.
- - Evaluate the new BA variants: The new variants of the algorithm were compared against other algorithms using standard test functions. Validation and verification procedures included applying the new versions to engineering problems.

1.4 Thesis outline

The rest of this thesis is presented as follows:

- Chapter 2: Reviews the literature on the BA as a nature-inspired metaheuristic algorithm and its applications, as well as the literature on other nature-inspired metaheuristic algorithms and the natural behaviour of bees.
- Chapter 3: Describes the development of BA relating to parameter reduction. This chapter examines the concurrent processes that occur during the bees' traplining behaviour. The primary exploitation and exploration processes are integrated in order to reduce the initial BA parameter.

- Chapter 4: Describes the development of BA in terms of the initialisation exploration approach. This chapter discusses how to determine the next visitation when bees generate initial foraging circuits in their traplining behaviour. To construct the initial solution, the second consideration of turning angle would be added.

- Chapter 5: Describes the development of BA's exploitation strategy in the neighbour search mechanism. This chapter examines how bees avoid a threat through their traplining behaviour. Refusing to visit certain threatening flowers would be used to evade a local optima trap.

- Chapter 6: Applies the BA to engineering problems, PCB assembly optimisation and vehicle routing problem.

- Chapter 7: Summarises the significant contributions and results of this work and makes recommendations for future research.

Chapter 2

Literature Review

2.1 Background

This chapter reviews the Bees Algorithm (BA) as an intelligent optimisation tool and its contributions to applications in multiple disciplines. The chapter focuses on work on the BA to date, both in the continuous and the combinatorial domains.

Most of the real-world applications involve an optimisation tool to solve their large complex problems. However, many algorithms are considered inefficient due to their computationally expensive operation. Even the modern computer is impractical to use enumeration approaches such as the branch and bound for more extensive dimension problem. Nature-inspired algorithms (metaheuristics) recently became a trend because they are simple, flexible, and surprisingly efficient (Yang, 2020). The fast growth and development of new metaheuristic algorithms recently are shown by 36 well establish algorithms that have reached more than 200 citations. Fifteen of them had over 1,000 citations (Lones, 2020), like BA, which was used in this work.

2.2 Optimisation and its classification

Optimisation is the study of how to determine the "best" solution to a complicated problem. The optimal solution is the minimum or maximum point in the solution space. Applications of optimisation can be found in many fields, from engineering to social sciences. These applications

comprise both domains of optimisation problems, continuous and combinatorial.

The classification or taxonomy of optimisation could help get a clear picture about it, but it is not the only classification approach. One category is based on the types of problems or their variables (see Figure 2.1). Real variables are used in the continuous optimisation problem, and discrete variables are used in combinatorial optimisation problems. They both could be a single or multi-objective problem.

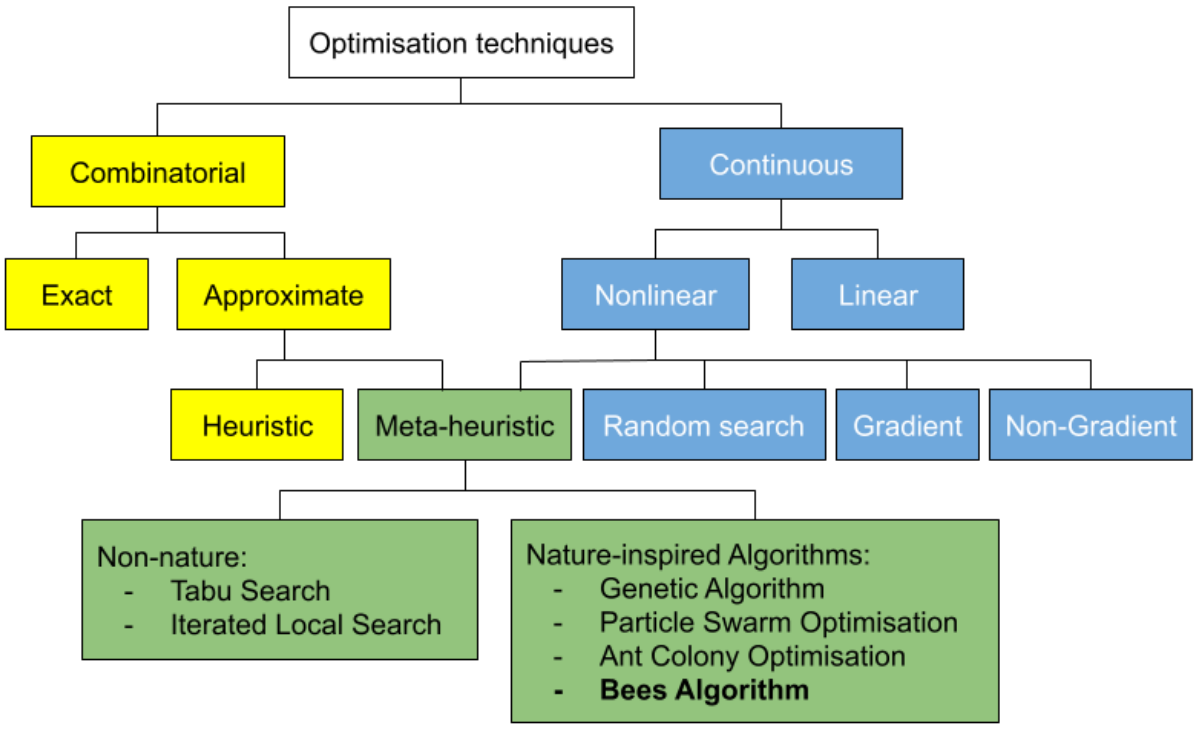


Figure 2.1: Taxonomy of optimisation methods

A typical continuous problem is usually involves numerical function optimisation. The travelling salesman problem (TSP), the sequencing problem, scheduling, the minimum spanning tree problem (MST), and the knapsack problem are all examples of common combinatorial problems. The brute force algorithm is an example of the exact method of a combinatorial way that evaluates every possible solution. On the other side, the continuous approach allows the calculus (gradient) techniques for the non-linear problem. Both exact methods seem impractical for more significant complex problems. The brute force method is extremely slow as the combinatorial solution-space grows factorially (Pham *et al.*, 2006c). At the same time, the gradient is very hard to derive for higher dimension problems.

Alternatively, the approximation algorithms seek an approximation that is near to the optimal solution. The approximation can be reached by either using an entirely random strategy or regulated it with a procedure. The simplicity and quick calculation of approximation algorithms are the main reason for this enormous intention (Yang, 2020). There are two big classes of this approach: heuristic and metaheuristic.

2.2.1 Heuristics

A heuristic is any approach to tackle a problem utilising a viable strategy that's not guaranteed to be optimal but considered adequate for reaching a fast estimation. In the 1970s and 1980s, the heuristic becomes popular because of its high accuracy and speed performance. This trend strengthens the opinion that there is no algorithm suitable for all problems, and later, Wolpert and Macready (1997) proved that this opinion likely to be true. Fundamentally, this approach can be divided into constructive and improvement heuristic.

Constructive heuristic

A constructive heuristic is a heuristic technique that starts with a null solution and incrementally attaches the current one before the current solution becomes complete. It differs from improvement (local search) heuristics which begins with a complete solution and improves the current solution with slight (local) moves.

This constructive heuristic may be classified as either greedy or non-greedy. The greedy method would often design the solution depends on the ideal solution for each step of development. There are many well-known greedy building algorithms, including the nearest neighbour, Dijkstra's algorithm (path planning), Kruskal's algorithm, and Prim's algorithm (minimum spanning trees) (Cormen *et al.*, 2009). While the non-greedy may consider comparatively lesser optimal solution on the solution development.

In TSP, the constructive heuristic serves a good but not optimal solution in a short time. One of the most popular heuristic (constructive) is the nearest neighbour. It was one of the first approach used to tackle the TSP approximately. It also believed as the basic instinct of insects

to forage their food. The procedure orders the operator to pick whatever is currently the best next step. The agent begins at a random vertex and repeatedly connects the nearest vertex until all have been attached (see Figure 2.2).

The latest nearest neighbour class algorithm called Domino sequence heuristic is a constructive heuristic that follows the procedure of domino games when creating the sequence (Ismail, 2019). Unlike the traditional nearest neighbour, the DSH consists of the number of players as an initial parameter (usually 2-4 players) to construct a solution. Every player has their own list of vertices, and they build a solution based on their turn extends the current solution until it becomes a complete sequence. The players generate a sequence from a set of matching tiles in which each tile represents a path consisting of 2 points as original and destination cities. A player would have the choice of forwarding or backwards construction. The new member of the vertex, the extension of the current solution, will be attached to the front (backward construction) or end (forward construction) of the uncompleted sequence (solution). If the number of players is set to one, the DSH behaves similarly to the conventional nearest neighbour (see Figure 2.3).

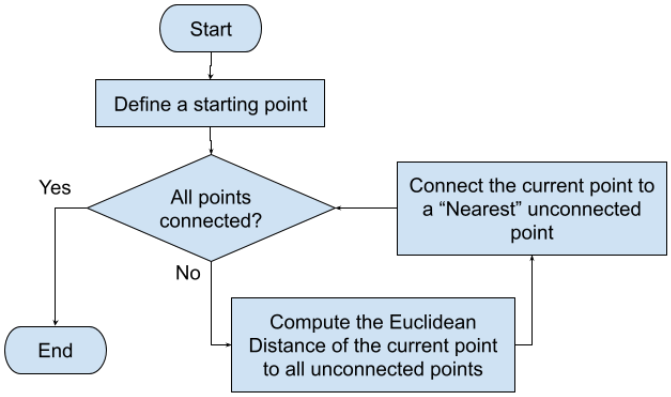


Figure 2.2: Procedure of nearest neighbour

The DSH is emulating the domino puzzle. It is a classic European tile-based strategy game that two to four players usually play. Each domino tile is rectangular, dividing it into two equal square-shaped halves: the front half and the back half. Either half is denoted by a blank or one to six dots. This is how the game is played. The first player chooses a tile; the second player chooses a corresponding tile with a similar value in its front or end half as the first

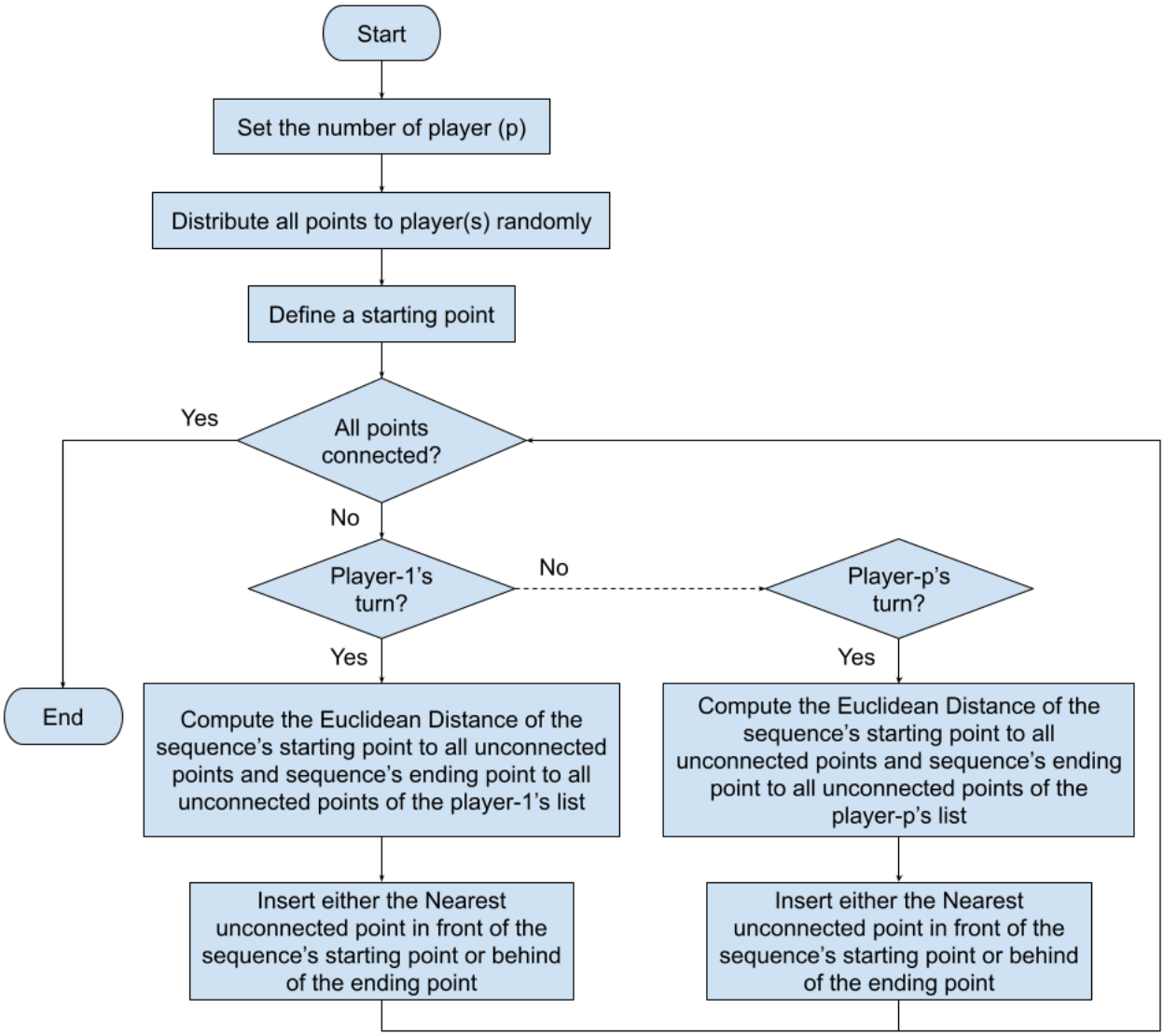


Figure 2.3: Procedure of Domino sequence heuristic

player's tile, thus forming a sequential series of matching tiles. They will continue in this manner until the game is completed. In general, this game consists of six major steps: shuffling the dominoes, drawing an open hand, determining the order of play, laying the first domino, taking turns adding dominoes, and concluding the round by awarding points. The procedure poses in Algorithm 1.

Improvement heuristic

The other kind of heuristic is the local search algorithm, which iteratively moves from one solution to another in the neighbour space of solutions until a near-optimal solution is found or

Algorithm 1: Domino Sequence Heuristic

```
input  : pl : is the number of players.
output: T = a tour of k cities
Start;
DominoesList ← random.permutation (k);
DominoesList ← resize (pl, k/pl);
T ← [ ];
I ← random.integer (k);
T.append(I);
DominoesList.remove(I);
while DominoesList ≠ 0 do
    for a in range (pl) do
        EarlyD ← Distance (DominoesList, T(0));
        MinEarlyD ← min(EarlyD);
        TardyD ← Distance (DominoesList, Tour(end));
        MinTardyD ← min (TardyD);
        if MinEarlyD < MinTardyD then
            Index ← DominoesList.Index(MinEarlyD);
            T.insert(0, Index);
        else
            Index ← DominoesList.Index(MinTardyD);
            T.append(Index);
        end
        DominoesList.remove(Index);
    end
end
End
```

a time criterion is met. The idea is by keeping the current solution and try to improve it. The benefits are the solution can be traced and used a very little memory. The most fundamental local search algorithm is the hill-climbing method, referred to as greedy local search due to its focus on nearby good neighbours. The investigation moves according to the increasing elevation value to find the top of the mountain. The algorithm will terminate if it reaches the peak. The issue could be the same with metaheuristic as the peak could be the global optimum or stuck on the local optima.

The local search algorithm's combinatorial domain includes simple swapping, insert, reversion or exchange, 2-Opt, 3-Opt, and Lin–Kernighan (LK). The last one is one of the most effective heuristics for solving the symmetric travelling salesman problem. It is a generalisation of the 2-Opt and 3-Opt operations, in which two or three edges are switched to shorten the

tour. In this study, the two edges reversion or exchange differs from 2-Opt. Moreover, three edges exchange with 3-opt. Although both pair methods used the identical move, the reversion or exchange does not repeat the procedure until no improvement like the 2-Opt or 3-Opt. The procedures of reversion (2 edges), insertion and swap could be seen in Equation 2.1 to 2.3. The pseudocode of 2-Opt could be seen in Algorithm 2.

$$Reverse := A - B - C - D - E - F \rightarrow A - E - D - C - B - F \tag{2.1}$$

$$Insert := A - B - C - D - E - F \rightarrow A - C - D - E - B - F \tag{2.2}$$

$$Swap := A - B - C - D - E - F \rightarrow A - E - C - D - B - F \tag{2.3}$$

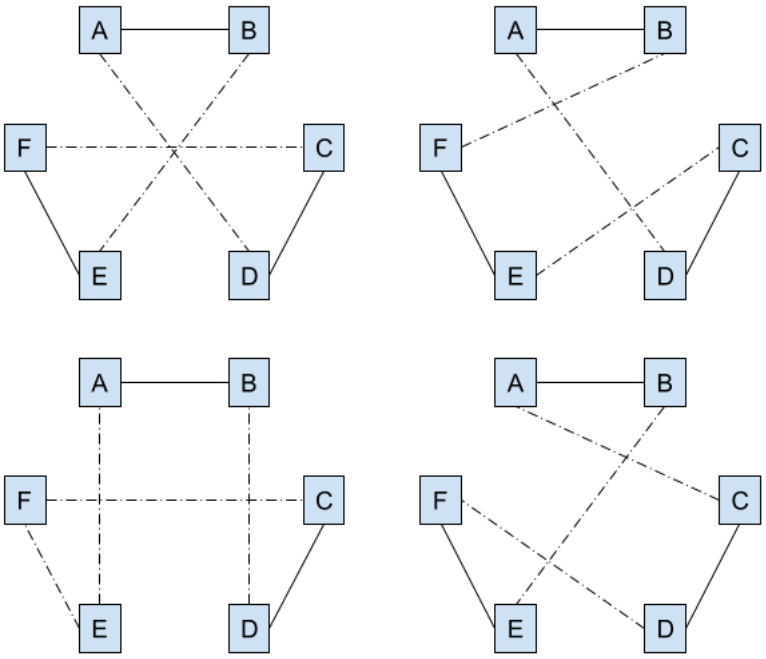


Figure 2.4: Four neighbour solutions of A-B-C-D-E-F with 3 edge exchange

The 3-Opt algorithm has a similar procedure to the 2-Opt algorithm (see in Algorithm 2) that checks for all possible combinations. The algorithm reconnects the deleted three-edge combinations to shorten the tour (see Figure 2.4). The 2-Opt, 3-Opt, and k-Opt is the iterative improvement algorithm based on 2, 3, and k edges exchange. While LK is a generalisation of 2-opt and 3-opt that is adaptive. It involves flipping two or three edges to shorten the tour.

The following section will discuss a metaheuristic or some refer to the nature-inspired al-

Algorithm 2: 2-Opt pseudocode

```
1  start;
2  best_solution = Distance(current_route);
3  for all eligible edges (i, j) do
4  |   new_route = Reverse (current_route, i, j);
5  |   new_distance = Distance(new_route);
6  |   if new_distance < best_solution then
7  |   |   current_route = new_route;
8  |   |   best_solution = new_distance;
9  |   |   goto start;
10 |   else
11 |   |   goto start;
12 |   end
13 end
```

gorithm which the basic principle is combining heuristics (constructive and local search) in higher-level frameworks.

2.2.2 Metaheuristic algorithms

Some definitions of metaheuristic resumed by Blum and Roli (2003):

"A metaheuristic is characterised formally as an iterative generation mechanism that guides a subordinate heuristic by intelligently merging disparate principles for exploring and exploiting the search space, and learning techniques are used to organise information to find near-optimal solutions efficiently." (Osman and Laporte, 1996).

"A metaheuristic is a main iterative mechanism that directs and modifies the operations of subordinate heuristics to generate high-quality solutions effectively. It can manipulate a complete (or incomplete) single solution or a series of solutions at each iteration. Subordinate heuristics can take the form of high (or low) level procedures, a basic local search, or simply a construction method." (Voß *et al.*, 2012).

“Metaheuristics are usually high-level techniques that direct a more complex, underlying heuristic to improve its efficiency. The primary objective is to prevent the drawbacks of iterative improvement, namely multiple descents, by allowing the local search to escape local optima. It is accomplished by allowing for worsening steps or by intelligently creating new

beginning solutions for the local search rather than simply offering arbitrary initial solutions. Numerous metaheuristic methods are based on probabilistic judgments taken during the search process. The primary distinction between metaheuristic algorithms and pure random search is that randomness is not used randomly but in an intelligent, biassed manner." (Stützle, 1999).

The features of this tool, such as its independence from the problems, its use of only general information and insight, and its near-optimal solution with a rational time estimate, make it very appealing to manufacturing practices. Metaheuristics are assumed to be more suitable for solving real-world problems because they do not need any advanced knowledge of the problem to explore the feasible solution space. Therefore, they can be extended to various scenarios with tiny modifications (Osaba *et al.*, 2018). On the other hand, heuristic algorithms can produce faster solutions than metaheuristic algorithms, but they are not independent of the problem since they rely on problem-specific information.

In comparison to exact approaches, metaheuristics do not guarantee the discovery of a globally optimal solution. Numerous metaheuristics employ some kind of stochastic optimisation, which means that the solution found is dependent on the set of randomly generated variables. The random operator is often fitted with a perturbation theory that regulates the algorithm's intensification and diversification.

The most intuitive classification scheme for metaheuristics is based on the algorithm's roots (Blum and Roli, 2003). There are algorithms inspired by nature, such as Genetic Algorithm (GA), Particle Swarm Optimisation (PSO), and Bees Algorithms (BA), and algorithms that are not motivated by nature, such as Tabu Search (TS) and Iterated Local Search (ILS). However, another technique categorises metaheuristics, such as population-based vs single result, dynamic vs static objective functions, and memory-based vs memory-less approaches. Furthermore, the majority of the nature-inspired algorithm is a population-based metaheuristic. Also, all of the nature-inspired algorithms mentioned below are population-based algorithms.

Population-based metaheuristics share many concepts. They should be thought of as an iterative improvement in a population of solutions. The population is first initialised. After that, a new population of solutions produces. Finally, using certain selection processes, this new

population is merged into the existing one. When a given requirement is met, the search procedure comes to an end (stopping criterion). This category of metaheuristics includes algorithms including GA, PSO, ACO, and BA.

Genetic Algorithm (GA)

Holland proposed the GA in the 1970s (Holland, 1975). It was influenced by Charles Darwin's natural selection and evolution theory and the scientific context of the "schema theorem" and can be classified as an evolutionary algorithm. This inspiration has been incorporated into operators that are used to boost the fitness of the population's individuals incrementally.

The GA is a population-based metaheuristic in which each iteration corresponds to a generation. Individuals are usually manipulated as binary-coded strings by the GA. This string is analogous to a chromosome, with substrings denoted by the term gene. Each problem parameter (dimension of the search space) is indicated by a binary substring gene (Pham and Karaboga, 2012).

The population's initialisation status is determined by arbitrarily assigning separate samples from the search space to each individual in the population. Following that, the individuals are measured and assigned a fitness score using an objective function. Following that, a replication collection is produced (to form a mating pool). In its simplest form, the GA selects individuals equal to their fitness (roulette wheel), ensuring that stronger individual has a greater probability of selection.

Typically, a GA employs a crossover operator on two significant solutions and a mutation operator that arbitrarily modifies the individual contents to facilitate diversity. GAs use a probabilistic selection technique that is derived from a proportional selection. The substitution (survivor selection) is genetic in that the offsprings systemically substitute the parents. The crossover operator is built on the uniform crossover operator, while the mutation operator is based on bit flipping. Other algorithms that used the concept of evolution are Genetic Programming (GP), proposed by (Koza, 1992) and Differential Evolution (DE) by (Storn and Price, 1997). Although GA is the most widely used evolutionary computing technique, it is

believed that Evolutionary Programming (EP) and Evolutionary Strategy (ES) were presented to the world in 1962 and 1965, respectively. According to Lones (2020), The GA has reached more than 60,000 citations, and ES has more than 5,000.

Particle Swarm Optimisation (PSO)

PSO is another population-based swarm metaheuristic. PSO was developed to imitate the flocking behaviour of living beings such as birds, insects, or fish (Kennedy and Eberhart, 1995). Like GA and ACO, PSO is a metaheuristic algorithm that the individuals collaborate to find an optimal solution. Individuals interact with one another either directly or implicitly in each search direction. Originally, PSO was effectively developed to solve problems regarding continuous optimisation.

The number of individuals in PSO remains constant in the search process. Each individual is referred to as a particle and is given a velocity and position that denotes the particle's flight path and speed. Each particle has a memory feature that stores the best location it has visited (local best) and the population's overall best spot (global best) since those particles' success would impact the actions of their peers. As a result, each particle's location can shift due to its own experience and that of neighbouring particles after each iteration. Unlike the majority of population-based metaheuristics, PSO does not employ any selection operators, implying that the principle of survival of the fittest is not applied. Other than that, all particles are moving during the algorithm's execution, except for updating the velocity.

The inertia weight and acceleration coefficients control how each particle's velocity is updated. Proper selection of inertia weights and acceleration coefficients will balance global and local search. A significant inertia weight value results in global exploration despite local exploitation with a limited inertia weight value (Engelbrecht, 2013). According to Lones (2020), The PSO has reached more than 50,000 citations.

Ant Colony Optimisation (ACO)

ACO is a population swarm-based algorithm (Dorigo and Di Caro, 1999). Prior to ACO, Dorigo suggested Ant System (AS) (Dorigo *et al.*, 1991) and Ant Colony System (ACS) (Dorigo and Gambardella, 1997). The fundamental idea behind ACO is to solve optimisation problems by imitating the cooperative behaviour of real ants. Initially, It was suggested as a novel algorithm for solving the combinatorial optimisation. The ACO's job is to find the optimal sequence of parameters in a combinatorial problem to minimise the cost function. The sequence of parameters is analogous to a path with several nodes, each node corresponding to a parameter of the solution.

ACO algorithms are based on the idea that an ant colony will find the shortest path between two points using a fundamental communication method when they transport their food. This search starts with a scattershot exploration of the area around the colony nest. When an ant comes across a food supply, it transports a portion of the food back to the colony nest and deposits a natural chemical compound called pheromone on the ground. The pheromone is used to guide other ants in locating the food supply (Dorigo and Blum, 2005). The more pheromone on a line, the more likely the ants would choose that path. One of the pheromone's properties is its evaporation over time. This result decreases the amount of pheromone deposited along the route to the food source.

Compared to other metaheuristics, a critical component of ACO is the ant's differential path length (DPL) consequence. Ants' decentralised and asynchronous design is crucial for solving distributed problems in which the objective function is not global. Decisions must be made from a local perspective on the problem.

Due to the fact that several practical optimisation problems can be expressed as continuous optimisation problems and the original works are in the combinatorial domain, there is a strong interest in creating ACO for continuous domain, the best of which is ACOR (ACO for real-valued continuous optimisation), which makes use of a continuous probability density feature (Socha, 2009). According to Lones (2020), The ACO has reached more than 10,000 citations.

Bees-inspired algorithms

Over the last few decades, algorithms focused on the intelligent behaviour of social beings such as ants, birds, fish, and bacteria have been widely researched and applied to computer-aided optimisation (Rajasekhar *et al.*, 2017). This section will discuss bee behaviour and the algorithms that have been influenced by it. This section is split into two subcategories: (1) marriage selection, (2) foraging, and (3) nest selection behaviours.

Bees are flying insects with four wings that feed on flowers and are well-known for pollination and honey production. Of all bee species, honey bees have an extraordinary life cycle, which has sparked the interest of numerous researchers. Computer scientists discovered that powerful metaheuristic methods could be developed using the honey bees' intelligent teamwork, job assignment, marriage, and food discovery. In nature, honey bees live in colonies and work in a highly organised social order. The honey bee hive serves as the breeding ground for new bees. The hive is made up of large vertical combs that provide a home for the bees.

Bees are classified into three types: queen, worker, and drone bees. The Queen bee is the hive's only mated adult female. Her reason for being is to reproduce bees by egg-laying. Worker bees are the colony's most numerous non-mating female bees population. This species of bee is critical for nectar collection and guarding the hive against intruders. Drones are unfertilised male honey bees. This results in the drone bees becoming fatherless. This singular behaviour motivated Abbass (2001); Haddad *et al.* (2006); Jung (2003) to suggest algorithms inspired by bees mating.

Abbass (2001) discussed marriage concerning the honey bee optimisation algorithm (MBO). The algorithm simulates many phases of honey bee evolution. It begins with a queen bee without a colony and progresses to establishing a eusocial colony (a colony with one or two queens in the chamber). Algorithm 3 contains the MBO algorithm's pseudocode. A series of mating flights are created by randomly initialising each queen's energy, speed, and position. Then, each queen travels between states at her own speed and mates with a drone. Probabilistically, a drone mates with a queen. If mating is successful, drones' sperm is applied to a repository of partial solutions known as the spermatheca. When the queen bee returns to the nest, she

Algorithm 3: Marriage Bee Optimisation

input : Q,W, and B to be the number of queens, workers, and broods respectively; M to be the spermatheca size; $energy$, and $speed$ to be the queen's energy and speed respectively

output: *queen*

Initialise the workers;
Randomly generate queens;
Conduct a local search to get a good queen;
for *a pre-defined Max number of mating flights* **do**
    for *each queen in the queen list* **do**
        Initialise energy, speed and position;
        Queen moves between states randomly;
        Drone is selected probabilistically using $prob(Q,D) = e^{-difference/speed}$;
        `//` $prob(Q,D)$ `represent the successful mating probability`
        if *(queen selects a drone)* **then**
            Sperm is deposited in spermatheca of queen;
        else
            Update queen's speed $(speed(it+1) = \alpha.speed(it))$ and energy $(energy(it+1) = energy(it) - step)$;
        end
    end
    Generate new broods by crossover and mutation operations;
    Employ workers to improve the broods;
    Update workers fitness;
    while *(best brood 4 worst queen)* **do**
        Replace the queen with best brood;
        Eliminate the best brood from brood family;
    end
end

Algorithm 4: Queen Bee Evolution

input : t = time; NP = population; ε = normal mutation rate; P_m = normal mutation probability; P'_m = strong mutation probability; I_q, I_m = Queen and selected bee.
output: $P(t)$

```
t=0 ;
Initialise and evaluate P(t);
while termination criterion is not satisfied do
    t = t + 1;
    Select P(t) from P(t − 1); P(t) = ((I_q(t − 1), I_m(t − 1)));
    Recombine P(t);
    Crossover;
    Mutation;
    for i = 1 to NP do
        if i ≤ εNP then
            Do Mutation with P_m;
        else
            Do Mutation with P'_m;
        end
    end
    Evaluate P(t)
end
```

begins reproduction by randomly choosing a sperm from the spermatheca. After that, crossover and mutation operators are used to generate distinct broods. Additionally, workers are used to strengthening the clans. If either of these broods proves to be superior to the queen, she is replaced. After that, the remaining clans are eliminated, and a new mating flight begins. Haddad *et al.* (2006) used the same technique as Algorithm 3 to solve a problem involving water supply optimisation. Since the process is identical, the algorithm was named the Honey-Bees Mating Optimisation (HBMO).

Jung (2003)suggested a queen bee evolution algorithm for GA to improve their optimisation capability. The queen bee's function influenced the algorithm in reproduction. The queen bee crosses with other bees chosen as parents by a selection algorithm as the fittest member in a generation in the algorithm. This technique would increase the likelihood of premature convergence. An intensive mutation process is proposed to address this problem. The suggested hybrid algorithm converged in most instances, as shown by experimental findings from one combinatorial and two continuous implementations. The pseudocodes for Queen Bee Evolution

(QBA) are seen in Algorithm 4.

Algorithm 5: Bee Colony Optimisation

input : B = Total number of bees in the hive; NC = The number of constructive moves during each forward pass
output: *Bee*

```
Initialisation;
Initialise and evaluate P(t);
for I Iterations do
    for all B Bees do
        Set i=1;        // counter for number of constructive moves
        Evaluate all possible constructive moves;
        According to the fitness obtained choose one move by using the roulette wheel
         selection;
        i = i + 1;
        if (i ≤ NC) then
            Go to step 6;
        end
    end
    All bees return back to the hive (backwardpass);
    Sort the bees using their fitness values obtained;
    for All bees B do
        Backward pass;
        Every bee decides randomly whether to become a recruiter or to become a
         follower via dances and fitness sharing;
        For every follower choose a new solution by roulette wheel basis;
        if Best solution obtained in iteration is global best then
            Update best-known solution;
        end
    end
end
Display the best result;
```

The other remarkable behaviour is the worker bees' mutual intelligence when it comes to food foraging. In the morning, a single forager, according to Frisch (1993) and Seeley (1986), can visit various flowers. If a bee is attracted to and rewarded by a particular flower form, the bee will return to it as often as possible during the day. The foraging process begins when worker bees depart the hive in search of food. A bee chooses which flower to visit based on her cognitive intelligence and perception of other bees nearby. Finally, the nectar is transported to the hive by foragers. After visiting a fruitful food supply, a bee returns to the hive and performs a special dance.

Algorithm 6: Artificial Bee Colony

```
1  Start;
2  Initialisation (randomly);
3  while termination criterion not satisfied do
4  |   Send the employed bees onto their food sources and evaluate their nectar amounts;
5  |   Place the onlookers depending upon the nectar amounts obtained from employed
   |     bees;
6  |   Send the scouts for exploiting new food sources;
7  |   Memorise the best food sources obtained so far;
8  end
9  Display the best food source obtained so far;
10 End;
```

Honey bees interact through the waggle dance, which relays knowledge about resources found more than 100 metres from the hive. Other workers watch the dancing bee and use its behaviour to assess the food source's direction and size. This foraging colony behaviour is the inspiration for the Bee Colony Optimisation (BCO) (Teodorovic and Dell'Orco, 2005), Artificial Bee Colony (ABC) (Karaboga, 2005), and the Bees Algorithm (BA)(Pham *et al.*, 2005).

Initially designed for combinatorial domains, the BCO was applied to TSP. The hive is placed in a random solution space and selected probabilistically in this algorithm. Partially solved problems are built in stages using a derived probabilistic equation. The recruitment of bees to these partial solutions is then extended. While the BCO algorithm, like the ACO, constructs solutions constructively, the primary distinction being that the BCO algorithm creates solutions partly. Bees solve a portion of the problem in each stage by flying a pair of nodes forward during a forward pass. Both bees are returned to the hive during the backward pass period. They share information about the consistency of the generated partial solutions and determine whether to leave the partial solution and revert to being an uncommitted follower. Continue expanding the same partial solution without hiring nestmates or dance to recruit nestmates before returning to the previously generated partial solution. Before relocating the hive after initial construction, the solution produced in the current iteration is improved using the 2-opt and 3-opt heuristic algorithms. The pseudocodes for BCO are seen in Algorithm 5below.

While ABC was proposed for numerical test function optimisation, three kinds of bees are

used in the algorithm: working bees, onlooker bees, and scout bees (Karaboga, 2005). The population is evenly divided into two groups: working bees and onlookers. Additionally, this algorithm employs a random scout bee to explore the search room. Every iteration of the algorithm consists of three major steps: employed bees are placed on food sources, onlooker bees are placed on food sources based on their nectar content, and scout bees are sent to the search area for exploration. Proportional sorting is used to attract onlooker bees to the promising patches. The neighbourhood search process generates new solutions through the extrapolation crossover procedure. During this process, an employed bee selects another employed bee at random and creates a new solution. If this approach is superior to the current one, a newly working bee is chosen as the patch's representative bee. ABC employs site abandonment, which entails essentially leaving a patch after a defined number of iterations if no further change is observed. Algorithm 6 illustrates the ABC pseudocodes. Later, the biologist develops their study to investigate the hidden bees foraging behaviour. They used artificial flowers (trapline) and radar technology to observe the solitary bee foraging behaviour.

Similarly to foraging, bees communicate with one another throughout their search for a new nest site through waggle dances. However, dance exhaustion occurs during nest-site picking, as scout bees weaken their dance ability through repeated visits to the prospective site. As reported previously, each time a scout bee returned to the swarm cluster, 15 waggle runs were decreased, raising the rate of consensus-building by preventing the broadcasting of less in quality locations. Additionally, there is a need to strike an equilibrium between speed and precision during the site selection period when bees searching for a new nest are hanging from an exposed tree limb.

Diwold *et al.* (2010) conducted a biological simulation of this behaviour in complex and noisy environments, with positive findings. At initialisation, the population of bees is put in a random location in space. Then, a fraction of bees comprising the scouts flies arbitrarily to a predetermined distance from the home point before conducting the local search to strengthen the solution. With recruiters moving to a random spot within a predefined overall length from the scout's venue, sites with the best fitness would see a higher recruiting rate. Additionally, the recruiters will conduct a local search in that location in order to strengthen the solution. If no

change is made, the site will be abandoned, and recruiters will transition to scouts. When the best solution is achieved and a sufficient number of scouts exceeds the threshold value at that spot, the whole swarm can lift off from their home site to the latest best solution. Otherwise, the swarm is arbitrarily assigned to a new home point or its current home. This procedure is replicated before a feasible solution is found.

Other popular natured-inspired metaheuristic algorithms

According to Lones (2020), there are ten additional algorithms with over 1,000 citations. They are referred to the Harmony Search (Geem *et al.*, 2001) with more than 4000 citations, Cuckoo Search (Yang and Deb, 2010) (>3000 citations), Bacterial Foraging Optimisation (Passino, 2002) (>2500 citations), Gravitational Search Algorithm (Rashedi *et al.*, 2009) (>2500 citations), Firefly Algorithm (Yang, 2009) (>2000 citations), Biogeography-Based optimiser (Simon, 2008) (>2000 citations), Imperialist Competitive Algorithm (Atashpaz-Gargari and Lucas, 2007) (>1500 citations), Grey Wolf optimiser (Mirjalili *et al.*, 2014) (>1000 citations), Shuffled Frog Leaping Algorithm (Eusuff and Lansey, 2003) (>1000 citations), and Teacher-Learning Based optimisation (Rao *et al.*, 2011) (>1000 citations).

2.2.3 Bees Algorithm

The Bees Algorithm is a well-known population-based metaheuristic algorithm for solving continuous and combinatorial optimisation problems Lones (2020). It is inspired by the foraging behaviour of honey bees (Pham *et al.*, 2005). This intelligent algorithm was motivated by honey bees' natural foraging behaviour, which involves finding nectar and sharing knowledge about food sources with the rest of the hive's bees. The Bees Algorithm was initially designed for searching in continuous space and was evaluated using numerical functions. On the other hand, other well-known algorithms such as GA, ACO, TS, and SA were originally developed to solve a combinatorial problem in discrete space. Their outputs were evaluated using TSP as the standard test function.

The basic version of Bees Algorithm

The basic version of this algorithm requires six initial parameters besides the stopping criterion to be set. These parameters include the number of scout bees (n), the number of patches selected from the n visited points (m), the number of elite patches selected from the m selected patches (e), the number of bees recruited for the best e patches (nep), the number of bees recruited for the remaining selected patches (nsp), and the size of the patches (ngh). The error threshold and the maximum number of function evaluations are often used as the stopping criteria. The simplest approach to measure the NFE is multiplying the number of iterations by the colony or population size of the bees. Equation 2.4 represents the size of a bee colony.

$$colony\,size = (e.nep) + (m.nsp) + (n - m) \tag{2.4}$$

$$patch\,position = random\,uniform[x_{min}, x_{max}] \tag{2.5}$$

The Bees Algorithm starts with the uniformly random placement of the n scout bees in the search space (see Equation 2.5). The second stage evaluates the fitness of the scout bees' visited points. The scout bees reflect on the quality of the sites visited by the waggle dance. In step 4, the bees with the highest fitness values are chosen as "elite bees," and the other potential as "selected bees". The algorithm then performs searches in the neighbourhood of the chosen sites and assigns additional bees to look for the best e site in steps 5 and 6. The worker bees will be randomly assigned to a neighbourhood (patch position ± ngh) (see Equation 2.6 or 2.7). The algorithm will choose only the bee with the highest fitness to shape the patch's next population. Searches in the best e bees region demonstrating the most viable solutions are made more detailed by attracting more bees to pursue them than the other selected sites.

$$forager\,position\,on\,a\,patch = (patch\,position - ngh) + 2.ngh.random\,uniform[x_{min}, x_{max}] \tag{2.6}$$

$$forager\ position\ on\ a\ patch = patch\ position \pm ngh.random\ uniform[x_{min}, x_{max}] \quad (2.7)$$

In step 7, the remaining bees in the colony are randomly distributed across the search space, scouting for new potential solutions using a similar principle when generated the initial solutions (see Equation 2.5). This global search procedure allows scout bees to explore random solution spaces to maximise their probability of avoiding local optima. The fitness of the locations visited by worker bees is determined for each flower patch. If one of the bees lands higher than the previous patch, the bee's spot becomes the current patch. The process continues until a stopping condition is reached. To summarise, Algorithm 7 depicts the BA's pseudocode in its simplest form.

Algorithm 7: Bees Algorithm

input : n = number of scout bees; e = elite sites; m = selected sites; nep = worker bees on the e sites; nsp = worker bees on the $m - e$ sites; ngh = the neighbour search range.

output: *Bees*

1 **Start**;
2 Initialise n scout bees population, randomly searching for sites or patches;
3 Evaluate the fittness of the sites;
4 **while** *termination criterion not satisfied* **do**
5 Select m sites for neighbourhood search;
6 Recruit worker bees (nep and nsp) for selected sites, more bees on e sites;
7 Exploit all selected sites inside ngh range and evaluate fitnesses;
8 Select the fittest bee from each patch;
9 Assign remaining bees to explore randomly and evaluate their fitnesses;
10 **end**
11 Report the best *Bees*;
12 **End**;

The standard version of the Bees Algorithm (Pham and Castellani, 2009) extends the basic version algorithm; the initial parameters become eight parameters, with site abandonment and shrinking strategies to become more efficient and effective. Two additional parameters are $stlim$ (the limit number of stagnation) and $shrinking$ rate. This strategy improves the Bees Algorithm ability to escape from local optima.

The Combinatorial Bees Algorithm (CBA)

The search principles for combinatorial and continuous problems are entirely contradictory. The combinatorial approach is distinguished from the continuous approach because there is no real distance value (Koc, 2010). The combinatorial domain exhibits many distinctive features. According to Otri (2011), the search space is discrete, the restrictions are finite, the solution has an ordered set, and the combination has a cost function.

In essence, the BA procedure for COP is almost equivalent to the continuous one. The primary distinction between continuous and combinatorial versions is in their search operators (global and local). Naturally, the combinatorial version would substitute a discrete random generator for the continuous version's real number generator. Besides the operator, another thing that needs to remember is the parameter of ngh. The parameter could be represented by the number of local search movement (maximum movement is equal to $Dims$-1).

The first combinatorial problem solved by the Bees Algorithm was the scheduling problem (Pham *et al.*, 2007a). The Bees Algorithm also unravelled the PCB assembly sequence in the same year as an early TSP-based COP (Pham *et al.*, 2007c). In the same year, the Bees Algorithm also had solved the PCB assembly sequence as an early combinatorial problem (Pham *et al.*, 2007j). After that publication in 2007, other important works began to emerge in the combinatorial problems, such as timetabling (Abdullah and Alzaqebah, 2013; Khang *et al.*, 2011; Phuc *et al.*, 2011b; Lara *et al.*, 2008), jobs scheduling (Xu *et al.*, 2016a; Tapkan *et al.*, 2013a; Phrueksanant, 2013; Tapkan *et al.*, 2012a; Ziarati *et al.*, 2011; Sadeghi *et al.*, 2011; Özbakir *et al.*, 2010; Ozbakir and Tapkan, 2010; Xu *et al.*, 2010a; Pham *et al.*, 2007c), PCB assembly planning (Castellani *et al.*, 2019; Ang *et al.*, 2013b, 2010), circuit designing (Mollabakhshi and Eshghi, 2013), and gene sequence (Ruz and Goles, 2013; Choon *et al.*, 2014b,c; Ruz *et al.*, 2014; Koo *et al.*, 2014), supply chain (Yuce *et al.*, 2017, 2015; Packianather *et al.*, 2014), vehicle routing problem (VRP) (Alzaqebah *et al.*, 2018; Ali and Al Masud, 2018; Fenton, 2011), disassembly sequence planning (Laili *et al.*, 2019; Liu *et al.*, 2018a), and path planning (Darwish *et al.*, 2018; Sabri *et al.*, 2018).

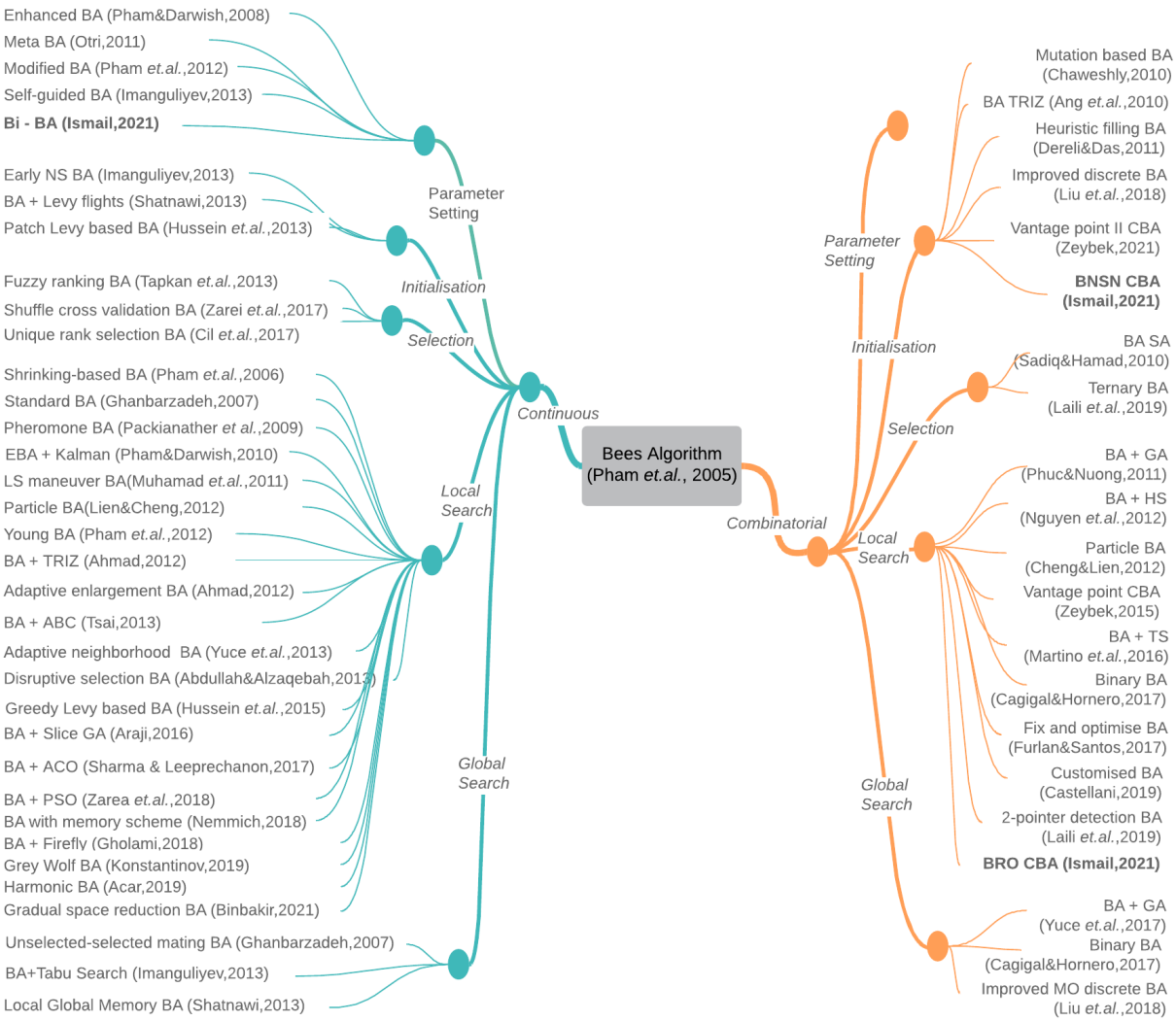


Figure 2.5: The developments of the Bees Algorithm

Figure 2.5 shows the development of BA in continuous and combinatorial domain with majority focusing in the local search element of the algorithm. In most of the combinatorial version works cited above, the neutral local search operators (swap, insertion, reversion) are used as the exploitation mechanisms. Castellani *et al.* (2019) used a mixture of local search operators for the first time and followed by Ismail *et al.* (2020) , who used a combination of swap-insertion-reversion with a probability of 33% per cent, each time using iterative best improvement search (see Algorithm 8). The algorithm's developments fall into five broad categories: parameter tuning and setting, selection, initialisation, local search (exploitation), and global search (exploration) (see Figure 2.6). The majority of developments are concentrated on the BA's local search procedure (Hussein *et al.*, 2017b). Researchers undertook at least nine-

teen development projects to enhance the CBA. Some of them modified the initialisation and population sorting of CBA by putting a constructive heuristic method to generate good initial solutions or a ranking selection method to select the sites that diversely cover the solution space (Chaweshly, 2010; Ang *et al.*, 2010; Dereli and Das, 2011; Tian *et al.*, 2013; Liu *et al.*, 2018a; Sadiq and Hamad, 2010; Laili *et al.*, 2019). On the other parts, some researchers develop the global search operator by attaching a constructive heuristic (Liu *et al.*, 2018a) and reinforcing the exploration (Packianather *et al.*, 2014). The rest were focusing the exploitation or local search part by either hybridising it with other improved heuristic algorithm or intensifying the solution with other metaheuristic or local search algorithm (Phuc *et al.*, 2011b; Nguyen *et al.*, 2012; Furlan and Santos, 2017; Zeybek and Koç, 2015; Castellani *et al.*, 2019). The list above shows that no development used nature (bees' behaviours) as the main source of concepts. All the works mentioned above focused on either the hybridisation with other existing metaheuristic and heuristic algorithms or the modification on the local search operator rather than explore possible approaches from bees' behaviour.

Algorithm 8: CBA's Neighbour Search Mechanism Procedure

input : T_{init} = initial tour from previous process.
output: T = a tour of k cities.

Start;
$LS \leftarrow$ random.integer(1,3);
$a^{th} \leftarrow$ random.integer(1,k);
$b^{th} \leftarrow$ random.integer(1,k);
if LS = *1* **then**
    $T \leftarrow$ Swap(T_{init}, a^{th}, b^{th});
else
    if $LS = 2$ **then**
        $T \leftarrow$ Insert(T_{init}, a^{th}, b^{th});
    else
        $T \leftarrow$ Reverse (T_{init}, a^{th}, b^{th});
End;

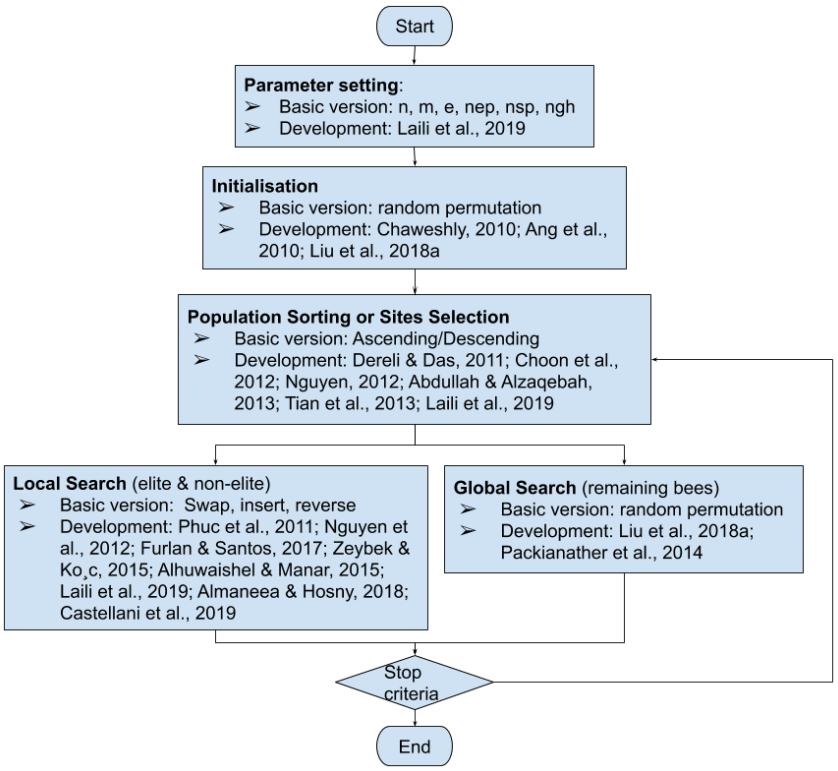


Figure 2.6: Developments of the Combinatorial Bees Algorithm

The test functions of the continuous optimisation problem

In intelligent optimisation computing, test functions referred to as artificial environments are used to determine the characteristics of optimisation algorithms, such as accuracy or precision, convergence speed, and robustness. It is normal to compare various algorithms using these test functions on a large test set. However, the efficiency of an algorithm in comparison to another algorithm cannot be quantified in terms of the number of problems solved more efficiently. The "no free lunch" theorem states that when two searching algorithms are compared with all available functions, the efficiency of any two algorithms would be approximately equal.

The artificial landscapes for single-objective optimisation problems discussed here are adapted from Jamil and Yang (2013). A few are used here since they used the test functions from previous research (Pham and Castellani, 2009). Table 2.1 contains some objective functions for single-objective optimisation scenarios. This section contains only the general form of the equation, a plot of the objective function, the limits of the object variables, and the global minima coordinates.

Table 2.1: The Benchmark Functions

ID	Function Name	Function
F1	Goldstein & Price (2D)	$f(x) = (1 + (x_1 + x_2 + 1)^2 + (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)).(30 + (2x_1 - 3x_2)^2 + 222(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$
F2	Branin (2D)	$f(x) = a(x_2 - b(x_1)^2 + cx_1 - r)^2 + s(1-t)cos(x_1) + s; a = 1, b = 5.1/(4\pi^2), c = 5/\pi, r = 6, s = 10, t = 1/8\pi$
F3	Martin & Gaddy (2D)	$f(x) = (x_1 - x_2)^2 + ((x_1 + x_2 - 10)/3)^2$
F4	Easom (2D)	$f(x_1, x_2) = -cos(x_1)cos(x_2)e^{(x_1-\pi)^2-(x_2-\pi)^2}$
F5	Schaffer (2D)	$f(x_1, x_2) = 0.5 + \frac{sin^2(\sqrt{x_1^2+x_2^2})-0.5}{[1+0.001\cdot(x_1^2+x_2^2)]^2}$
F6	Rosenbrock	$f(x) = \sum_{i=1}^{n-1}(100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2)$
F7	Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$
F8	Griewank	$f(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$
F9	Schwefel	$f(x_1 \cdots x_n) = \sum_{i=1}^{n}(-x_i sin(\sqrt{\lvert x_i \rvert})) + 418.982887 \cdot n$
F10	Rastrigin	$f(x_1 \cdots x_n) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$

Table 2.2: The Benchmark Functions' Interval and optimum solutions

ID	Function Name	Interval	Optimum
F1	Goldstein & Price (2D)	$-2 \leq x_i \leq 2$	$f(0, -1) = 3$
F2	Branin (2D)	$-5 \leq x_i \leq 10$	$f((-\pi, 12.275)$ or $(\pi, 2.275)$ or $(9.424, 2.475)) = 0.397$
F3	Martin & Gaddy (2D)	$0 \leq x_i \leq 10$	$f(5, 5) = 0$
F4	Easom (2D)	$-100 \leq x_i \leq 100$	$f(\pi, \pi) = -1$

F5	Schaffer (2D)	$-100 \leq x_i \leq 100$	$f(0,0) = 0$
F6	Rosenbrock	$-1.2 \leq x_i \leq 1.2$	$f(1,1) = 0$
F7	Sphere	$-5.12 \leq x_i \leq 5.12$	$f(0,\cdots,0) = 0$
F8	Griewank	$-512 \leq x_i \leq 512$	$f(0,0) = 0$
F9	Schwefel	$-512 \leq x_i \leq 512$	$f(420.968746, 420.968746) = 0$
F10	Rastrigin	$-5.12 \leq x_i \leq 5.12$	$f(1,1,1,1) = 0$

The test set contains many well-characterised functions that will allow us to obtain and generalise, to the greatest extent possible, the results for each type of function. The dimension of the search space is another critical element in determining the problem's complexity. The test functions used to validate algorithms are used to determine how well the proposed algorithm performs.

Travelling salesman problem as the test function of the combinatorial optimisation problem

TSP is a NP-complete problem that cannot be solved efficiently using an exact algorithm. Numerous pieces of literature have shown that exact algorithms, such as the brute-search force (Fellows *et al.*, 2012), the integer programming methods (Climer and Zhang, 2006), and dynamic programming methods (Bellman, 1962; Held and Karp, 1962), are capable of resolving the TSP with fewer cities. They will use an efficient Turing machine to find the optimum solution in a reasonable amount of time. As the TSP scale was expanded, the metaheuristics algorithm performed admirably (Johnson and McGeoch, 1997).

This problem consists of k cities, with a path (edge) connecting each pair of cities. The objective is to find the Hamilton cycle with the fewest possible costs. The simplest solution to this mathematical problem is to compare all feasible routes, which often requires significant computational resources due to the fact that the number of possible routes increases factorially as the number of places to visit increases.

TSP is often used in practical applications such as the delivering of products or services,

determining the shortest passenger lane, and designing bus routes. Still, it is also used in fields unrelated to transit paths, such as scheduling. Although TSP is a straightforward, simple problem, its complexity makes it an excellent basis for evaluating the performance of all types of algorithms for combinatorial optimisation problems. The most frequently used type of TSP for benchmarks is symmetrical TSP, or most practitioners simply refer to TSP, and the most popular data set issue is TSPLIB (Reinelt, 1991, 1994). Many metaheuristics algorithms can quickly achieve the best-known solution for simple TSP instances with cities range 50 to 100. However, in larger situations, over 200 cities, the accuracy efficiency of certain algorithms degrades dramatically, necessitating hours of CPU time on supercomputers (Karaboga and Gorkemli, 2011). Koc (2010), Otri (2011), Zeybek and Koç (2015), and Ismail *et al.* (2020) were the researchers who developed the CBA for resolving TSPs from the TSPLIB dataset for up to 100 towns.

TSP is illustrated in the graph $G = (V_x, E)$, where V_x is a collection of vertices representing cities and E is a collection of connecting lines between cities. Each edge represents a potential path connecting two linked vertices or cities. The variable $d_{i,j}$ is connected with the edge (i, j) and represents the Euclidean distance between the vertex (x_i,y_i) and the vertex (x_j,y_j) using equation (2.8). Until the CBA is executed, these distances between all edges are measured and stored in a distance matrix.

$$d_{i,j} = \sqrt{(x_i - x_j)^2 - (y_i - y_j)^2} \tag{2.8}$$

The TSP objective is to determine the shortest possible total tour duration for the final closed Hamilton cycle (visiting each city only once) as described in equation (2.9).

$$tour\ \ length = \sum_{i=1}^{n} d_{i,i+1} + d_{n,1} \tag{2.9}$$

TSP may be symmetrical or asymmetrical. The distances between two cities in symmetrical TSPs are not dependent on the trajectory path. For instance, if the distance between two cities i and j is denoted by $d_{i,j}$ and if $d_{i,j} = d_{j,i}$, then the TSP is symmetrical and vice versa.

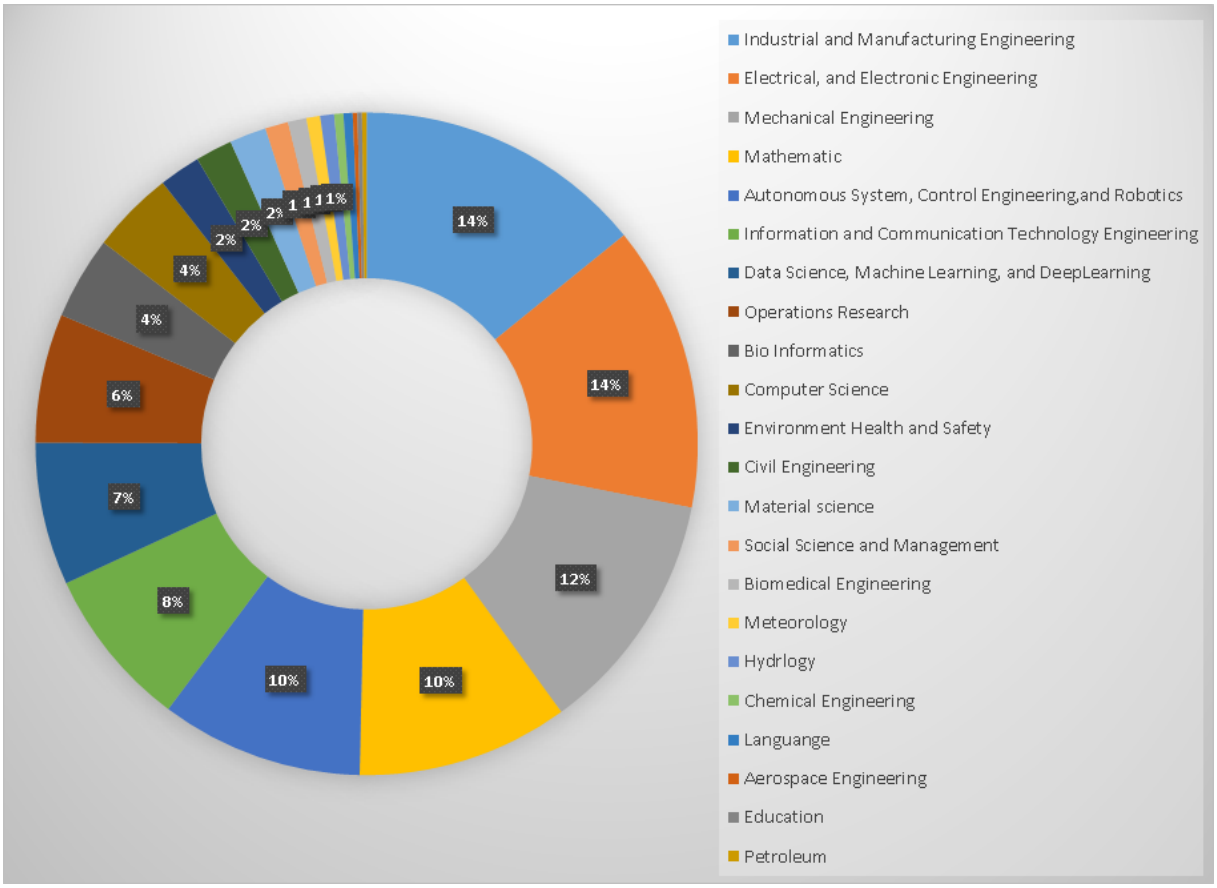
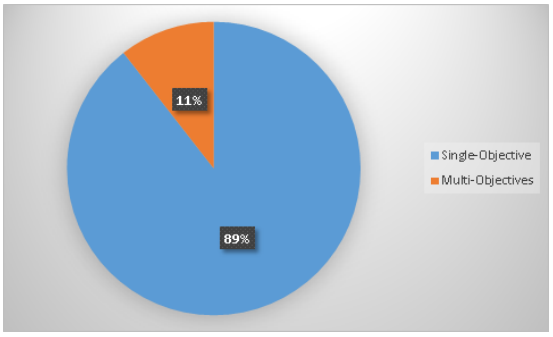


Figure 2.7: Percentages of applications using the Bees Algorithm by specialised area

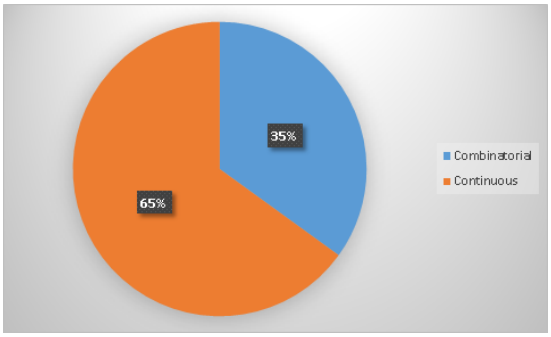
2.2.4 Applications of the Bees Algorithm

The Bees Algorithm has been successfully applied to an enormous variety of continuous and combinatorial optimisation problems. By May 31, 2021, there had been 513 published applications. The applications are found in 22 different specialised fields, ranging from engineering to social science. Technology domains such as industrial, mechanical, and electrical engineering contribute to more than 40 per cent of all applications (see Figure 2.7). Additionally, continuous domain and single objective problems dominated the application, contributing to 65 and 89 per cent of the total (see Figure 2.8b and 2.8a). While the majority of the applications are continuous problems, 90 per cent of industrial engineering applications are combinatorial problems such as scheduling, sequencing, routing, and so on. There are over 50 variants of BA, 40 per cent of which are combinatorial variants that have been dominated by hybridisation with other methods. Figure 2.9 demonstrates that the Firefly Algorithm is the most often hybridised of

the thirteen metaheuristic algorithms used for development. Furthermore, the algorithm is surprisingly popular in both Turkey and Iran, as shown in Figure 2.10. Tabel A.1 on the Appendix section displays a complete list of all The Bees Algorithm applications along with their variants.



(a) Single Objective Vs Multi Objectives



(b) Continuous Vs Combinatorial

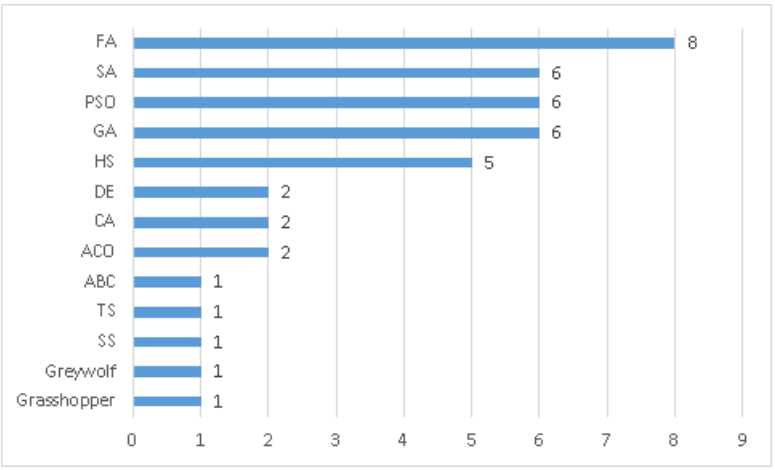


Figure 2.9: The Bees Algorithm with other metaheuristics (Hybrid)

2.3 Behaviour of bees in nature

2.3.1 Bees in nature: the behaviours of food foraging and nest site selection

A honey bee colony may spread over great distances (more than 10 kilometres) and in several directions simultaneously to access a diverse array of food sources (von Frisch and Lindauer, 1956; Seeley, 1986). A colony succeeds as the foragers are assigned to productive areas. In theory, flower patches with abundant nectar or pollen that can be gathered with minimal effort

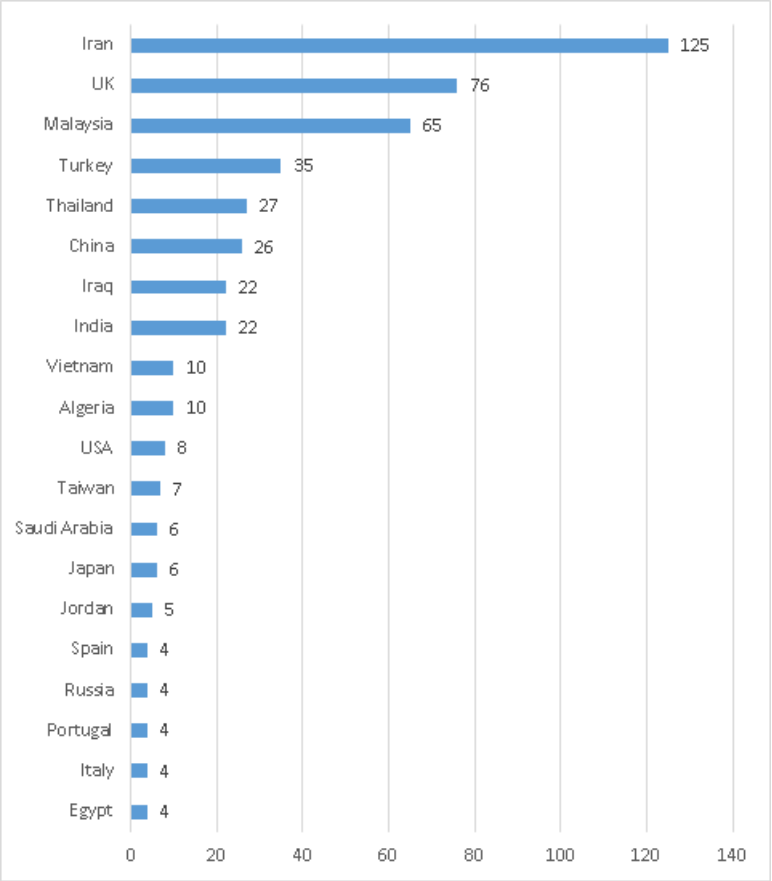


Figure 2.10: Top 20 of the Bees Algorithm users by Country

should attract more bees. In comparison, patches deficient in pollen or nectar should attract fewer worker bees (Fries and Camazine, 2001).

The foraging process starts inside a colony with the deployment of scout bees to look for promising flower patches. Scout bees flit randomly between patches. During the harvesting season, a colony begins to explore, with a portion of the population serving as scout bees (Seeley *et al.*, 1996). When the scout bees return to the hive, they deposit the nectar and proceed to perform the "waggle dance" to remind the potential patches of their existence (Von Frisch, 1974).

This mystical dance is critical for colonial contact. It provides three pieces of information about a flower patch: the position in which it will be located, the distance between the flower patch and the hive, and the quality rating of the flower patch. This knowledge enables the colony to direct its bees specifically to flower patches without using guides or maps. The waggle dance is the sole source of information about the external world for each individual. This dance allows

the colony to assess the relative merits of various patches based on the quality of food they offer and the resources required to harvest it. After waggle dancing on the dance floor, the dancer (the scout bees) returns to the flower patch with the assistance of follower bees waiting inside the hive. Extra follower bees are distributed to more potential patches. Bees track their food supply when gathering from a patch. It is required to determine the next waggle dance to perform upon returning to the hive (Fries and Camazine, 2001). If the patch is a viable food supply, it will be marketed in the dance, attracting more bees to the patch.

The behaviour of nectar source selection is a complex but critical activity for honey bee colonies. When a honey bee colony gets overcrowded, it must be dispersed to control the source effectively (Seeley, 1986). This vital decision-making method operates autonomously. Nectar source selection behaviour is primarily concerned with how a colony chooses between several nectar sources by evaluating many variables simultaneously and comparing them to alternative solutions. The decision is taken as all scout bees are dancing for the same place, and it takes a few days for half of the colony to move to a new hive.

Yonezawa and Kikuchi (1996) proposed a honey selection model focused on bee collective intelligence of foraging. The paradigm studied the concept of mutual and reciprocal knowledge in a dynamic world. In the model, one and three worker bees were simulated. The simulation results showed that the model with three bees generated more balanced results than the model with a single bee.

Cox and Myerscough (2003) pioneered a model of foraging in honey bee colonies. This model fills up the gaps in Camazine and Sneyd (1991). The consequences of environmental and colony conditions are studied in this model. The differential equations collection also incorporates the results of the source (distance from the hive, and the rate of nectar flow) and the implications of worker bees behaviour.

Schmickl *et al.* (2003) published a detailed model of honey bee nectar supply collection. Although the model is based on individual processes, it generates intriguing findings at the global colony level. Another promising aspect of the model is that it can project the imaginary honey bee colony's regular net honey benefit. Consequently, this enables the exploration of the

most significant number of possible outcomes of foraging decisions. The model presented here is used to investigate the complexities and productivity of a honey bee colony's decentralised decision-making mechanism in an evolving environment. However, in a significant departure from previous ones, the model incorporates goal selection and workload balancing methods and the energy balance of each foraging bee. Additionally, foragers are viewed as intelligent entities that waste resources and exhibit distinct behaviours.

Nest-site selection is another critical activity that involves optimisation like nectar source selection behaviour in honey bee colonies. In honey bee colonies, nest site selection can be summarised as a group decision-making method. Scout bees conduct this process by locating multiple possible nest sites, evaluating them, and selecting the best one using competitive signalling (Seeley *et al.*, 2006).

2.3.2 Bees traplining foraging behaviour in nature

The recursive movement pattern, in which an animal returns to previously visited locations, is typical in the animal world, including bees. Around 40 years ago, D. H. Janzen invented the word traplining to describe euglossine bees visiting flowers in a repetitive manner (Heinrich, 1979; Thomson and Goodell, 2001). The name is derived from an analogy of human trappers who regularly inspect their traps (Thomson *et al.*, 1997). Since early observations of traplining relied solely on detailed accounts to describe it, various standards have been used to conclude that animals engage in trapping behaviour. In the last two decades, more sophisticated computational statistical methods have been used to classify traplining behaviour (Thomson *et al.*, 1997; Ohashi *et al.*, 2007; Lihoreau *et al.*, 2012b; Woodgate *et al.*, 2017). Since the 2000s, they have tracked foraging behaviour using harmonic or heat radar.

Trapline foraging is characterised as a series of predictable visits to a series of resource points or patches (Thomson *et al.*, 1997). Traplining has been seen in a wide range of animals and has mostly been synonymous with foraging natural resources (e.g., fruits, nectar, insects, and foliage). Traplining foraging has been shown to enhance foraging success by strengthening the forager's expertise, thus increasing the forager's competitive advantage over random

foragers and the forager's ability to perceive and respond to environmental changes such as fluctuating competition strength (Ohashi *et al.*, 2008). These traplines are often the shortest circuits for pollinating insects, such as bumblebees, to visit all known flower locations precisely once before returning to the nest, and hence are solutions to the well-known travelling salesman problem (Lihoreau *et al.*, 2012b).

In the wild, honey bees forage on dozens of flowers during a single foraging journey. Numerous early studies of traplining behaviour concluded that the closest unvisited flower would be the next stop before all flowers were reached, which was speculated to account for the routing behaviour of certain species, including bees (Lihoreau *et al.*, 2012b). Numerous experiments, however, have shown that trapping animals may not often take the shortest path available, as this route is not often the most valuable. Lihoreau *et al.* (2012b); Ohashi *et al.* (2007); Ohashi and Thomson (2012, 2013); Klein *et al.* (2017); Buatois and Lihoreau (2016) discovered several factors (distance) that influence the choice of the next unvisited flower. They are the angle of rotation and spatial memory. Bees favour short distances over straight steps, according to Ohashi *et al.* (2007); Lihoreau *et al.* (2012a).

While Woodgate *et al.* (2017) discovered the opposite, bees did not choose visit sequences that resulted in the shortest overall course but instead prioritised movements to nearby flower. He chastised his competitors for using an array with a relative disparity between the optimal and nearest neighbour routes.

According to Woodgate, experienced bees can reduce exploration outside the feeder collection, with their flights being straighter as they gain experience, rather than improving the sequence of feeder visits. The seasoned bees will cease returning to an empty flower and will increase their exploration pace. They will return after the flower has replenished the honey supply. Additionally, they discovered that bees would cease re-visiting a flower for some purpose. Apart from honey depletion, intruders and rivals can cause bees to avoid a flower that needs to be visited.

2.4 Population-based metaheuristic development

Population-based metaheuristics are naturally more explorative. In contrast, single-based metaheuristics are more exploitative since the vast diversity of initial populations. In the design of a Population-based metaheuristic, the operation of the initial population is often ignored. Nonetheless, the efficacy and efficiency of the algorithm are dependent on this phase. As a result, this move needs further focus. However, not only diversification must be considered when developing metaheuristics. Intensification is equally critical to the algorithm's robustness. These two contrasting conditions, the exploration of the search space and exploitation of the best solutions discovered, must be considered when developing a metaheuristic (Talbi, 2009).

Diversification obtained "good" solutions defining promising areas. Intensification entails a deeper examination of promising areas with the expectation of identifying more effective solutions. Diversification requires visiting previously unexplored regions to ensure that all regions of the search space are searched evenly and that the search is not limited to a small number of regions. In this design space (see Figure 2.11), the most intense search algorithms are random search in terms of exploration. For exploitation, it is the iterative improvement local search. Each iteration of a random search produces a random answer in the search space. At each iteration of the simple steepest local search algorithm, the best neighbouring solution that improves the current solution is chosen.

Using more extensive diversification can increase the likelihood of discovering (high-quality) solutions to a given problem in less local search steps; however, the time complexity of deciding improving search steps is much higher (Hoos and Stützle, 2004). Blum and Roli (2003) believe that finding the balance between diversification and intensification is the best way to achieve a robust metaheuristic.

The metaheuristic development may simultaneously combine these two factors on its different elements of the algorithm. The metaheuristic's initialisation could be rearranged to accommodate the high diversity level without jeopardising the neighbour search mechanism's intensification level.

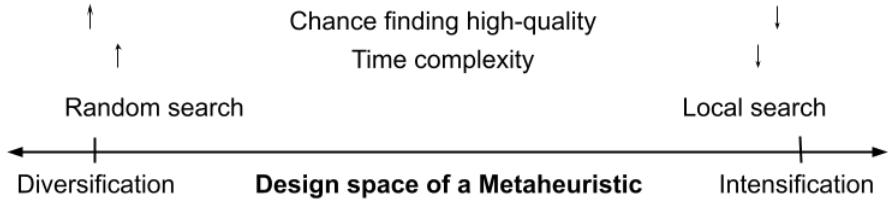


Figure 2.11: Exploration (diversification) versus exploitation (intensification)

2.5 Measuring metaheuristic performance

Following the development phase, it is essential to evaluate the algorithm's performance. The performance data will be analysed statistically.

In the exact optimisation technique, the algorithm's efficiency in search time is the primary criterion for evaluation since it guarantees the optimal global solution. Three types of performance metrics may be used to classify a metaheuristic's performance: solution quality, computational effort (such as CPU time, wall clock time preprocessing/postprocessing time, etc.), and robustness. Additional qualitative requirements such as development expense, simplicity, ease of usage, adaptability (general applicability), and maintainability can be applied.

The performance metric used to define the precision of a solution is usually dependent on the distance or per cent variance of the obtained solution from one of the following solutions (Figure 2.12).

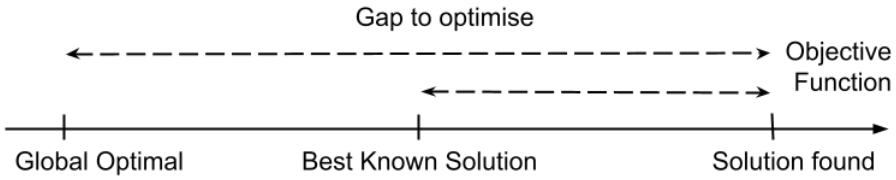


Figure 2.12: The performance assessment of a solution (minimisation problem)

The usage of a global optimum solution approach enables a more precise evaluation of the performance of various metaheuristics. Alternatively, the absolute difference can be described as $|f(s) - f(s^*)|$ or $|f(s) - f(s^*)| / f(s^*)$. where s denotes the obtained solution and s^* denotes the optimum global solution.

The optimal global solution may be discovered using an exact algorithm or may be discovered using "constructed" instances under which the optimal solution is defined a priori. Unfortu-

nately, optimum global methods are not always available for many complicated problems. This condition is sometimes tackled by using a Best Known Solution as the quality measurement.

Once experimental results for solution quality are collected, statistical test methods may be used to perform a comparative study of the designed metaheuristics and the benchmark algorithms. Numerous statistical tests may be used to evaluate and compare metaheuristics. Statistical analyses are used to determine the degree of confidence in the results' scientific validity. The statistical hypothesis testing method is chosen based on the characteristics of the results. (Refer to Figure 2.13).

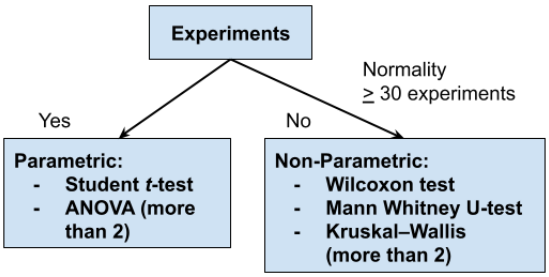


Figure 2.13: The selection procedure of a statistical test

The paired t-test is the most commonly used test under normality conditions. Additionally, confidence intervals (CI) and standard deviation may be used to show the experiment's reliability. Confidence intervals are used to approximate the range of experimental values. In practice, the majority of confidence intervals are expressed at the 95% rate. Otherwise, a nonparametric study such as the Wilcoxon test or Mann Whitney U-test can be performed. ANOVA and Kruskal–Wallis are well-established methods for determining the confidence in parametric and non-parametric results by comparing more than two algorithms. All tests can be stated in the form of a table (p-value) or graph (see Figure 2.14). When there are no overlapping error bars on a line, all techniques are significantly different.

$$success\,rate = \frac{number\,of\,successful\,runs}{total\,number\,of\,runs} \tag{2.10}$$

$$performance\,rate = \frac{number\,of\,successful\,runs}{number\,of\,function\,evaluations\,x\,total\,number\,of\,runs} \tag{2.11}$$

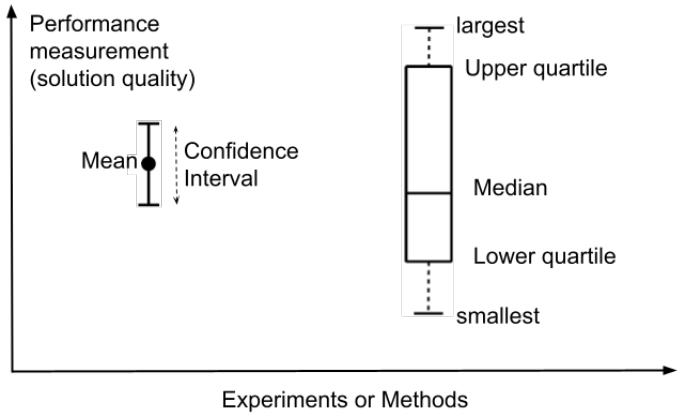


Figure 2.14: The visualisation of hypothesis statistical test

Other metrics include the success rate, which is calculated as the number of successful tests divided by the total number of trials (see Equation 2.10), and the performance rate, which accounts for the computational effort by taking into consideration the number of objective function evaluations (see Equation 2.11) (Talbi, 2009).

2.6 Summary

As can be seen from the bees traplining behaviour outlined above, the biologists discovered three unique behaviours that could be lead to improvements to the Bees Algorithm. The bees traplining behaviour studies are the primary source of searching potential new concepts. Using nature as a source of ideas has to concern the equilibrium situation between intensification and diversification. The equilibrium situation between intensification and diversification is not as straightforward as ensuring that the population size of specific components is likely comparable. The solution's perturbation has a more profound impact. The significance of the perturbation is self-evident: a perturbation that is too small may prevent the system from escaping the basin of attraction of the recently found local optimum. On the other hand, an excessive perturbation results in the algorithm reflecting a random restart local scan.

Three concepts could be a potential solution for the development of the Bees Algorithm retrieved from bees' traplining foraging behaviour.

1. A Bee does exploration and exploitation in the same foraging bout. This behaviour could

lead to a reduction in the number of parameters to be tuned which in the basic version of the Bees Algorithm is six. Although parameter tuning enables greater stability and robustness, it necessitates meticulous initialisation. Tuning these six parameters is considerably more time consuming and difficult to achieve the same level of flexibility and robustness as with less initial parameters. In the basic version, the parameter e describes the number of best sites, the m is the number of potential sites and $n - m$ is the number of poorest sites. The worker bees will exploit the e and $m - e$ sites, and explore on the $n - m$ sites. The global search and neighbour search mechanisms use a uniform random distribution with the maximum and smaller size (ngh) of solution space. According to Blum and Roli (2003), the global search could be the larger perturbation of the local search mechanism. This definition could lead to the integration of searching processes of BA. All the sites could be assigned based on the perturbation set. For more promising patch, the worker bees will focus more on exploitation than exploration, while the less one will be explored. The sites do not have to separate by e and m parameters. How to integrate the searching mechanism of BA? Does the integration affect the performance of BA?

2. Bees consider not only distance in the selection of successive unvisited flower when generating a trapline. As we know from the previous section, the traplining foraging behaviour represented the TSP in the optimisation problem. So the specific finding of the bees traplining behaviour could contribute to the enrichment of CBA. Strauss in 2004 and Gibson et al. in 2007 initially considered the traplining behaviour of bees as a simple nearest neighbour heuristic, but later Lihoreau *et al.* (2012b) work present more complex than that. The nearest neighbour heuristic, which is considered a representation of the standard animals (include bees), could be enriched by considering additional factor like the turning angle. The proposed enrichment will make the CBA have a higher diversification rate on the initial solution supplied to the local search mechanism. Yang *et al.* (2014) showed that the balance of diversification and intensification would lead to a more accurate final solution. How can CBA generate an initial solution considering the distance and turning angle? Is the higher diversification rate on the initial solutions could impact the

accuracy performance?

3. Bees do ignore a flower by chance for some reasons. Temporarily, the experienced bees will not revisit either the empty flower or the flower occupied by an intruder or competitor (Lihoreau *et al.*, 2012b). The temporary forgetting flower behaviour could add a perturbation to the foraging process. The neighbour search mechanism could have a chance to examine a complete solution (as in the basic version) and a partial solution. This balancing behaviour could make the algorithm more likely to visit the unvisited area of the solution space that can improve the ability to escaping a local optima trap. How can CBA with additional perturbation on the neighbour search mechanism escape from the local optima?

Chapter 3

Reduction in the number of Bees Algorithm parameters using triangular distribution by integrating exploration and exploitation

3.1 Preliminaries

This chapter focuses on reducing the parameter set while preserving the performance of the algorithm. The proposed approach followed the bees' traplining behaviour studied intensively by Ohashi and Thomson (2012), Lihoreau *et al.* (2012b) and Woodgate *et al.* (2017). It revealed that bees did an exploration behaviour while exploiting a patch based on the resource distribution. This proposed version has fewer initial parameter by following the integration of the bees' foraging process (exploration and exploitation). This proposed version was compared using seventeen benchmark functions for 50 independent runs. The results showed a similar performance with the basic BA. The Appendix F contains the Matlab code for this work.

Preparing a metaheuristic is a time-consuming and challenging process. All metaheuristic algorithms have parameters that must be tuned in order for the algorithm to have the optimum

solution. If the number of parameters to change rises, the preparation gets more complicated (Riff and Montero, 2013). Previous work (Pham *et al.*, 2005, 2006c) has shown that in order for the BA's basic version to run, a significant number of tunable parameters must be set. The basic BA has six initial parameters (besides the stopping criterion), which requires an inconvenient process to tune the parameters, particularly for new users (Pham and Darwish, 2008a). This work suggested a BA variant with fewer tunable parameters to make it more convenient for the user.

Previously, there was study that sought to simplify the BA's parameters. Pham and Darwish (2008a) proposed the Enhanced Bees Algorithm (EBA) which utilising a fuzzy scheme that produced the initial parameters automatically using only two initial parameters: the number of scout bees (n) and the maximum number of worker bees (nw). The fuzzy greedy selection algorithm can dynamically recruit worker bees to the chosen patch or local search places: the more potential sites, the more worker bees. The recruitment mechanism determined the number of worker bees for each site based on fitness evaluation and patch ranking. The fuzzy operator selects the potential patches m, and recruits the worker bees for all selected patches $nw_1, ..., nw_m$ while the ngh (neighbour search range) is initialised to the maximum range of solution value divided by two and then shrinking for each iteration.

In this part, a simpler version of the BA is proposed that does not rely on a system to generate certain initial parameters. The proposed algorithm integrates the searching mechanism of exploration and exploitation by using a different of random distribution generators. The idea was influenced by recent observations of bees' traplining foraging behaviour by Woodgate *et al.* (2017), Ohashi and Thomson (2012), and Lihoreau *et al.* (2012b). They discovered that bees carry out exploration when exploiting a site's resource. By combining those two main searching mechanisms, this concept can simplify the initial parameters setting of the algorithm. This integration of searching procedures would result in a different reduction approach of BA parameters than EBA.

3.2 Benchmark versions of Bees Algorithm

Apart from the basic BA explained previously, two different comparable BA versions will be compared to the proposed algorithm. The Enhanced BA versions use a fuzzy scheme to produce the parameters, while the abandonment strategy is used in the Standard BA version. The primary justification for comparing to such versions is to do a fair comparative study and demonstrate that the reduction would not jeopardise the algorithm's accuracy or speed.

3.2.1 Enhanced Bees Algorithm (2008)

The Enhanced Bees Algorithm (EBA) is based on fuzzy greedy selection and hierarchical abandonment (Pham and Darwish, 2008a). Only two parameters are needed to run the algorithm: the number of scout bees (n) and the maximum number of worker bees in each patch (nw). These two parameters could be identical. The initial size of patches (ngh) is proportional to the size of the search space. Local search is initially described over a wide area and is characterised by an exploratory nature. If the algorithm progresses, a more intensive search is needed to refine the currently selected local optimum. As a result, the search becomes more exploitative, and the region around the optimum is thoroughly scanned. As a result, each explored patch is associated with a piece of local memory. This memory is transmitted by worker bees clustered together in a field. It stores the patch's peak fitness and the patch's most recent size. The EBA's pseudo-code is represented in its simplest form in Algorithm 9.

3.2.2 Standard Bees Algorithm (2009)

This version of BA enhances the basic BA with two new procedures that improve the algorithm's search precision and exclude superfluous computations (Pham and Castellani, 2009). This variant is referred to as the Standard Bees Algorithm (SBA). The new procedures include neighbourhood shrinkage and a plan of abandonment. When no improvement is obtained by exploiting all patches using the neighbourhood shrinking process, the new procedure is used. Any time the exploitation fails to provide a lower solution, the neighbourhood search size is reduced.

Algorithm 9: Enhanced Bees Algorithm

input : n = number of scout bees; nw = maximum number of worker bees on the selected sites; ngh = the neighbour search range.
output: *Bees*

```
Start;
Initialise n scout bees population, randomly searching for sites or patches;
Evaluate the fittness of the sites;
Form Fuzzy Greedy System with initial value of fitness and rank;
while termination criterion not satisfied do
    Select m sites for neighbourhood search;
    Recruit worker bees (nw) for selected sites;
    Exploit all selected sites inside ngh range and evaluate fitnesses;
    Select the fittest bee from each patch;
    Assign remaining bees to explore randomly and evaluate their fitnesses;
    Updates Fuzzy Greedy System parameters;
end
Report the best Bees;
End;
```

After a specified number ($stlim$) of consecutive stagnation iterations, the patch is abandoned, and a new random solution is produced. If the abandoned site matches the best-so-far fitness score, the peak's position is registered. If no other flower patch produces a more accurate fitness measurement over the course of the search, the best fitness position previously reported is used as the final solution. The SBA's pseudo-code is represented in its simplest form in Algorithm 10.

3.3 Searching mechanism of Bees Algorithm

In BA, the search mechanism is divided into two distinct processes: exploitation and exploration. The terminologies can apply to mechanisms for local and global search in the solution space. The basic version is often more concerned with exploitation than exploration (Pham and Darwish, 2008a). In basic BA, the main portion of the exploration mechanism uses an uniform random number to approach space's full edge. The scout bees with the lowest rank visit the fewest foraging areas (or call the remaining bees). The EBA retains this mechanism and generates the initial parameters using a fuzzy operator.

Algorithm 10: Standard Bees Algorithm

input : n = number of scout bees; e = elite sites; m = selected sites; nep = worker bees on the e sites; nsp = worker bees on the $m - e$ sites; ngh = the neighbour search range; $shrink$ = shrinking rate; $stlim$ = limit of stagnation cycles for site abandonment.

output: *Bees*

```
Start;
Initialise n scout bees population, randomly searching for sites or patches;
Evaluate the fittness of the sites;
while termination criterion not satisfied do
    Select m sites for neighbourhood search;
    Recruit worker bees (nep and nsp) for selected sites, more bees on e sites;
    Exploit all selected sites inside ngh range and evaluate fitnesses;
    Select the fittest bee from each patch;
    Assign remaining bees to explore randomly and evaluate their fitnesses;
    Shrink the patch size when it is failed to have lower point;
    Abandon site when trapped in a local optima;
end
Report the best Bees;
End;
```

The BA and EBA continue to exploit and explore on m and $n - m$ locations independently. Bees do not hire bees or have followers on $(n - m)$ sites because the site or patch is less enticing to manipulate. In the standard and basic versions, the exploitation process manipulates the honey on potential sites using the uniform distribution number (ngh). The most promising places, called elite sites (e), would be exploited by elite worker bees (nep) recruited by a special dance by the scout.

The remaining potential locations, named selected non-elite sites ($m - e$), would be worked by additionally hired worker bees (nsp). The potential sites are decided using greedy sorting or ranking based on the objective function assessment of all scout bees or patch positions. A random number generator with a uniform distribution is used in both global and local searches with varying neighbour search radiuses.

This (searching) element of BA will be modified systematically in this study. The basic variant operator, which used a random number generator with a uniform distribution, would be substituted by a random number generator with a triangular distribution. By varying the searching intention for each patch, the more promising the patch, the more intent on exploita-

tion; the new generator will assign more bees to search deeply near the patch and fewer bees to explore around it. This integration searching strategy is compatible with worker bees' natural concurrent exploration and exploitation behaviour, as defined in the following section.

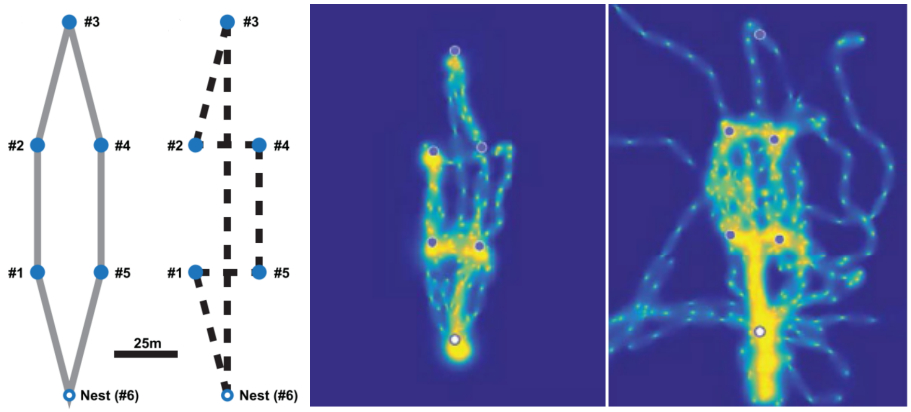
3.4 Bees do exploration and exploitation in the same foraging trip

Honey bees are only one of the many species that forage in colonies or groups. In other terms, success is based upon the collective behaviour of all foraging member activities. Pollen and nectar from flowers that bloom within the flight range of their home provide food for bees. Bees usually design a circuit to visit many flowers in such a manner that the total travel distance is minimised.

Ohashi and Thomson (2012), Lihoreau *et al.* (2012b), and Woodgate *et al.* (2017) conducted experiments on this routing behaviour and concluded that bees do possess the ability for routing optimisation. The researchers examined how bees build and optimise circuits on vast spatial scales using cutting-edge technologies called harmonic or heat radar. They created an animated sonar or heat map that illustrates how many route segments were habitual while others were abandoned until the desired flight path was discovered. It demonstrates how bees use a combination of observation, learning, and sequential optimisation to establish stable routes between flowers. Bees depend on their vision and scent in addition to their small brain to memorise a spatial memory of their position, reward value, and flower sequence (Ohashi and Thomson, 2012). Lihoreau *et al.* (2012b) quantified the similarity of bee foraging visits to the optimal sequence, which improves dramatically with time and experience.

Woodgate *et al.* (2017) assert that worker bees exploit and explore the patch. They would explore the field in compliance with the region's resource distribution. The bees will explore the patch if the honey supply runs out. As shown in Figure 3.1b, one bee is devoted entirely to exploitation (left), while the other is equally devoted to exploitation and exploration (right). According to the heat radar report, the bees' exploitation pattern is identical to that of the

straight path series. By contrast, the dominated exploration trend will have routes that are not part of the array.



(a) The array (b) Tracking using radar

Figure 3.1: The bee's foraging behaviour (Woodgate *et al.*, 2017)

Based on the Woodgate's finding, the proportional searching intentions assumption may be applied. The degree of intensification can vary between locations depending on the availability of the resource (see Figure 3.2). On the most promising patch-1, the bee focuses on exploitation. Bees will mainly take the straight array route in more promising areas, which means they will spend the majority of their time exploiting. If the bees are inside the least good patch, they are more interested in exploration than exploitation. Consequently, we can conclude that bees perform both local and global searches based on the promising parameters of the sites. This principle would reduce the parameters of the Basic BA significantly. It is possible to eliminate the initial classification of workers and patches. It is no longer necessary to classify e and m locations since they can be captured during the patch ranking process.

3.5 The triangular distribution searching mechanism

The strength of a triangular distribution is its ability to accurately represent random conditions when there is insufficient information (Law, 2013). This distribution can be used to define a wide range of random conditions with straightforward parameters. Three fundamental parameters are: the minimum (a), likely (b), and maximum (c) values.

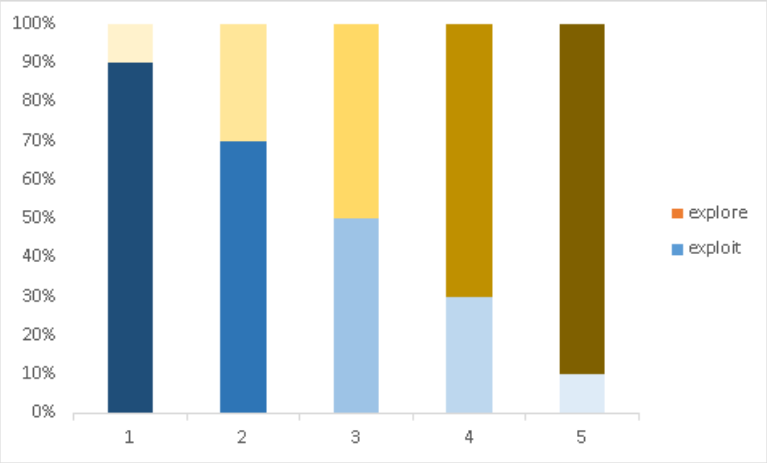


Figure 3.2: The ratio of exploration and exploitation of 5 different patches

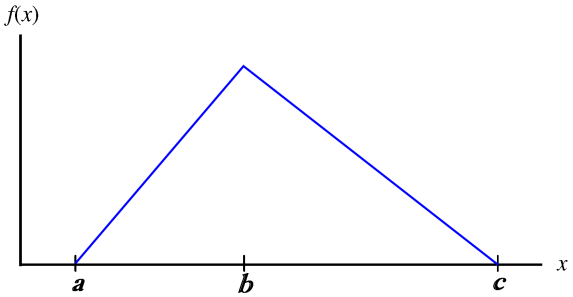


Figure 3.3: The triangular distribution on the interval $[a, c]$ with mode b

By changing the likely value according to the potential rank, the adaptability to many random conditions can be used to define the integration ratio of exploration and exploitation. When b is set close to the patch coordinate, worker bees will spend the majority of their time exploiting and very little time exploring in a region. When b is far to the patch position, worker bees, on the other hand, can explore more than they can exploit. The term "assignment" refers to this process of prioritisation. Furthermore, the number of worker bees will be determined by their significance ranking. The more visually appealing a location is, the more bees it attracts.

Due to the preliminary nature of this report, the basic recruiting and assignment principles would be focused on linear interpolation of the fitness value. Every worker bee has a different likely value (b) in the search range (0–100%). This indicates that the search distribution has a minimum of 0 and a maximum of 1. The assignment for the elite candidate site would have a value close to 0, indicating that the colony will prioritise exploiting the neighbouring patch

location. The bees on a less desirable location would most definitely have a value of close to 1, indicating that the bees would be concentrating on exploring the neighbouring patch spot. As a result, the workers at all locations are simultaneously engaged in exploration and exploitation.

3.6 The proposed algorithm

Unlike the EBA in 2008, which employed a fuzzy greedy method to generate site selection and worker recruiting parameters, this proposed version would reduce the original parameters by merging the BA's searching processes (exploration and exploitation) using triangular distribution random number generator. Based on the rank of the fitness evaluation, simple linear interpolation can be used in the recruitment and assignment procedures.

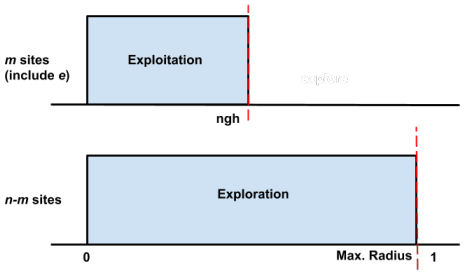


Figure 3.4: The searching of the basic Bees Algorithm using the uniform distribution

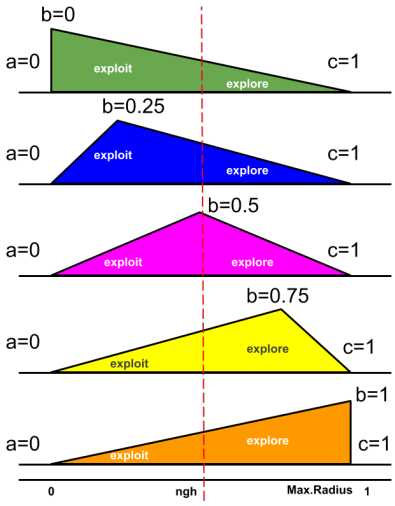


Figure 3.5: The triangular distribution searching assignment

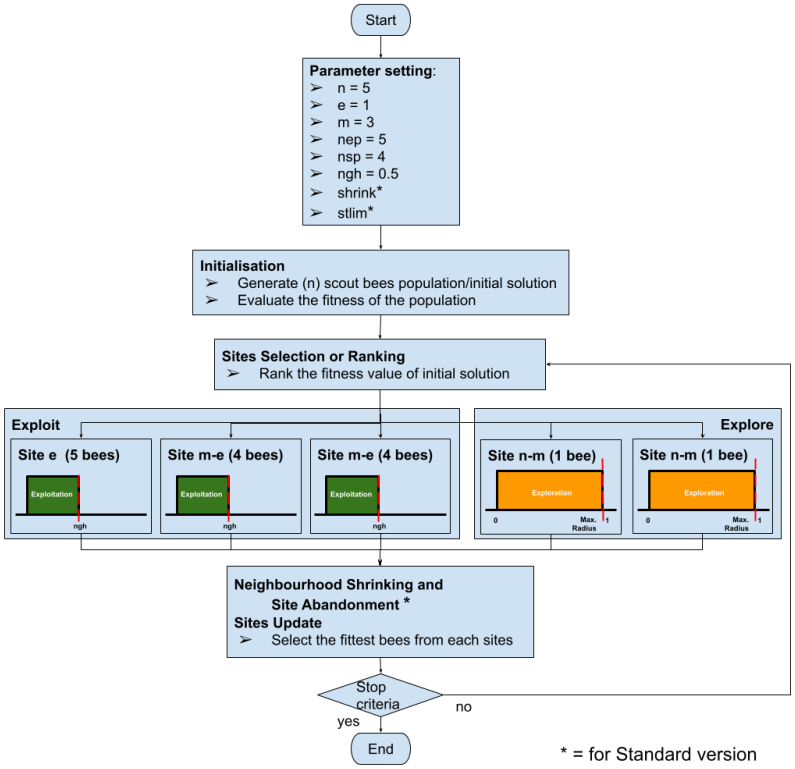


Figure 3.6: The flowchart of the Basic and Standard(*) Bees Algorithm with a uniform distribution searching mechanism

As previously mentioned, the basic variant of BA conducts uniform random searches during the exploitation and exploration processes. The exploratory uniform random search can be viewed as a complete neighbour search (covering 100% of the search space) (see Figure 3.4). The suggested approach would not use this distribution rather than a triangular distribution since it can deal with arbitrary exploration-exploitation ratio assignments. This integration principle, which necessitates a stronger exploitation on the more promising position, can be sufficient when Gaussian (including normal and beta) and triangular distributions are used.

However, Gaussian distributions seem impractical due to their complexity of nature. To construct a normal distribution, for example, a mean and standard deviation value are required. On the other hand, the triangular distribution is the most straightforward method of establishing a random distribution. Due to its flexibility, this method is well-suited for representing a system for which data collection is difficult (Law, 2013). The triangular distribution needs three parameters (a, b, and c), of which only b must be tuned because a and c are predefined in the

algorithm. The searching space's minimum size (a) must be zero per cent, and its maximum size (c) must be one hundred per cent. This random assignment would give each worker bee a mission, either to exploit or to explore. By placing the b value close to the patch spot, the high intensity of exploitation in the searching phase can be achieved (see Figure 3.5).

Besides the assignment, recruitment (to ascertain the number of worker bees in each patch) is needed to carry out the foraging processes. The protocol for recruiting and assignment is based on basic linear interpolation of given data (see Figure 3.9). The proposed version retains the original ascending (minimisation) orderings for determining the rank of sites and identify potential sites. In its simplest form, Algorithm 11 represents the pseudo-code for the BA and SBA with triangular distribution assignment, which we can abbreviate as Bi-Parameter (S)BA.

The difference between BA, EBA, and the proposed form is depicted in Figure 3.6, 3.7, and 3.8. The significant differences between these two algorithms are their search mechanisms, with the proposed implementation using a triangular distribution operator. The Matlab code of this proposed version is available in the Appendix F.

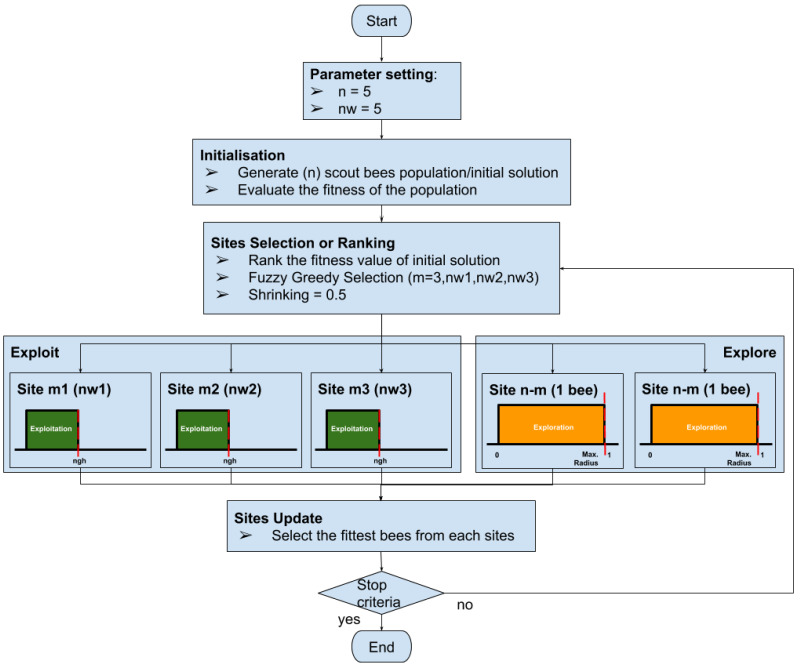


Figure 3.7: The flowchart of the Enhanced Bees Algorithm

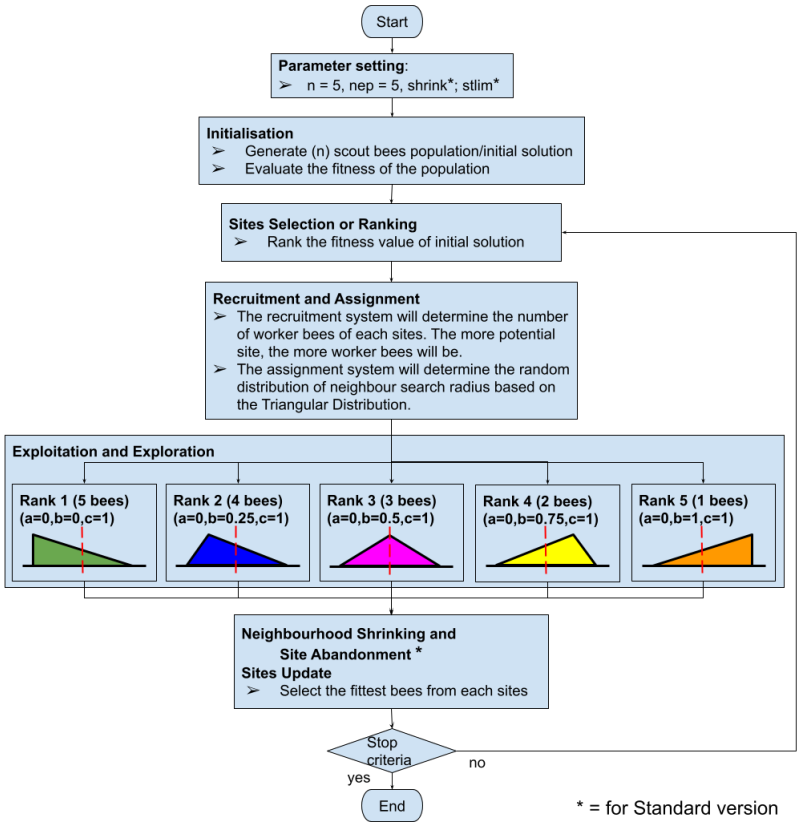


Figure 3.8: The flowchart of the Bi-Parameter (Standard=*) Bees Algorithm (Bi-(S)BA)

3.7 Experiments, results and discussion

3.7.1 Experiment design

The experiment in this section was designed to determine the proposed approach's performance in comparison to the standard BA. The proposed BA version, named Bi-Parameter BA and SBA (Bi-BA and Bi-SBA), would be applied to the seventeen benchmark functions listed in (Pham and Darwish, 2008a; Pham and Castellani, 2009) and compared to three other BA variants, namely the basic BA, the EBA, and the SBA. The SBA used shrinking and abandonment strategies, whereas the BA and EBA did not. This section compared additional BA versions in order to provide a comprehensive comparison through the two comparative studies. First, Bi-BA was compared to Basic BA and EBA (Pham and Darwish, 2008a) on datasets with two to six dimensions and population sizes ranging from 28 to 70. The second comparison was Bi-SBA with SBA (Pham and Castellani, 2009), which uses up to ten benchmark functions and a population

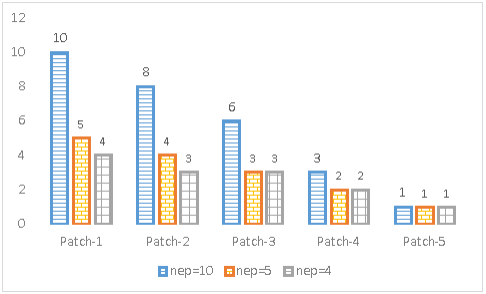


Figure 3.9: The linear interpolation of the recruitment mechanism of Bi-(S)BA

Algorithm 11: The Bi-Parameters (Standard=⊛) Bees Algorithm

input : n = number of scout bees; nep = Maximum number of worker bees on the best sites; $shrink$ = shrinking ⊛ rate; $stlim$ = limit of stagnation cycles for site abandonment ⊛.
output: *Bees*

Start;
Initialise n scout bees population, randomly searching for sites or patches;
Evaluate the fittness of the sites;
while *termination criterion not satisfied* **do**
    Recruit and assign worker bees for all patches, more bees on a more potential patch;
    Exploit and explore according to the assignment of all patches and evaluate fitnesses;
    Select the fittest bee from each patch;
    Shrink the patch size when it is failed to have lower point ⊛;
    Abandon a patch when trapped in a local optima ⊛;
end
Report the best *Bees*;
End;

scale of 105 to 120.

Seventeen benchmark functions were simulated: seven ([2,6] dimensions) for Bi-BA vs EBA and ten ([2,10] dimensions) for Bi-SBA vs SBA comparative analysis for 50 independent runs. The parameters for each scenario (three scenarios for comparative-1 and five scenarios for comparative-2) are mentioned in Table 3.1. In general, scenarios are generated using a straightforward theory for balanced and unbalanced n and nep scenarios. The initial values of ngh, $shrink$, and $stlim$ for comparative-2 are 0.5, 0.8, and 10, respectively, consistent with previous work. The equations for the benchmark functions are mentioned in Table 2.1 in the literature review section. The benchmark functions were a series of continuous minimisation function.

All running experiments has stopping criteria of 0.001 accuracy and 500,000 evaluations. After the gap between the obtained objective function and the global optimum is less than 0.001, or the number of evaluations (NFE) reaches 500,000, the simulation will terminate (Pham and Darwish, 2008a; Pham and Castellani, 2009). If a solution is found that meets 0.001 conditions prior to 500,000 NFE, the algorithm calls the condition "succeeded". Otherwise, it finds the operation "failed." After obtaining the mean difference and NFE, the statistical test was conducted using the Whisker box-plot error bar. The column bar graphs depict the accuracy, while the row bar graphs depict the results in terms of speed. The colour of the bars denotes the version and scenario. The darker colour indicates a more intense searching process scenario ($nep > n$) for the Bi-BA and Bi-SBA versions. The proposed version would be statistically compared to previous versions to determine the output gap. However, since the previous work (Pham and Darwish, 2008a) omitted the standard deviation value, the distinction was made entirely on the basis of their mean values.

Table 3.1: The scenarios of the comparative studies

	Comparison-1			Comparison-2				
Scenario	S-1-1	S-1-2	S-1-3	S-2-1	S-2-2	S-2-3	S-2-4	S-2-5
Proposed methods	Bi-BA-1	Bi-BA-2	Bi-BA-3	Bi-SBA-1	Bi-SBA-2	Bi-SBA-3	Bi-SBA-4	Bi-SBA-5
n	[5,7,9]*	[7,9,11]*	[10,12,14]*	7	10	14	20	30
nep	[10,12,14]**	[7,9,11]**	[5,7,9]**	30	20	14	10	7
population size	[28,46,68]	[28,45,66]	[30,48,70]	109	105	105	110	120

* = [n2d, n4d, n6d] = [n for 2 dimensions, n for 4 dimensions, n for 6 dimensions]

** = [nep2d, nep4d, nep6d] = [nep for 2 dimensions, nep for 4 dimensions, nep for 6 dimensions]

It should be emphasised that the experiments were not designed to establish the proposed version's supremacy over other versions of BA. The study's objective was to design a BA with fewer parameters while maintaining the basic or standard version's efficacy, efficiency and robustness. Additionally, the test can assist us in identifying the novel version's characteristics and may serve as brief instructions on how to use it. For the proposed algorithm, various learning parameter settings (scenarios) were examined to determine their impact on the search procedures.

3.7.2 Experiment result

Table 3.2 shows the accuracy result of applying Bi-BA to the first series of benchmark functions. The following Table 3.3 compares the number of functions evaluated when Bi-BA, BBA, and EBA are applied to the same benchmark set. The mean, standard deviation, and confidence interval error (p=0.05) for the benchmark version algorithms are shown in the tables. The accuracy performances of the BBA and EBA are not plotted because they fell below the accuracy criteria.

Simultaneously, the second comparison study's accuracy and convergence characteristics are summarised in Tables 3.4 and 3.5. Table 3.4 summarises the accuracy and convergence of the comparative study-2's best performance, while Table 3.5 summarises the robust performance. The conclusions in this section are based on the statistical analyses of Figure B.3,3.12, 3.13, B.9 and B.10. The box indicates the benchmark with the highest average precision crossed. If there was no statistically significant difference in the average accuracy of the two models, they were declared equal (both have the cross). Similarly, the second box indicates the fastest convergence results. From Figure 3.10 to B.10, a bar chart with an error bar indicates significant differences between versions. If the error bars associated with the confidence intervals do not overlap, the difference is statistically significant at the specified confidence level (p-value).

As illustrated in Table 3.2, all functions except Rosenbock (2D and 4D) and Griewangk have a accuracy performance mean value greater than the threshold (0.001). However, only Rosenbrock-4D does not have 0.001 accuracies with a confidence interval 95 per cent (statistical analysis in Figure 3.10). This result is consistent with previous research (Pham and Castellani, 2009), which established that these three functions (with Rastrigin) were the most challenging to solve. Bi-BA had the lowest success rate of 36-56 per cent on the Rosenbrock-4D function stand. Rosenbrock-2D and Griewangk-2D come in second and third place with 68-88 per cent and 92-96 per cent, respectively. If the algorithm counts until the maximum evaluation number is reached, the success rate will likely fail. A success rate of 100% in executing experiments guarantees an algorithm's accuracy of 0.001 or less. The statistical analysis in Figure 3.10, on

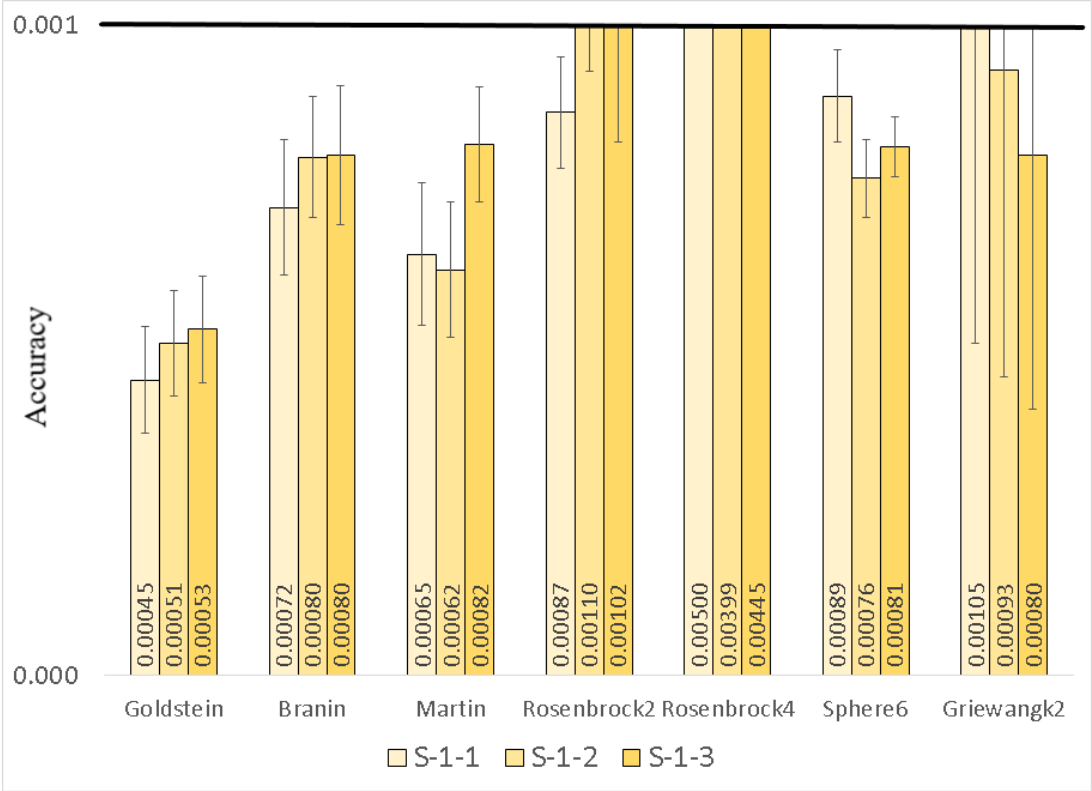


Figure 3.10: Statistical test (95% confidence level) on the accuracy of Bi-BA

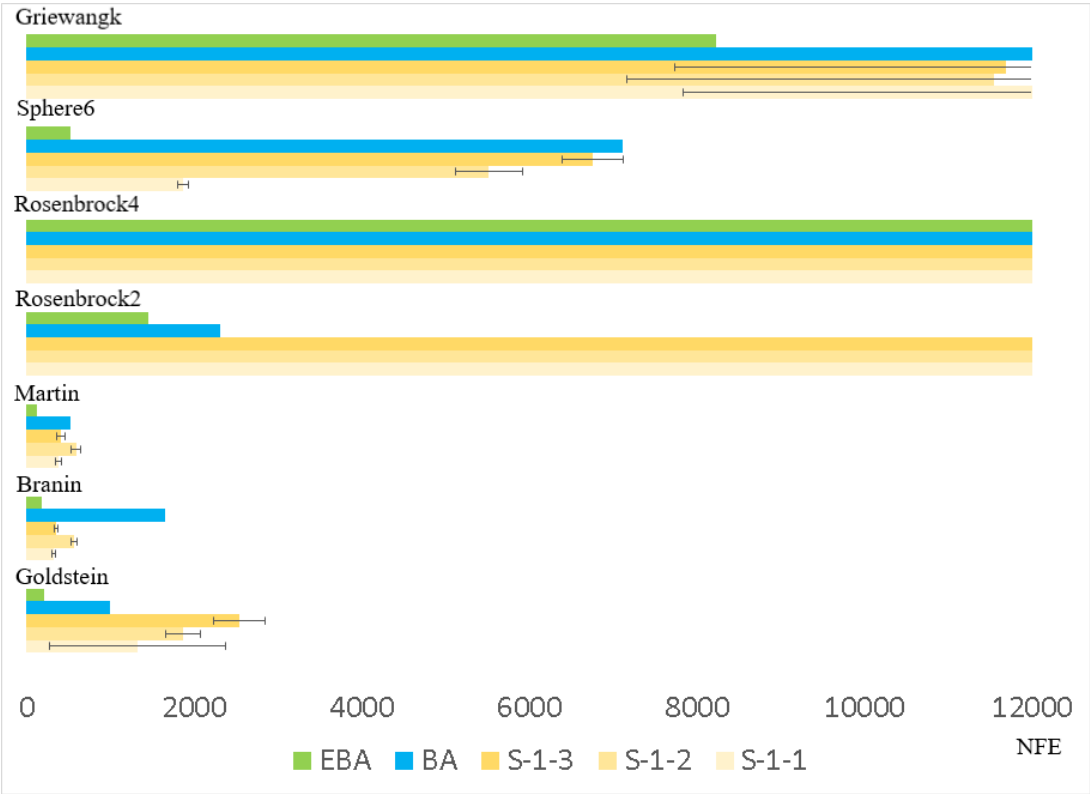


Figure 3.11: Statistical test (95% confidence level) on the number of function evaluations (NFE) of Bi-BA

Table 3.2: The summary of accuracy results of the first comparative study of Bi-BA

	S-1-1 (Bi-BA-1)				S-1-2 (Bi-BA-2)				S-1-3 (Bi-BA-3)			
ID($Dims$)	Mean	StdDev	C.I.	Succ.	Mean	StdDev	C.I.	Succ.	Mean	StdDev	C.I.	Succ.
F1(2D)	0.00045	0.00029	0.00008	50	0.00051	0.00029	0.00008	50	0.00053	0.00029	0.00008	50
F2(2D)	0.00072	0.00037	0.00010	50	0.00080	0.00033	0.00009	50	0.00080	0.00038	0.00011	50
F3(2D)	0.00065	0.00039	0.00011	50	0.00062	0.00036	0.00010	50	0.00082	0.00031	0.00009	50
F6(2D)	0.00087	0.00030	0.00009	44	0.00110	0.00060	0.00017	34	0.00102	0.00072	0.00020	37
F6(4D)	0.00500	0.00584	0.00166	18	0.00399	0.00595	0.00169	25	0.00445	0.00732	0.00208	28
F7(6D)	0.00089	0.00025	0.00007	50	0.00076	0.00021	0.00006	50	0.00081	0.00016	0.00005	50
F8(2D)	0.00105	0.00189	0.00054	46	0.00093	0.00166	0.00047	47	0.00080	0.00138	0.00039	48

Table 3.3: The summary of evaluation results of the first comparative study of Bi-BA

	S-1-1 (Bi-BA-1)			S-1-2 (Bi-BA-2)			S-1-3 (Bi-BA-3)			BA	EBA
ID($Dims$)	Mean	StdDev	C.I.	Mean	StdDev	C.I.	Mean	StdDev	C.I.	Mean	Mean
F1(2D)	1323.6	3717.4	1056.5	1868.3	735.5	209.0	2538.1	1085.7	308.6	999	212
F2(2D)	325.6	88.2	25.1	559.5	122.6	34.9	350.3	85.3	24.2	1657	184
F3(2D)	378.0	138.0	39.2	589.8	196.7	55.9	410.8	176.3	50.1	526	124
F6(2D)	24866.3	31071.0	8830.3	40691.9	44807.2	12734.1	34643.7	46626.5	13251.1	2306	1448
F6(4D)	115283.5	76246.2	21668.9	88154.1	79810.9	22682.0	83872.7	84208.5	23931.8	28529	33367
F7(6D)	1863.4	215.9	61.4	5518.2	1420.2	403.6	6756.5	1296.1	368.3	7113	526
F8(2D)	12445.9	16222.7	4610.4	11547.1	15418.4	4381.9	11682.3	13887.7	3946.8	20998	8224

the other hand, exposes only the error tails for Rosenbrock-4D that do not overlap the 0.001 lines. It is reasonable to conclude that Bi-BA accuracy is comparable to that of BBA and EBA.

In terms of convergence performance, achieving an NFE of fewer than 500,000 evaluations indicates that the experiment attained an accuracy of less than 0.001. The mean of 50 experiments equals 500,000 evaluations demonstrates that no single trial success meets the accuracy requirement. So the convergence performance analysis will solely perform statistic analysis to compare the convergence findings. According to Table 3.3, all experiments have a mean value of less than 500,000 evaluations. No function has a success rate of zero per cent. Figure 3.11 shows that the convergence performance of three functions, Sphere-6D, Martin&Gaddy, and Goldstein, is statistically comparable between Bi-BA and BBA. Bi-BA is significantly superior to BBA in two other cases (Branin-2D and Griewangk-2D). In contrast, the reverse is true for Rosenbrock-2D and 4D. On the other hand, the EBA outperforms the Bi-BA in every function except Branin, Martin&Gaddy, and Goldstein. The Branin-2D function has the best convergence performance of Bi-BA, requiring an average of 325 to 559 evaluations to achieve ≤ 0.001 accuracy. On the other hand, BA and EBA have the fastest convergence on the Martin&Gaddy function. However, Branin, Martin&Gaddy, and Goldstein all rate among the top three in terms of speed. Except for Branin-2D, there is no statistically significant difference in speed between Bi-BA and BBA. The Bi-BA outperforms BBA and is comparable to EBA on the Branin datasets. Only Griwangk-2D's error bars overlap the mean of the EBA evaluation (see Figure 3.11) and suppress (better) the mean of the BBA. The last two (Rosenbrock-4D and Rosenbrock-2D) are thought to be significantly different from BBA and EBA. We can deduce that the Bi-BA performs similarly to the BBA but not to the EBA in terms of accuracy and convergence.

Table 3.4 summarise the best outcomes achieved by the Bi-SBA and SBA (Pham and Castellani, 2009) in the second set of scenarios. Bi-SBA successfully executes all 50 independent runs for all functions except Rosenbrock-10D (76 per cent) and Griewangk-10D. (82 per cent). However, statistical analysis with a confidence level of 95% indicates that Bi-SBA success to accomplish the aim for all except the Griewank function, the error tail that not overlap the 0,001

Table 3.4: The best performance of Bi-SBA Vs SBA

		Best Performance Bi-SBA			Best Performance SBA		
	Scenario	succ.	Acc	NFE	succ.	Acc	NFE
Easom (2D)	S-2-5	50	X	X	50	X	X
Schaffer (2D)	S-2-3	50	X	X	50	X	X
Martin and Gaddy (2D)	S-2-4	50	X	X	50	X	
Goldstein and Price (2D)	S-2-3	50	X	X	50	X	
Schwefel (2D)	S-2-2	50	X	X	50	X	X
Rastrigin (10D)	S-2-1	50	X	X	0		
Rosenbrock (10D)	S-2-1	38	X	X	0		
Sphere (10D)	S-2-3	50	X	X	50	X	
Ackley (10D)	S-2-5	50	X	X	50	X	
Griewangk (10D)	S-2-3	41		X	36	X	
Total		479	9	10	386	8	3

Table 3.5: Comparison of robustness (Bi-SBA Vs SBA)

		Robust Performance Bi-SBA			Robust Performance SBA		
	Scenario	succ.	Acc	NFE	succ.	Acc	NFE
Easom (2D)	S-2-3	50	X	X	50	X	X
Schaffer (2D)	S-2-3	50	X	X	50	X	X
Martin and Gaddy (2D)	S-2-3	50	X	X	50	X	
Goldstein and Price (2D)	S-2-3	50	X	X	50	X	
Schwefel (2D)	S-2-3	50	X	X	50	X	X
Rastrigin (10D)	S-2-3	46	X	X	0		
Rosenbrock (10D)	S-2-3	26	X	X	0		
Sphere (10D)	S-2-3	50	X	X	50	X	
Ackley (10D)	S-2-3	50	X	X	50	X	
Griewangk (10D)	S-2-3	41	X	X	10	X	
Total		463	10	10	360	8	3

line (see Figure 3.12). While SBA has three functions that have failed to reach 100% success rate, they are Rastrigin-10D with 0%, Rosenbrock-10D 0%, and Griewangk-10D 72%. Even though the SBA has a success rate of just 72% for the Griewank-10D, lower than the Bi-SBA, the mean value is lower and statistically better than the Bi-SBA's. To ensure consistency, on Rosenbrock-10D and Ratrigin-10D, the Bi-SBA outperforms the SBA, while the SBA outperforms with the Griewangk-10D. The Bi-SBA achieves the fastest solution across all functions in terms of speed, while the SBA achieves three comparable speeds (Easom-2D, Schaffer-2D, Schwefel-2D) (see Figure 3.13).

Table 3.5 summarise the robust results from the Bi-SBA with S-2-3 and the SBA with

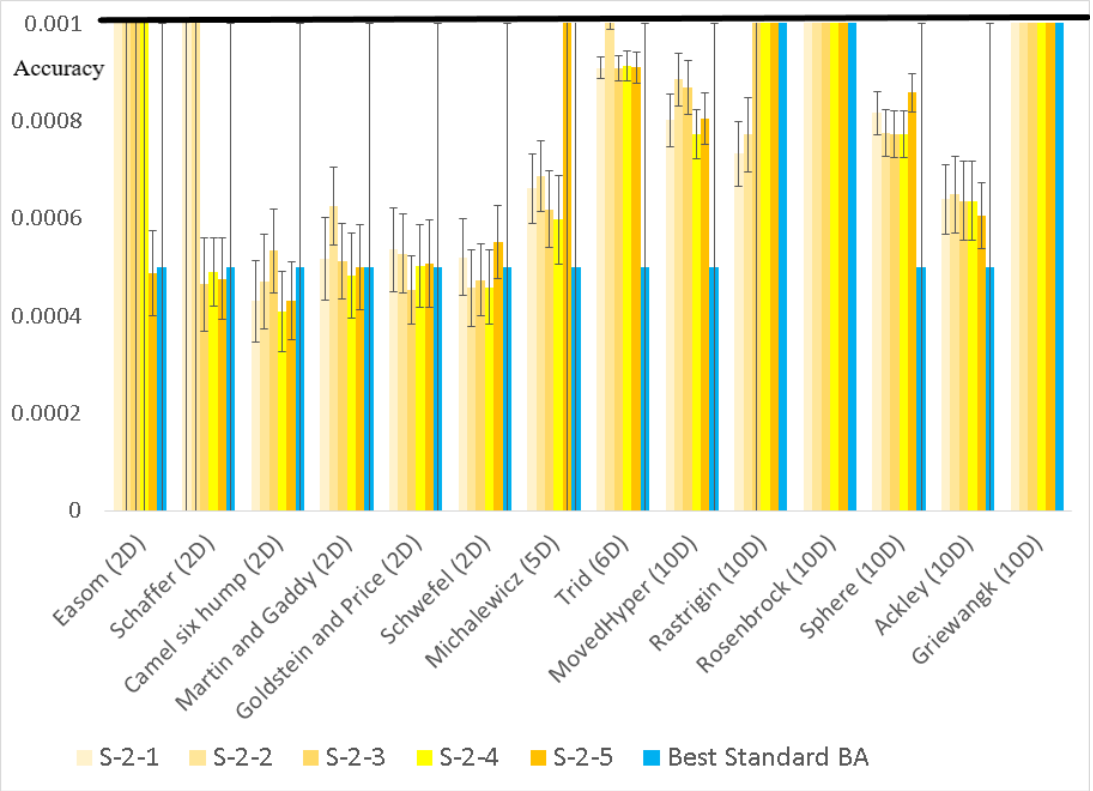


Figure 3.12: Statistical test (95% confidence level) on the accuracy of Bi-SBA

scenario-1 ($e = 2, nep = 30, m - e = 4, nsp = 10$) (Pham and Castellani, 2009). 80% of the functions of robust Bi-SBA and SBA are equivalent. Bi-SBA is preferable to the remainder, Rastrigin-10D and Rosenbrock-10D. When Rastrigin-10D was examined, significant differences were seen. The best and robust SBA can reach precision at 7.4821 and 8.8201, respectively, whereas the Bi-SBA can approach the accuracy criterion (see Figure B.3 and B.4 in the appendix).

3.8 Summary

Bi-(S)BA, a BA with two setting parameters, was applied to the benchmark functions specified in (Pham and Darwish, 2008b; Pham and Castellani, 2009) and compared to three other BA versions, namely the basic BA, the EBA, and the SBA. This chapter attempted to produce a BA with fewer parameters while retaining the basic or standard version's performance and robustness. By replacing a triangular distribution for uniform distribution in order to decrease

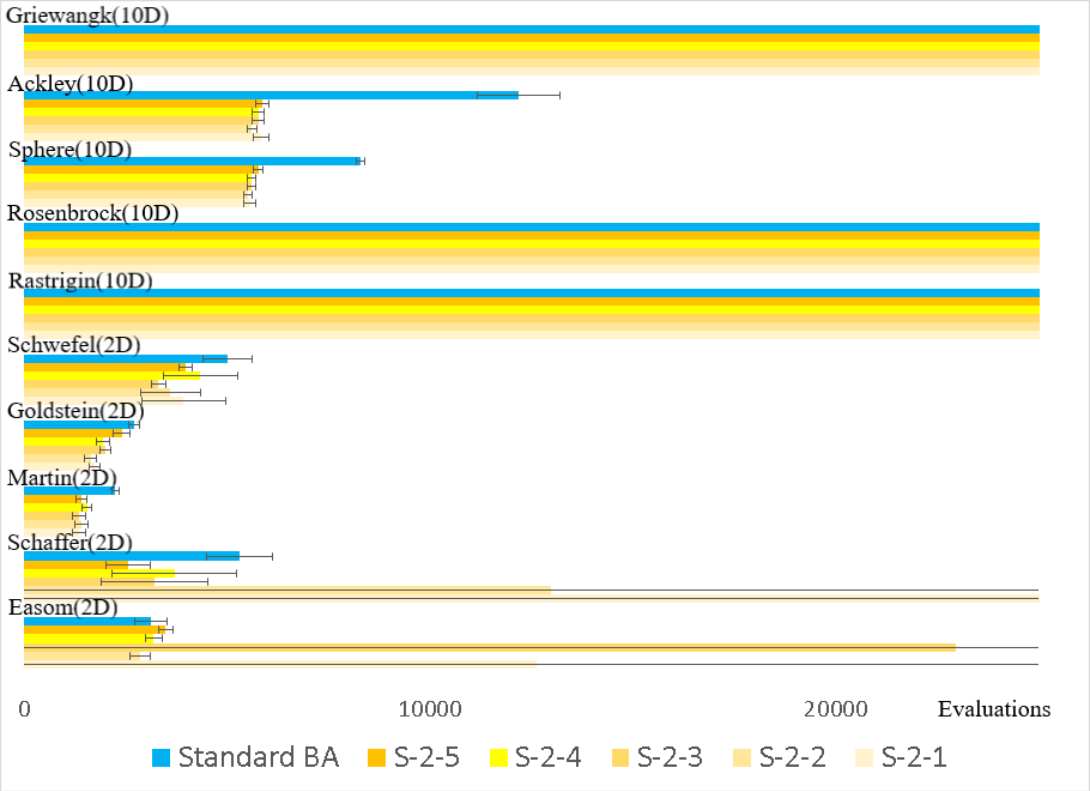


Figure 3.13: Statistical test (95% confidence level) on the NFE of Bi-SBA

the starting parameters, this suggested technique combines the exploitation and exploration search for BA. The triangular distribution has replaced the typical uniform distribution, which is more adaptable to the new way of exploitation and exploration. The outcome indicated that the Bi-BA and Bi-SBA have fewer tuneable initial parameters but maintain the same degree of performance as the basic and standard versions. As seen in the results section, the proposed Bi-BA and Bi-SBA perform comparably to the basic and standard versions on the majority of benchmark functions. However, Bi-SBA has a higher success rate than SBA for the three challenging functions -Rastrigin, Rosenbrock, and Griewangk- identified by a prior study (Pham and Castellani, 2009). The research's shortcoming is its use of unfair comparisons. A confidence interval's error tails should not be comparable to the mean value in the statistic analysis. This comparison was conducted due to the prior study's inadequate data.

Chapter 4

Novel exploration strategy of Combinatorial Bees Algorithm

4.1 Preliminaries

This chapter focuses on the ability to reach high-quality initial solutions as a good starting position. The chapter suggested a novel exploration technique for increasing the probability of encountering the optimal solution by diversifying the initial solution. The proposed approach, called Bee's Nearest Straight Neighbour Heuristic (BNSN), is inspired by how bees generate initial traplining when they have no experiences yet. TSP as test function was used in this development. Other applications, such as the VRP and PCB assembly problems, will be evaluated in the following chapter. The novel exploration (initialisation) operator's performance will be compared to the normal instinct of the animal's routing plan, the Nearest Neighbourhood Heuristic (NNH). The Appendix G contains the matlab code for this work.

Numerous efforts have been made recently to improve the efficiency of population-based metaheuristic. For combinatorial domains, the efficiency of a nature-inspired algorithm can depend on the initial solution or starting point (Yang, 2020). Diversification is the primary requirement to consider when generating the initial population. If the initial population is not sufficiently diverse, every population-based metaheuristic will experience premature conver-

gence. It could happen, for example, if the initial population is created using a greedy heuristic or a single-based metaheuristic for each population solution (Talbi, 2009).

That is why the majority of researchers combine a metaheuristic with a constructive heuristic to improve results. A constructive heuristic is a form of heuristic that begins with nothing and gradually constructs the initial solution (Ismail, 2019). Though constructive heuristic algorithms are frequently the quickest approximate methods, their solutions are frequently imprecise (Dorigo and Stützle, 2019). Later, this imprecise approach may be improved using a local search algorithm or neighbour search mechanism. The most well-known success story involving applying a constructive heuristic to a metaheuristic algorithm for TSP is Ant Colony System with NNH, which was published by (Dorigo and Gambardella, 1997).

The NNH, which was previously assumed to be the fundamental routing instinct of humans and animals (bees), is no longer valid (Kyritsis *et al.*, 2018). There were counter-arguments from biologists who have conducted extensive research on bees' trapline behaviour (Lihoreau *et al.*, 2012b; Ohashi and Thomson, 2012; Buatois and Lihoreau, 2016; Klein *et al.*, 2017; Woodgate *et al.*, 2017). They discovered that bee performs routing optimisation and generates their initial trapline by considering distance and angle. Trapline foraging is characterised as a series of predictable visits to a series of resource points or patches. Traplining has been observed in a variety of different species and has mostly been associated with foraging on natural resources (e.g. nectar, fruits, and leaves) (Berger-Tal and Bar-David, 2015).

The discovery of how bees generate initial tours motivated the author to propose a constructive heuristic as a component of CBA. Bee's Nearest Straight Neighbour Heuristic (BNSN), inspired by bees' traplining foraging behaviour, was proposed as a novel strategy for developing an initial solution operator. It is designed to take the place of CBA's initial solution generator, which employs a random permutation number generator (RNG). In a continuous version of BA, the random number represents the initial solution as the bees' random location when looking for food. By comparison, the combinatorial solution is the sequence of all flowers visited. Using a random number generator to initialise the CBA does not accurately represent bees' foraging behaviour, as bees do not randomly organise their initial tour. According to previous research,

bees follow the NNH procedure described previously. Subsequent researches demonstrate that bees consider not only distance but also turning angle and spatial memory when produce a foraging tour. However, when the initial tour is constructed, the bees' experience-based information must not have existed, such as spatial memory. The proposed method was inspired by bees' distance and angle selection procedures during their initial tour generation. The experiments in this study will determine how well the proposed approach fits CBA as a constructive heuristic and whether diversity contributes significantly to the final solution's accuracy.

This study conducted a preliminary experiment to assess the diverse performance of constructive heuristic methods as initial solution generators, followed by the primary investigation to evaluate the overall performance of the proposed strategy using the CBA. The preliminary experiments were carried out with a basic hexagonal array (Ohashi and Thomson, 2012), while the primary investigation was carried out with TSPLIB datasets (Reinelt, 1991). The result will be compared to that of other CBA variants (basic CBA and CBA+NNH) using up to 200 cities instances.

4.2 Bees' initial foraging tour behaviour in nature

According to Ohashi and Thomson (2012), Lihoreau *et al.* (2012b), Woodgate *et al.* (2017), when creating the initial tour, the bees did not prioritise the order of visits that would result in the shortest total tour. Rather than that, they prioritised going to the nearest flower, owing to their inexperience. If another straighter flower is in their line of vision, the bees will avoid the sharper turn on the second flight. As illustrated in Figure 4.1, the bee will choose flower-4 over flower-3 for her next visitation due to its more linear angle. Turning the flower-3 looks to demand more motoric effort, whilst turning the flower-4 appears to require less.

Although bees take distance and angle into account (see Figure 4.3), Ohashi *et al.* (2007) revealed that they prefer shorter movements over straighter ones. He examined this behaviour through the perspective of arrays' positive, negative, and independent principles. In the "positive" array, proximity and directionality were positively associated, indicating that the nearest neighbour could be reached with straight-ahead motions (the yellow line-the optimal route is

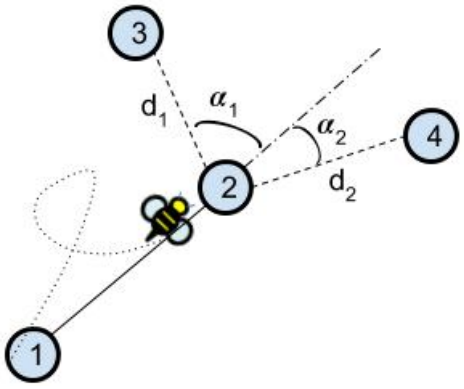
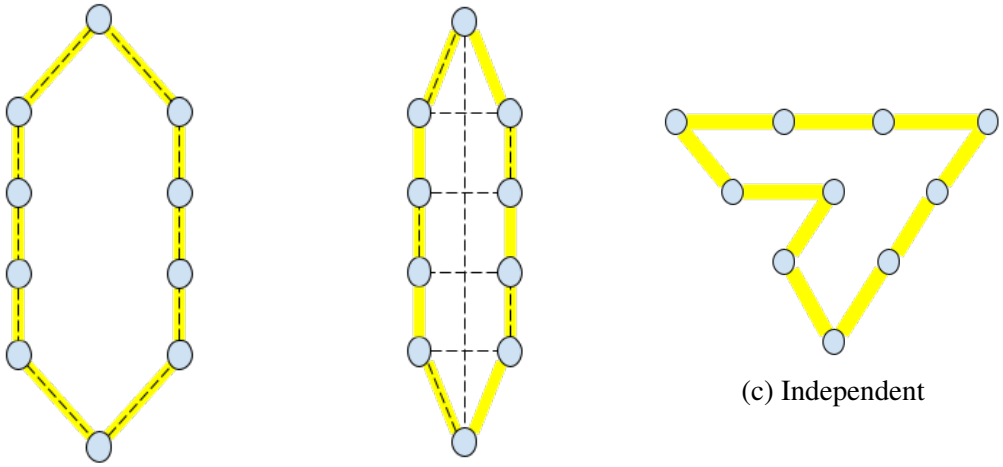


Figure 4.1: Bees consider the distance and angle

equal to the black dashed line- the nearest neighbour route). On the other hand, in the "negative" array, closeness and directionality were inversely connected, implying that bees may make a sharp turn by selecting nearest neighbours. While flowers in the "independent" array frequently had two–six equidistant closest neighbours in opposite directions, bees had complete freedom of travel distance and turning angle selection (see Figure 4.2).



(a) Positive (b) Negative (c) Independent

Figure 4.2: The experiment arrays (Ohashi *et al.*, 2007)

Both distance and angle had high relative ranks in the positive array, indicating that selections of nearest neighbours were compatible with choices of straightest motions. In the negative array, where nearest neighbours' choices contradicted choices of straightest moves, the angle had a substantially lower relative rank than distance. In the independent array, bees selected distance and angle in an intermediate fashion. Additionally, both positive and independent ar-

rays demonstrated a gradual increase in preference for shorter, straighter motions over time. However, as bees gained experience, their preference for short movements increased, but their preference for straight movements remained the same. Thus, bees prioritised shortness over straightness while designing their foraging pathways. This discovery refutes the long-held belief that bees reproduce according to the nearest neighbour principle.

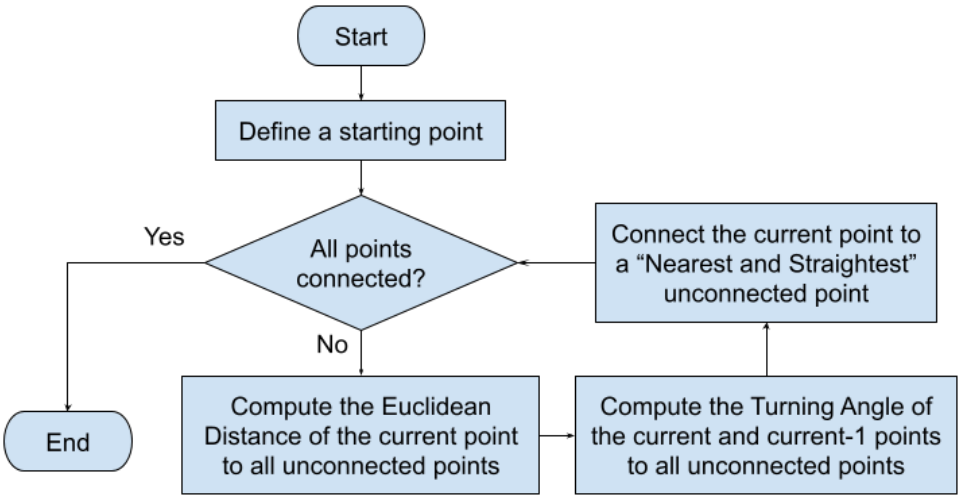


Figure 4.3: Constructive procedure with a distance and angle consideration, the Bee's Nearest Straight Neighbour (BNSN)

4.3 A Combinatorial Bees Algorithm with the bee's strategy for tour construction

This section will discuss the proposed initialisation technique, BNSN, and how it integrates with CBA. CBA will use it to generate the initial solution for TSP (Algorithm 12). The TSP is a problem that researchers identified as a model of bee traplining (Ohashi and Thomson, 2012; Lihoreau *et al.*, 2012b; Woodgate *et al.*, 2017). The problem is composed of k flowers, and the bee must visit each one only once in order to determine which sequence has the shortest total distance. The solution to the problem is a series of all flowers visited throughout the Hamilton cycle tour. The goal function is the total distance of the representative solution.

The suggested CBA begins with randomly generated initial solutions (tour) for scout bees using BNSN. The solutions will be sorted, and m of them will be selected as potential areas

of exploitation. The elite bees (nep) and non-elite bees (nsp) will gradually improve these m chosen sites solutions through local search operators. It was repeated several times prior to the termination conditions being met. Algorithm 12 contains the pseudocode for CBA+BNSN.

Algorithm 12: Combinatorial Bees Algorithm with BNSN

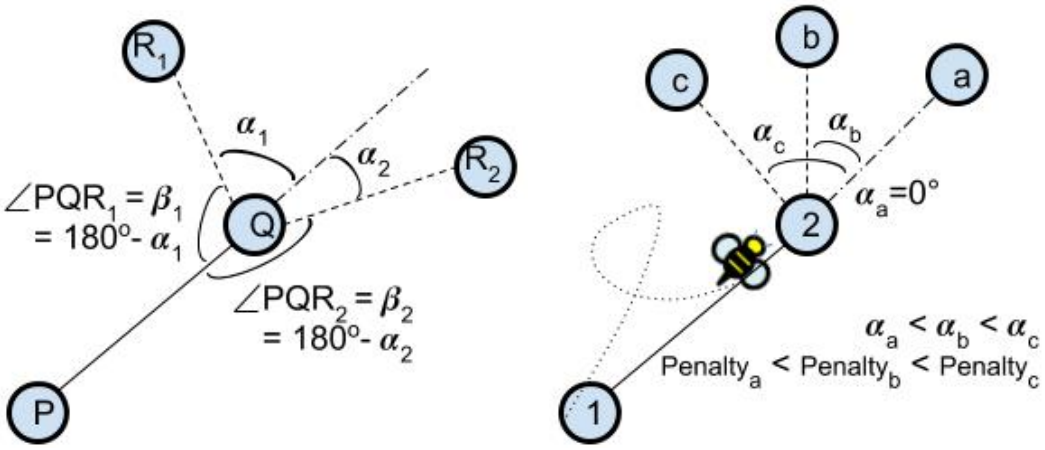
input : n = number of scout bees; e = elite sites; m = selected sites; nep = worker bees on the e sites; nsp = worker bees on the $m - e$ sites; FL = the maximum number of flowers in bee's vision; Δr= the bee's range vision.
output: *Bees*

```
Start;
Initialise n scout bees population using BNSN (FL, Δr);
Evaluate the fittness of the sites;
while termination criterion not satisfied do
    Select m sites for neighbourhood search;
    Recruit worker bees (nep and nsp) for selected sites, more bees on e sites;
    Exploit all selected sites inside ngh range and evaluate fitnesses;
    Select the fittest bee from each patch;
    Assign remaining bees to explore randomly and evaluate their fitnesses;
end
Report the best Bees;
End;
```

In the BNSN, the bee starts by visiting a random flower (as there is no hive location) and then regularly visits the closest straight flower before all flowers have been visited. The bee could decide the closest flower from its starting position. Still, the turning angle could not be calculated until the two flowers had been visited and the bee's direction determined. The third flower is the first one whose angle can be calculated, and it is selected depending on its distance to the bee and angle relative to the bee's location. The bee will repeat the visit until no flowers remain. The pseudocode can be found in Algorithm 14.

$$\cos\beta = \frac{\vec{PQ}.\vec{QR}}{\left\|\vec{PQ}\right\| . \left\|\vec{QR}\right\|} \tag{4.1}$$

The fundamental guiding concept in selecting the BNSN is to minimise the total cost of distance and angle penalty. The distance is treated as a primary constraint, while the turning angle is treated as a secondary constrain. The length is calculated using Euclidean distance, while the angle is calculated using the cosine law. This role is shown in Figure 4.4a using a



(a) Angle (b) Penalty

Figure 4.4: Measuring the bees turning angle and penalty

plain array of three flowers. The values of $\beta 1$ and $\beta 2$ provided by three flowers ($P-Q-R1$ and $P-Q-R2$) could be calculated using Equation 4.1. The angle β is diametrically opposite to the angle α, as shown in the same figure. It demonstrates that the following straight flower could have either an $\alpha = 0^o$ or a $\beta = 180^o$ without incurring any penalty. The more intense the shape of the turning angle, the larger the penalty cost (see Figure 4.4b). Based on an assumption, a penalty factor is determined. If the range of α angles is $0^o - 180^o$ or the range of β angles is $180^o - 0^o$, the penalty factor is $0\% - 100\%$ (Equation 4.2 and 4.3). The cost of the next flower with a 1-meter distance and 90^o movement is equal to a 1.5-meter straight movement flower. The pseudocode for calculating the penalty based on the turning angle is shown in Algorithm 13.

$$penaltyfactor = 1 - (\frac{\beta}{180^\circ}) \tag{4.2}$$

$$penaltyfactor = 1 - (\frac{180^\circ - \alpha}{180^\circ}) \tag{4.3}$$

This approach allows the assumption that several surrounding flowers are visible to the bees. Determining the vision range without considering the nearest flower opens the possibility of the nearest vertex being unidentified. These conditions must be met for the heuristic to work correctly. It demonstrates that if the distance between the second nearest vertex and the bee's vision is more extended than ($d_{2^{th}nearest} - d_{1^{th}nearest} > \Delta r * d_{1^{th}nearest}$), the bee would consider

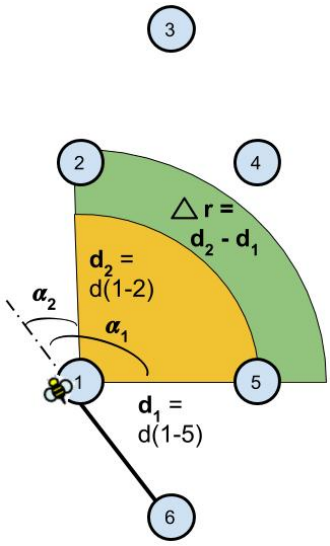


Figure 4.5: Bee thinks distance and angle

only the closest vertex. If the second, third, or FL^{th} flower is within her vision, the bee will identify them. The bee's next visitation will be to review flowers 2 and 5 (see Figure 4.5). By setting these FL and Δr values, we can control the diversity of BNSN solutions. As shown in the preceding section, this proposed method will behave identically to the NNH if FL is set to one.

4.4 Experiments, results and discussion

In this chapter, we performed two sequential investigations. The first is to determine the degree of diversity among the constructive heuristic methods as the initial solution generator. Then comes the primary investigation, which compares the accuracy of all CBA versions to those constructed heuristic methods used as the initial solution generator. The study will compare three different initial solution generator, namely the random number generator (RNG), NNH, and BNSN. The comparison study selects NNH as one of the CBA variants because it is assumed to be the bees' fundamental routing instinct and is widely regarded as the most effective constructive heuristic for resolving TSP (Dorigo and Stützle, 2019). Furthermore, both benchmark methods utilised the CBA with the same neighbour search mechanism (combination of a swap, insert, and reverse).

Algorithm 13: Measure the distance and angle

Def `Distance(`x, y`)`**:**
  for *all unvisited vertices* **do**
    $d_{(i,j)} \leftarrow \sqrt{(x_i - x_j)^2 - (y_i - y_j)^2}$; `// j = unvisited vertices connecting the current visited vertex (i)`
  end
  $d \leftarrow \min d(:)$;
  return d;
Def `Angle(`x, y`)`**:**
  $P \leftarrow [x_{(i-1)}, y_{(i-1)}]$; `// previous visited vertex`
  $Q \leftarrow [x_i, y_i]$; `// current visited vertex`
  for $fl = 1$ *to* FL **do**
    $R(fl) \leftarrow [x_{Nearest(f)}, y_{Nearest(fl)}]$; `//` $fl = 1, \ldots, FL$
    $\beta(fl) \leftarrow \arccos \frac{\vec{PQ}.\vec{QR(fl)}}{\left\|\vec{PQ}\right\|.\left\|\vec{QR(fl)}\right\|}$;
  end
  $\beta \leftarrow \min \beta(:)$;
  return β;

4.4.1 Preliminary experiment

The proposed method would be evaluated in this preliminary experiment for its ability to generate a good solution and for its diversity in covering the solution space. The greater the diversity of the initial solution, the greater the chance of discovering new regions of the solution space that could contain near-optimal solutions (Lobo *et al.*, 2020). While the chance of encountering a better solution increases, the calculation becomes more costly. The proposed method was evaluated using a simple hexagonal array, the Oshashi's array, to verify and validate the algorithm's creation and to ensure that the algorithm functioned correctly in a simple case.

The Oshashi's array is a hexagonal array of six vertices, the sixth of which represents the beehive (see Figure 4.6a). This array has a different order between optimal (see Figure 4.6b) and NNH tour (see Figure 4.6c); later it called the negative array. Two initial parameters will be used to run the BNSN. FL is the maximum number of flowers within the bee's vision, and Δr is the bee's vision range gap (see Algorithm 12).

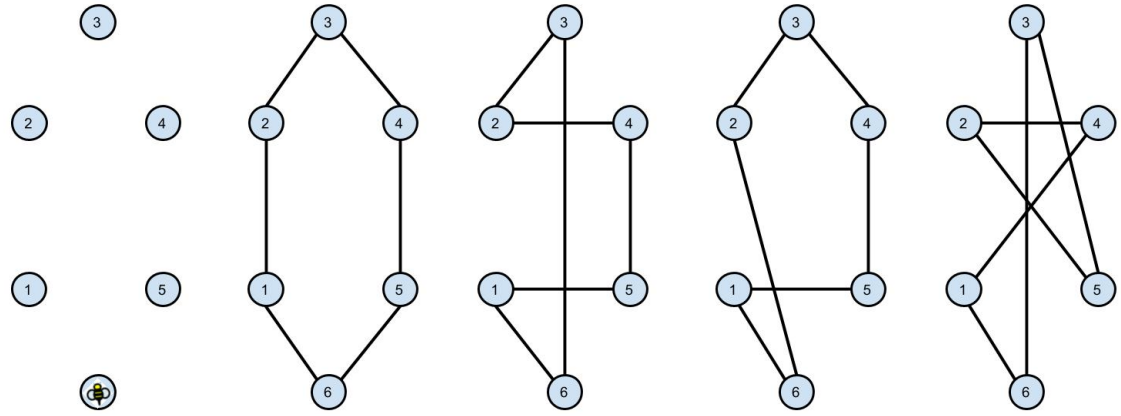
The result of the proposed method on this simple array (with FL=2, Δr=0.5) showed that it could generate more possible tours than NNH but less than RNG. If the hive (vertex 6) is the

Algorithm 14: Bee's Nearest and Straight Neighbour Heuristic

input : F = the maximum flowers in bee's vision; Δr= the bee's range vision.
output: T = a tour of k flowers

Start;
$T \leftarrow [\,]$;
T.append(s);
$Unvisited$.remove(s) ;
for 2^{th} *visits* **do**
    $d \leftarrow$ Min **Distance** (s, $Unvisited$, x, y);
    $Next \leftarrow d$.index(min(d));
    T.append($Next$);
    $Unvisited$.remove($Next$);
    $Current \leftarrow Next$;
end
for 3^{th} *to* k^{th} *visits* **do**
    $d \leftarrow$ Min **Distance** ($Current$, $Unvisited$, x, y);
    $\beta \leftarrow$ Min **Angle** ($Current$,$Nearest$,x_i, y_i);
    if $d(F) - d(1) \leq \Delta r * d(1)$ **then**
        for *fl* = 1 *to* FL **do**
            $Penalty(fl) \leftarrow (1 - (\beta(fl)/180°))$;
            $Cost$.append$(d(f) * (1 + Penalty(fl)))$;
        end
        $Next \leftarrow Nearest$ ($Cost$.index(min($Cost$)));
    else
        $Next \leftarrow Nearest(1)$;
    end
    T.append($Next$);
    $Unvisited$.remove($Next$);
    $Current \leftarrow Next$
end
End

starting point, the RNG could generate $(n-1)!/2 = 60$ possible tours, the BNSN three tours (6-1-2-3-4-5; 6-1-5-4-3-2; 6-1-5-4-2-3), and NNH only one tour (6-1-5-4-2-3). It implies that BNSN has a bigger chance to meet the optimal solution (more divers) than the NNH but not to RNG. Although the RNG has the best chance of finding the optimal solution, it is impractical due to RNG's initial solution having poor quality (accuracy). Using this simple array, we found that the solutions of BNSN with $FL = 1$ always had identical to NNH's solutions.



(a) Hexagonal array (b) Optimal tour (c) NNH tour (d) Other tour (e) Random tour

Figure 4.6: Circuits of the difference constructive heuristics and random number generator (RNG)

After this simple examination, the BNSN tested using 50-200 TSPLIB's instance. The BNSN set to 100 running experiments for each instance. This experiment was designed to identify how good constructive heuristics serve the local search operator. The average error toward BKS (Equation (4.4)) and the percentage of the unique sequences of scout bees population ($u\%$) was used to measure it. Table 4.1 presents the minimum, maximum, and average value of total tour length in column 2,3,4. The percentage of unique (u) tour sequence represents the diversity in column 5 for each heuristic method. All of them were tested ten times of 100 run experiments to conduct statistical test (all the data is available in the appendix section). Equation (4.4) below shows the average error rate calculation toward BKS refer to the quality measurement in Ch.2.

$$Avg\ Err = \frac{(Avg - BKS)}{BKS}.100\% \tag{4.4}$$

Table 4.1: The Initial solution of Random Number Generator (RNG), BNSN-1 (FL=1; Δr=[0,1]), BNSN-2 (FL=2; Δr=[0,1]), and BNSN-3 (FL=3; Δr=[0,1])

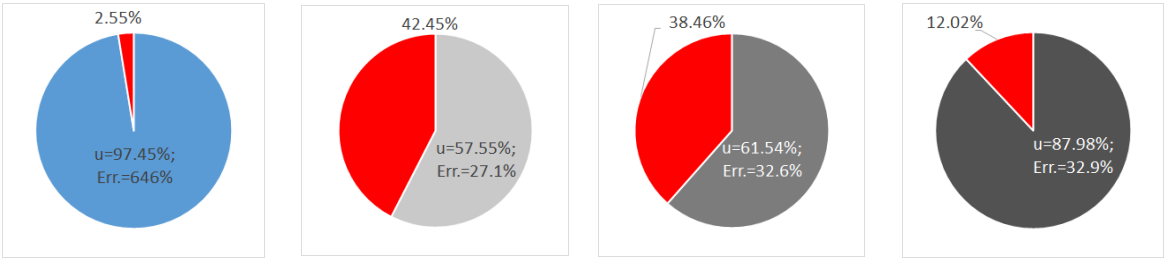
	RNG			BNSN-1			BNSN-2			BNSN-3		
No.	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
F-1	1,401	1,914	1,652	482	563	526	495	640	563	483	659	566
F-2	25,420	33,447	29,937	8,181	10,296	9,339	8,672	10,890	9,432	8,300	11,269	9,642
F-3	3,196	4,066	3,657	796	906	842	790	967	868	775	986	867
F-4	2,211	2,791	2,526	608	710	666	619	763	687	610	788	691
F-5	514,234	642,082	577,298	131,058	157,058	147,259	130,514	162,875	146,884	130,598	164,415	147,914
F-6	7,375	9,424	8,410	1,447	1,630	1,537	1,459	1,784	1,613	1,448	1,845	1,612
F-7	151,250	191,860	171,402	24,842	28,616	27,063	25,373	33,410	28,949	25,409	33,479	28,803
F-8	146,531	188,752	169,051	25,884	30,190	27,836	26,406	33,439	30,089	25,882	35,141	29,678
F-9	151,869	189,105	170,237	23,660	29,419	26,123	24,883	33,381	28,911	23,801	33,479	28,412
F-10	142,818	180,268	162,697	24,852	29,671	27,375	25,227	31,415	27,512	24,951	32,745	27,960
F-11	153,557	193,412	172,935	24,699	30,543	27,434	25,625	32,951	28,910	24,978	34,160	28,990
F-12	231,246	282,583	257,126	31,587	35,888	33,718	31,979	39,720	35,469	32,263	40,534	35,515
F-13	230,263	284,630	256,889	31,626	38,661	35,359	31,936	39,808	**35,060**	31,519	40,725	36,070
F-14	305,840	366,885	340,252	34,650	41,812	37,559	35,792	44,424	40,347	34,935	44,683	39,616
F-15	303,346	359,888	332,609	35,486	38,749	37,160	35,613	42,669	38,641	35,575	44,753	38,834

Table 4.2: The Initial solution's diversity level of RNG, BNSN-1 (FL=1; Δr=[0,1]), BNSN-2 (FL=2; Δr=[0,1]), and BNSN-3 (FL=3; Δr=[0,1])

No	datasets	RNG	BNSN-1	BNSN-2	BNSN-3
F-1	Eil51	87.5	27.5	38.6	68.1
F-2	Berlin52	98.8	42.1	62.5	90.3
F-3	St70	92.2	37.5	46.2	70.2
F-4	Eil76	87.3	46.3	43.5	66.6
F-5	Pr76	100	56.5	67.6	95.1
F-6	Rat99	97.8	52	56.6	82.4
F-7	KroA100	99.4	59.6	66.2	93.5
F-8	KroB100	100	63	70.3	94.2
F-9	KroC100	99.4	59	68.2	93.1
F-10	KroD100	100	59.9	66.1	92.5
F-11	KroE100	99.4	60.7	69.1	94.4
F-12	KroA150	100	71.8	68	92.9
F-13	KroB150	100	69.8	61.6	94.5
F-14	KroA200	100	77.2	71	96.8
F-15	KroB200	100	80.4	70.9	95.1
	average	**97.45**	**57.55**	**61.76**	**87.98**

Table 4.1 shows the initial solutions quality of all benchmark initial solution generator for all instances. It indicates that BNSN(1) supplies the CBA with the better initial solution or less error (compare to BKS) for all instances except in F-13 instances. However, all of BNSN solutions have no significant difference, they all have similar error performance. Table 4.2 shows the percentage of unique solution of initial solutions of all benchmark initial solution generator for all instances. BNSN(FL=2,Δr=[0,1]), called BNSN(2), supplies the operators with more divers than NNH's initial solutions for the majority of instances, but it is not a significant difference. For Eil51, the RNG could provide 87.5 per cent unique solutions, which means that around twelve per cent of them were identical to the others. The BNSN(2) and NNH initial solution quality and diversity were significantly different from the RNG. It could imply that BNSN(2) and NNH have a similar quality to serve CBA neighbour search. So, the new setting of BNSN was re-explored to find a significant difference in diversity between NNH and BNSN. Later called BNSN(3), this new setting would be F equal to a random integer of 3, and Δr equal to a random number of 0-100%. Table 4.1 presents the comparison of RNG, NNH or BNSN(1), BNSN(2), and BNSN(3). Table 4.2 showed the diversity level of BNSN(3) and the significant differences compared to the NNH. The statistical test using Kruskal Wallis ANOVA could be

seen in Figure 4.8.



(a) RNG (b) BNSN(1) (c) BNSN(2) (d) BNSN(3)

Figure 4.7: The diversity of the constructive heuristics and RNG

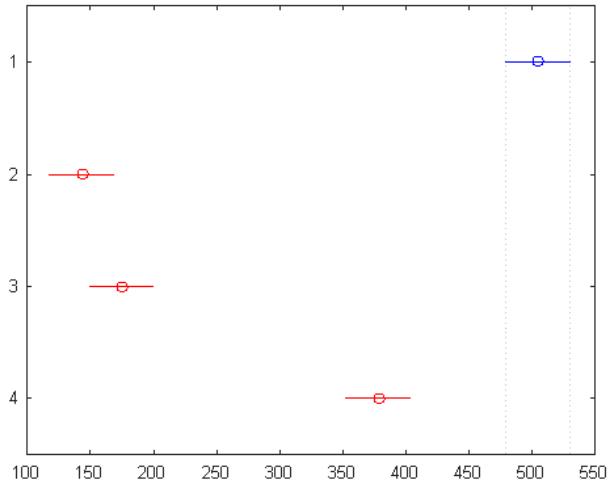


Figure 4.8: Kruskal Wallis ANOVA test on the heuristic's diversity (1=RNG; 2=BNSN(1); 3=BNSN(2); 4=BNSN(3))

Figure 4.7 presented the error and diversity level of the benchmark initial solution generators. It indicates that the RNG has a deficient performance by reaching up to 646% average error rate for all instances and has the diversity level performance of 97.45% unique solutions. On the other side, BSNS(3) has the best overall performance as an initialisation solution generator based on the balance of the error rate and diversity level with 32.9% error and 87.98% unique solutions. The non-parametric ANOVA test (see Figure 4.8) shows the diversity difference of all methods. Except for NNH and BNSN(2), all the methods have a significant difference.

4.4.2 Main experiment

The entire CBA+BNSN is evaluated in this section using 50-200 cities instances from TSPLIB. We used 15 datasets to compare the proposed version of CBA to basic CBA (with RNG) and CBA+NNH. Table 4.3 contains the setting parameters, and each dataset was tested in ten independent experiments. The setting parameter was following the previous study that found that the adequate colony size is equal to ten times its dimension.

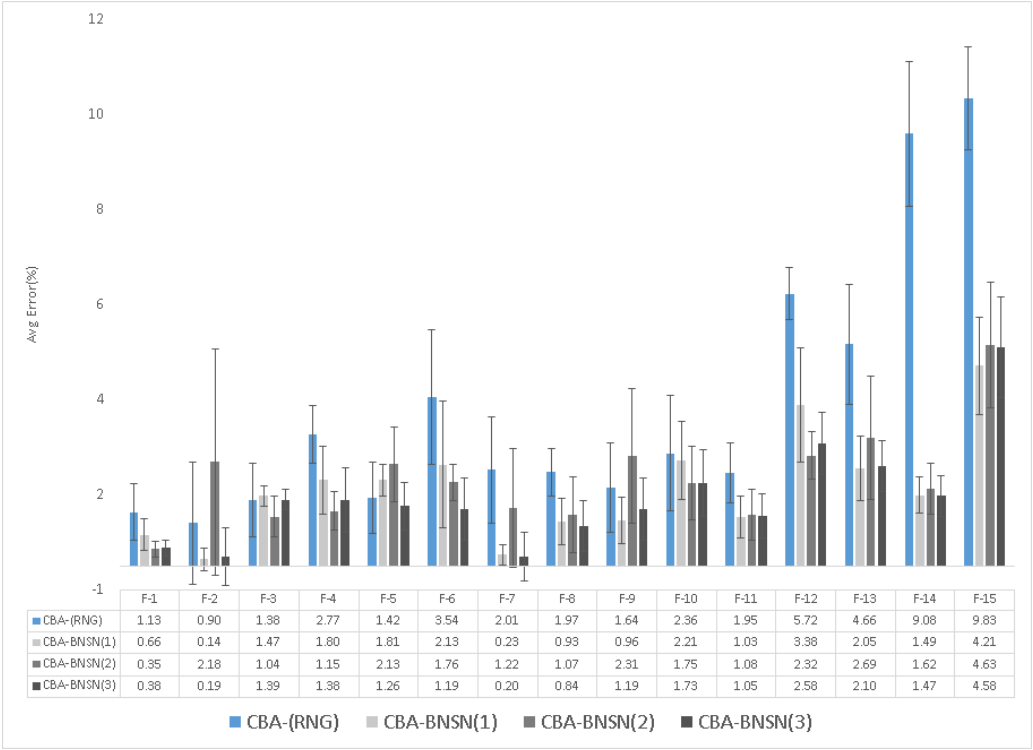
Table 4.3: Parameters setting of CBA+BNSN

Parameters	values
Colony Size	10*$Dims$
Max Iterations	3,000
Number of scout bees (n)	$Dims$
Number of elite sites (e)	5
Number of selected sites (m)	14
Number of bees for elite sites (nep)	$Dims$
Number of bees for selected sites (nsp)	$0.5 * Dims$
Number of flowers in the bees' vision (FL)	[1,3]
Range of bee's vision extension (Δr)	[0,1]

Table 4.4 compares the results of three different versions of CBA. The table contains the best-known solution (BKS), the best calculation of all experiments ($Best$), the average result of all experiments (Avg), the standard deviation ($StdDev$), the error rate of the $Best$ and Avg results against BKS ($BErr$, $AErr$), and the standard deviation of the error results ($SErr$).

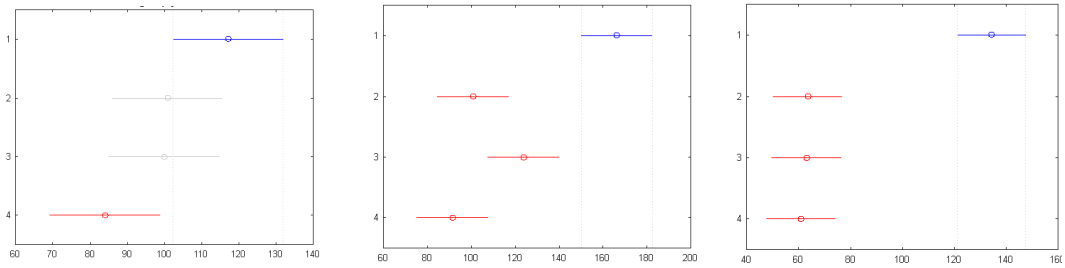
Table 4.4 indicates that there is no single version that dominated other methods for all datasets. CBA+BNSN(3), CBA+BNSN(2), CBA+NNH, and Basic CBA have 8, 8, 6, 3 minimum $Best$ result out of 15 (bold font). While the average CBA+BNSN(2) has six minimum Avg results, CBA+BNSN(3) five, CBA+NNH two, and Basic CBA zero.

The bar chart in Figure 4.9 represents the algorithm's average error toward BKS. The figure also shows there were no significant differences between all CBA versions with a constructive method (CBA+NNH, CBA+BNSN(2), CBA+BNSN(3)) -later called CBA+heuristic- for all instances. And there are no significant differences between Basic CBA and CBA+heuristic version for all 50-100 instances except Eil76. For 150-200 instance, there were significant differences between them. The easiest instance was the Berlin52 which all methods could reach



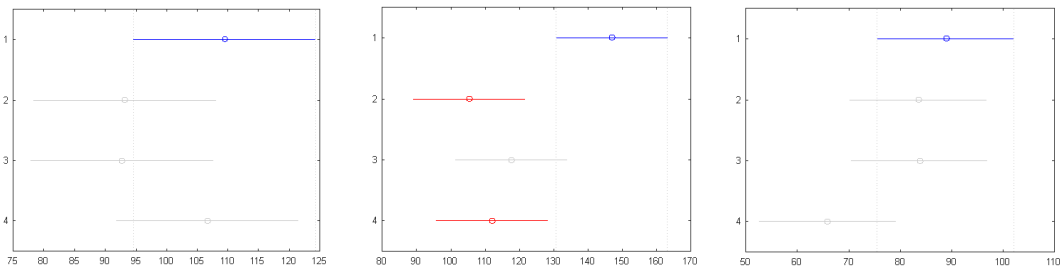
	F-1	F-2	F-3	F-4	F-5	F-6	F-7	F-8	F-9	F-10	F-11	F-12	F-13	F-14	F-15
CBA-(RNG)	1.13	0.90	1.38	2.77	1.42	3.54	2.01	1.97	1.64	2.36	1.95	5.72	4.66	9.08	9.83
CBA-BNSN(1)	0.66	0.14	1.47	1.80	1.81	2.13	0.23	0.93	0.96	2.21	1.03	3.38	2.05	1.49	4.21
CBA-BNSN(2)	0.35	2.18	1.04	1.15	2.13	1.76	1.22	1.07	2.31	1.75	1.08	2.32	2.69	1.62	4.63
CBA-BNSN(3)	0.38	0.19	1.39	1.38	1.26	1.19	0.20	0.84	1.19	1.73	1.05	2.58	2.10	1.47	4.58

Figure 4.9: Accuracy performance of all the Combinatorial Bees Algorithm versions



(a) [50,76] (b) [99,100] (c) [150,200]

Figure 4.10: Kruskal Wallis ANOVA test on the accuracy of all versions of CBA for 15 datasets (1=RNG; 2=BNSN(1); 3=BNSN(2); 4=BNSN(3))



(a) [50,76] (b) [99,100] (c) [150,200]

Figure 4.11: Kruskal Wallis ANOVA test on the NFE of all versions of CBA for 15 datasets (1=RNG; 2=BNSN(1); 3=BNSN(2); 4=BNSN(3))

Table 4.4: The comparison of 3 scenarios of CBA+BNSN

		CBA basic (RNG)			CBA BNSN(1)			CBA BNSN(2)			CBA BNSN(3)		
No	BKS	Best (BErr-%)	Avg (AErr-%)	Std (SErr-%)	Best (BErr-%)	Avg (AErr-%)	Std (SErr-%)	Best (BErr-%)	Avg (AErr-%)	Std (SErr-%)	Best (BErr-%)	Avg (AErr-%)	Std (SErr-%)
F-1	426	428.00	430.80	2.53	**427.00**	428.80	1.40	**427.00**	**427.50**	0.71	**427.00**	427.60	0.70
		(0.46)	(1.12)	(0.59)	**(0.23)**	(0.65)	(0.32)	**(0.23)**	**(0.35)**	(0.16)	**(0.23)**	(0.37)	(0.16)
F-2	7542	**7542.00**	7609.70	96.87	**7542.00**	**7552.60**	18.02	**7542.00**	7706.70	179.50	**7542.00**	7556.30	45.22
		(0)	(0.89)	(1.28)	**(0)**	**(0.14)**	(0.23)	**(0)**	(2.18)	(2.38)	**(0)**	(0.18)	(0.59)
F-3	675	**675.00**	684.30	5.21	684.00	684.90	1.37	**677.00**	**682.00**	2.94	682.00	684.40	1.51
		(0)	(1.37)	(0.77)	(1.33)	(1.46)	(0.20)	(0.29)	**(1.03)**	(0.43)	(1.03)	(1.39)	(0.22)
F-4	538	549.00	552.90	3.28	541.00	547.70	3.86	541.00	**544.20**	2.15	**540.00**	545.40	3.63
		(2.04)	(2.76)	(0.60)	(0.55)	(1.80)	(0.71)	(0.55)	**(1.15)**	(0.39)	**(0.37)**	(1.37)	(0.67)
F-5	108159	**108234.00**	109696.50	809.06	109653.00	110115.00	360.25	109523.00	110466.30	838.74	**108234.00**	**109524.30**	539.02
		(0.06)	(1.42)	(0.74)	(1.38)	(1.80)	(0.33)	(1.26)	(2.13)	(0.77)	**(0.06)**	**(1.26)**	(0.49)
F-6	1211	1227.00	1253.90	17.08	**1212.00**	1236.80	16.05	1222.00	1232.30	4.64	1213.00	**1225.40**	7.85
		(1.32)	(3.54)	(1.41)	**(0.08)**	(2.13)	(1.32)	(0.90)	(1.75)	(0.38)	(0.16)	**(1.18)**	(0.64)
F-7	21282	21315.00	21710.50	238.09	**21282.00**	21330.90	45.63	**21282.00**	21542.60	265.92	**21282.00**	**21324.20**	110.18
		(0.15)	(2.01)	(1.11)	**(0)**	(0.22)	(0.21)	**(0)**	(1.22)	(1.24)	**(0)**	**(0.19)**	(0.51)
F-8	22141	22334.00	22577.60	109.94	**22179.00**	22347.60	109.06	**22179.00**	22378.80	177.64	**22179.00**	**22327.80**	118.31
		(0.87)	(1.97)	(0.49)	**(0.17)**	(0.93)	(0.49)	**(0.17)**	(1.07)	(0.80)	**(0.17)**	**(0.84)**	(0.53)
F-9	20749	20892.00	21090.20	194.39	20785.00	**20947.80**	102.22	20852.00	21228.50	293.05	**20749.00**	20996.60	134.21
		(0.68)	(1.64)	(0.93)	(0.17)	**(0.95)**	(0.49)	(0.49)	(2.31)	(1.41)	**(0)**	(1.19)	(0.64)
F-10	21294	21567.00	21894.30	258.87	21704.00	21862.30	175.99	**21501.00**	21763.10	164.85	21514.00	**21758.90**	149.91
		(1.28)	(2.82)	(1.22)	(1.93)	(2.67)	(0.83)	**(0.97)**	(2.20)	(0.77)	(1.03)	**(2.18)**	(0.70)
F-11	22068	22344.00	22498.40	138.84	22116.00	**22295.00**	97.69	**22107.00**	22306.50	118.88	**22107.00**	22299.00	102.71
		(1.25)	(1.95)	(0.62)	(0.21)	**(1.02)**	(0.44)	**(0.17)**	(1.08)	(0.53)	**(0.17)**	(1.04)	(0.46)
F-12	26524	27792.00	28041.70	146.88	**26916.00**	27420.80	315.58	26954.00	**27139.80**	132.29	26965.00	27208.10	173.19
		(4.78)	(5.72)	(0.55)	**(1.47)**	(3.38)	(1.18)	(1.62)	**(2.32)**	(0.49)	(1.66)	(2.57)	(0.65)
F-13	26130	26642.00	27347.40	327.82	26394.00	**26665.40**	174.23	**26381.00**	26833.30	339.58	26382.00	26678.80	139.42
		(1.95)	(4.65)	(1.25)	(1.01)	**(2.04)**	(0.66)	**(0.96)**	(2.69)	(1.29)	**(0.96)**	(2.10)	(0.53)
F-14	29368	31546.00	32033.80	446.23	29656.00	29805.10	114.63	**29623.00**	29843.80	160.13	29654.00	**29800.20**	126.10
		(7.41)	(9.07)	(1.51)	(0.98)	(1.48)	(0.39)	**(0.86)**	(1.62)	(0.54)	(0.97)	**(1.47)**	(0.42)
F-15	29437	31769.00	32329.40	315.70	30321.00	**30675.60**	301.49	**30205.00**	30800.80	388.22	30245.00	30786.40	311.17
		(7.92)	(9.82)	(1.07)	(3.00)	**(4.20)**	(1.02)	**(2.60)**	(4.63)	(1.31)	(2.74)	(4.58)	(1.05)

BKS with very small deviations. The KroA100 was the easiest instance for CBA+heuristic methods.

4.4.3 Discussion

As shown in the preliminary experiment, the BNSN has the potential to raise the degree of diversity by introducing another factor (turning angle). The disadvantage of NNH is that it usually generates a minimal number of unique candidate solutions (Hoos and Stützle, 2004), as the randomise greedy construction successfully addressed by treating angle as a secondary constraint and distance as a primary constraint. The BNSN parameters successfully control the initial solution's diversity. By limiting it to a single flower within the bees' view, the diversity degree of BNSN is equivalent to that of NNH's.

On the larger dimension problem, the output of CBA+BNSN(2 and 3) that represent more divers on their initial solution is significantly different from that of CBA+NNH or CBA+BNSN(1). It implies that with 3,000 iterations, the low diversity degree (NNH's) is effective enough to achieve a near-optimal solution up to 100 dimensions.

Additionally, we discovered that a low error rate initial solution produced by a constructive heuristic does not guarantee a stronger final solution generated by a metaheuristic. The NNH (see Table 4.1) dominated the initial solutions with the lowest error rate, but this dominance vanished on the final solution. Another CBA+BNSN has a stronger final solution than the CBA+NNH even though the original solution has a higher error rate, as long as the diversity level is higher. The usage of a constructive heuristic in the initialisation process for small instances is pointless since it has slight distinguishable advantage over RNG. However, the effect is critical in a more complex instance.

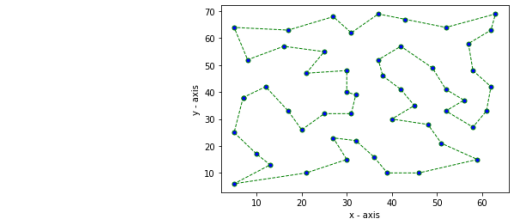
As a result, we can deduce that the CBA needs a constructive heuristic only when it deals with more than 150 cities. Below 150 dimensions, no significant difference in precision was observed between the benchmark methods (see Figure 4.9 and 4.10). However, the constructive heuristic approaches can provide substantial improvement in convergence for more than 99 dimensions. CBAs using a constructive heuristic approach will achieve the optimal solution in

fewer evaluations than RNGs. For 50-76 datasets, the Kruskal Wallis ANOVA test revealed no discrepancy in the convergence between all methods (see Figure 4.11). Convergence test details are accessible in the appendix section.

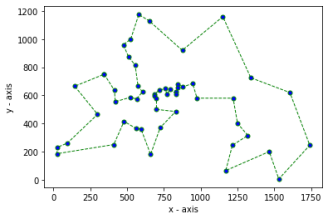
4.5 Summary

This chapter proposed a novel constructive heuristic for CBA named Bee's Nearest Straight Neighbour (BNSN). It is introduced to enrich the Bees Algorithm's combinatorial variant. The experiment findings indicate that BNSN is an impressive constructive heuristic since it could control the balance of error rate and diversity with its parameters. The BNSN could assist CBA in diversifying the initial solution by recognising up to 88 per cent of scout bees as unique feasible candidate solutions with an average error to the optimal solution of roughly 30%.

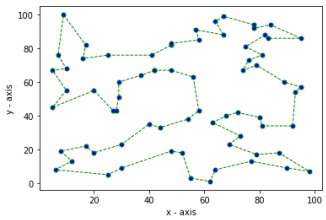
Additionally, the CBA+BNSN shows statistically significant higher accuracy for larger instances (99-200 dimension) when compared to the basic CBA. It is possible to conclude that by using the proposed constructive heuristic, the overall error rate of the basic CBA might be reduced by 1.82 per cent.



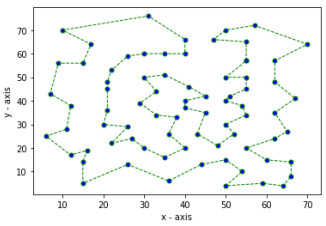
(a) Eil51(427)



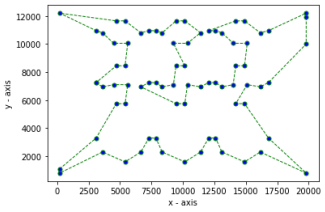
(b) Berlin52(7,542)



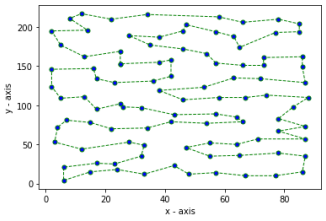
(c) St70(677)



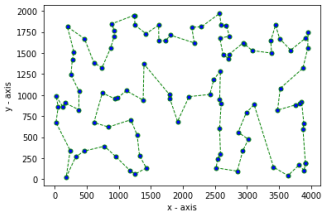
(d) Eil76(540)



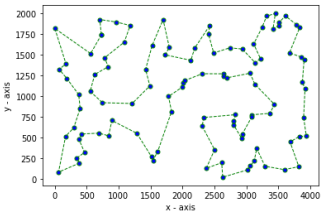
(e) Pr76(108,234)



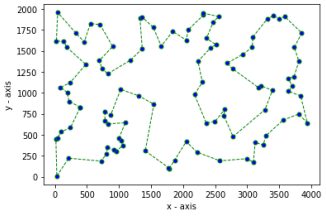
(f) Rat99(1,212)



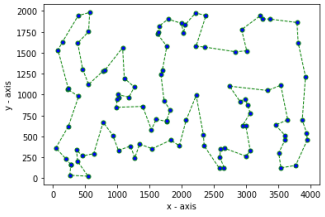
(g) KroA100(21,282)



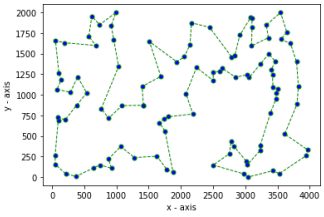
(h) KroB100(22,179)



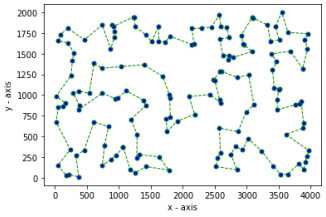
(i) KroC100(20,749)



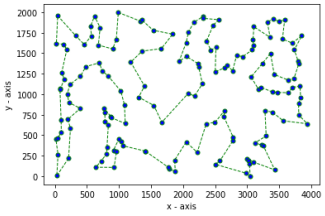
(j) KroD100(21,501)



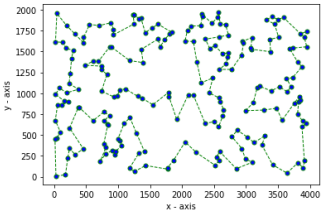
(k) KroE100(22,107)



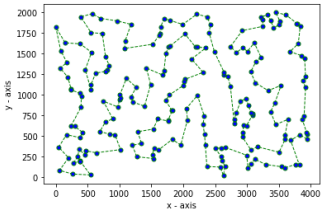
(l) KroA150(26,916)



(m) KroB150(26,381)



(n) KroA200(29,623)



(o) KroB200(30,205)

Figure 4.12: The best solution tours of CBA+BNSN for [51,200] TSPs' instances

Chapter 5

Novel exploitation strategy of Combinatorial Bees Algorithm

5.1 Preliminaries

This chapter focuses on the ability to effectively reach a high-quality candidate solutions when a good starting point is provided. The development of CBA at the exploitation phase is discussed in order to intensify the solution. The novel neighbour search mechanism, called Bees Routing Optimiser (BRO), proposed here was inspired by bees' natural traplining foraging behaviour. The proposed method is motivated by the way bees optimise their path by preventing repeated visits to such flowers in the following bout. The Appendix G contains the matlab code for this work.

As previously mentioned, the success of a nature-inspired algorithm can be dependent upon the initial solution or starting point (Yang, 2020). However, constructive heuristic algorithms often provide imprecise solutions (Dorigo and Stützle, 2019). Later on, this imprecise technique can be enhanced with the aid of a local search algorithm or a neighbour search mechanism. The Lin-Kernighan method is the most well-known success story of local search resolving TSP (Lin and Kernighan, 1973).

According to Hoos and Stützle (2004), various scholars -including Brady in 1985; Suh

and Gutch in 1987; Mühlenbein et al., 1988; Ulder et al., 1991; Merz and Freisleben, 1997- discovered that implementing a local search or intensification after applying mutation and re-combination dramatically improves the efficiency of metaheuristic algorithms for combinatorial problems.

By proposing a new local search operator for TSP, this thesis attempted to enhance the combinatorial variant of CBA. The rationale for suggesting this mechanism is that the local search operator substantially impacts an algorithm's performance (Karaboga and Gorkemli, 2019). Another explanation for conducting this analysis is to address the lack of bee behaviour metaphors in the previous CBA studies.

This novel intensification technique, BRO, is influenced by how bees optimise their routing by avoiding threats. The intensifier employs two primary procedures: avoiding and re-visit phases, which were influenced by bees' trap-lining actions. This behaviour was first noticed by a researcher a few years back by Ohashi and Thomson (2012), Lihoreau *et al.* (2012b), and Woodgate *et al.* (2017). The following section will discuss bee traplining foraging behaviour. The result section will show the proposed algorithm's output findings and comparison analysis.

5.2 Bees' behaviour of avoiding threats in nature

Bees form a circuit to visit many flowers and create an effective tour that minimises travel costs between those flowers analogous to the TSP. (Ohashi and Thomson, 2012), Lihoreau *et al.* (2012b), Woodgate *et al.* (2017) conducted experiments on this routing behaviour and examined how bees build and optimise circuits on large spatial scales using cutting-edge technologies. The map illustrates how some segments of the route developed into habits whilst others were abandoned once the preferred flight path was discovered. It demonstrates how bees establish stable routes between flowers through a process that combines observation, learning, and experience. Bees rely on their vision and scent in addition to their tiny brain to memorise a spatial memory of their position, reward value, threats, and flower series Ohashi and Thomson (2012). Lihoreau *et al.* (2012b) determined the similarity of bee foraging visits to the optimal sequence, which improves dramatically with time and practise (experience).

All researchers mentioned above also agree that bees are sometimes avoiding to visit a flower for reasons. The primary reasons are to prevent an ineffective visit (source depletion) and to avoid threats (from the competitor, predator). Later, whether the conditions are stable or the supply is replenished, bees may return to these forgotten flowers.

Temporarily eliminating and reintroducing flowers becomes crucial during routing development because it enriches bees' experiences by providing a new information route (see Figure 5.1c). The bees learn more about the distance between two new flowers by navigating an incomplete new habitual route in their tour. They will then recalculate the new unique order based on their new knowledge. This knowledge will aid them in optimising the route in future tours.

For humans, this instinctive skill of bees is exceptional since they can not plan routes using spatial maps but instead gradually gather information about the position and the direction that links it. As a result of this simple behaviour of forgetting (avoiding) and reintroducing (revisiting) the flower(s), the authors tailored it to the local search mechanism in order to improve the performance of CBA. Furthermore, the existence of renewable honey and the competitive environment imposed on the algorithm by optimising incomplete sequence are more representative of real-world conditions.

5.3 Proposed exploitation search or intensification strategy

This section would discuss the suggested technique for exploiting the bees' traplining behaviour. According to Dorigo and Gambardella (1997), the improvement heuristic is most efficient when applied after the best solution is generated or is referred to as an intensifier. The BRO, as an intensifier, can improve the accuracy of CBA's local search operator's best solutions. The pseudo-code for Bees Routing Optimiser is seen in Algorithm 15 below.

To begin, the BRO will operate once it receives a danger signal. The threat is posed by a predator, a rival, or a resource depletion in the natural world. The danger signal in the proposed algorithm comes from the stationary value (the value does not improve from the previous iterations). The bees would reorganise the stagnant solution produced by a neighbour search into a sub-tour and forgotten flower (s). Bees, unlike humans, are unable to anticipate all possible

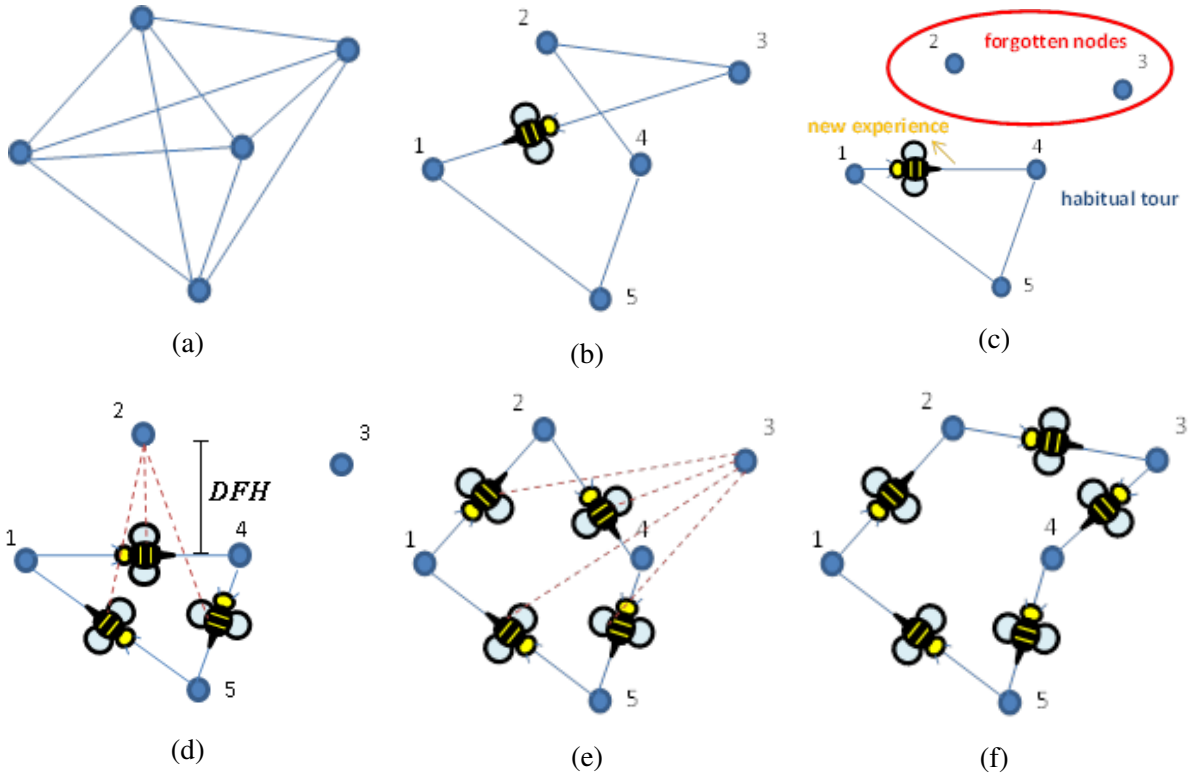


Figure 5.1: (a): The completed (paths) distance information; (b)-(f): the optimisation behaviour of bees

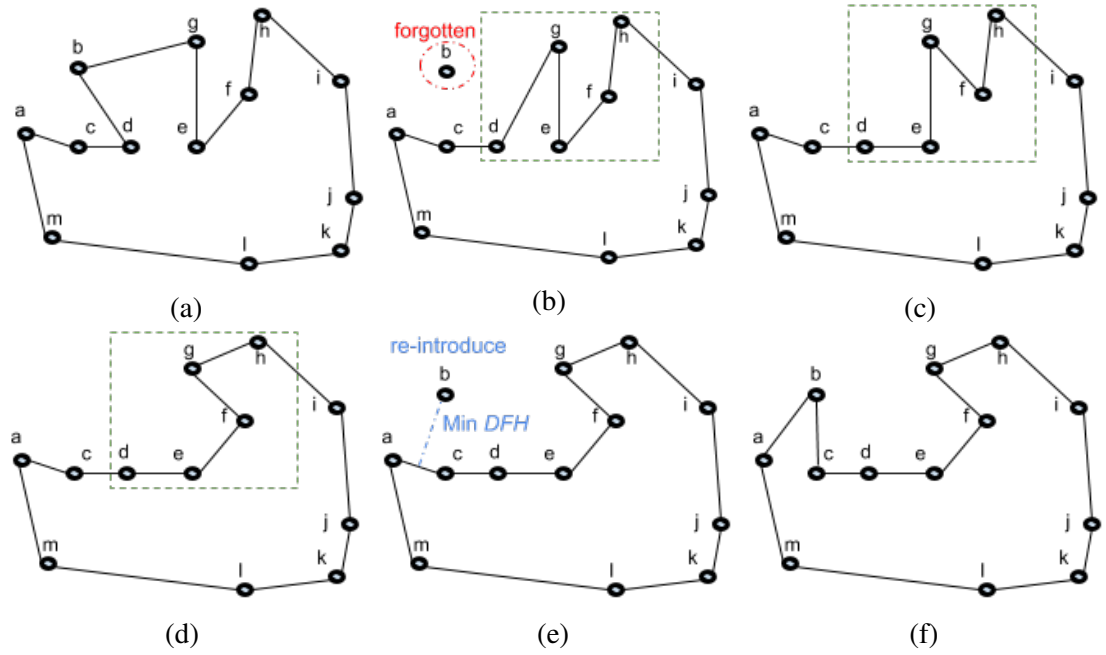


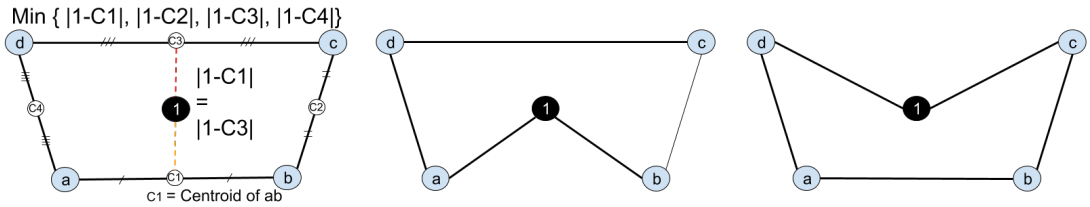
Figure 5.2: (a) Trap in local optima, (b) a forgotten node and sharp turning path inside the remaining nodes, (c) swap nodes e and g, (d) swap nodes f and g (e) a habitual tour and the re-introduction of the forgotten node, (f) inserting the forgotten node to the habitual tour.

paths (See Figure 5.1a). They focus on their prior experience collecting distance information by forgetting or dropping certain flowers (see Algorithm 15 line 4 and 10). They collect new route distance information while travelling on a new tour (See Figure 5.1b and 5.1c). Later, when the conditions are safe, the bees may return to the forgotten flower (see Figure 5.1d, 5.1e, 5.1f). Following an improved sub-tour on the sharp turn routes (see Algorithm 15 from line 14 to 20), the bees will be reintroduced to the forgotten flower(s) (fn) to reform the whole tour (see Algorithm 15 line 5 and 22).

Algorithm 15: Bees Routing Optimiser Heuristic

input : T_{init} = initial tour from previous process.
output: T = a tour of k cities.

1 **Start**;
2 **while** $T(it) >= T(it-1)$ *or has not reach maximum loop* **do**
3 **for** *for all bouts* **do**
4 $HT \leftarrow$ **Forgotten** (T, nfn); `// nfn: a number of forgotten flowers`
5 $T \leftarrow$ **Re-Introduction** (HT, fn); `// fn: the forgotten flowers`
6 **end**
7 $T(it) \leftarrow min[T_{(1)}, ..., T_{(bouts)}]$; `// it = iteration`
8 **end**
9 **End**;
10 **Def** `Forgotten(T, nfn):`
11 $fn \leftarrow$ random list of nfn forgotten flowers;
12 $T' \leftarrow$ a tour T subtract by fn;
13 $HT \leftarrow T'$;
14 **for** *all possible edge-pairs in* HT **do**
15 $HT* \leftarrow$ tour by reversing end points in edge-pair HT;
16 **if** $Cost(HT) \leq Cost(HT*)$ **then**
17 $HT \leftarrow HT*$;
18 **else**
19 **end**
20 **end**
21 return HT;
22 **Def** `Re-Introduction(HT, fn):`
23 $he \leftarrow$ list of all centroid of edges in HT as habitual tour;
24 **for** *each item* i *in* fn **do**
25 $DFH_{fn(i),he} \leftarrow$ list of distances from $fn(i)$ node to he;
26 $T \leftarrow$ tour by inserting a $fn(i)$ into the closest edge;
27 **end**
28 return T;



(a) (b) (c)

Figure 5.3: Insertion procedures (a) 2 minimum distance of forgotten nodes and habitual path (DFH) with long and short edges (b) inserting vertex-1 to edge ab generates optimal tour (c) inserting vertex-1 to edge cd generates near-optimal tour.

To build the complete tour, the bees will memorise the closest path to the forgotten flowers when travelling all the paths (he) on a habitual tour and insert the flowers in the centre of the closest paths on the subsequent trip (see Algorithm 15 line 26). The minimum distances (DFH) between the positions of all reintroduced flower (s) ($x(fn), y(fn)$) and the centroid of habitual tour paths (Equation (5.1)) was used to evaluate the nearest path (Equation (5.2)).

In Figure 5.2a, a basic array example of a trapping state is shown. Due to the absence of a sharp turn within the tour, this trapped array was extremely difficult for the CBA's local search operator to solve. In BRO, the bees can choose flowers to forget at random (Figure 5.2b). This method creates a sub-tour with sharp turns, and they then refine iteratively using their perception and tertial memory. They can enhance the sub-tour by avoiding a zigzag path, thus transforming it into a habitual tour (Figure 5.2c and 5.2d). The proposed approach employs the two-opt heuristic to remove the sub-tour's sharp turn direction. A two-opt heuristic can evaluate each possible swap reverse movement (Croes, 1958). The insertion of forgotten flowers will likely increase the tour's overall distance (Figure 5.2e, 5.2f)

$$C(he) = (Cx_{(he)}, Cy_{(he)}) = (\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}) \tag{5.1}$$

$$DFH_{(fn,C(he))} = |fn - C(he)| = \sqrt{(x_{(fn)} - Cx_{(he)})^2 - (y_{(fn)} - Cy_{(he)})^2} \tag{5.2}$$

Although this insertion does not guarantee a better solution for any movement, condition with two minimum DFH (see Figure 5.3), using 'break' on the maximum looping number

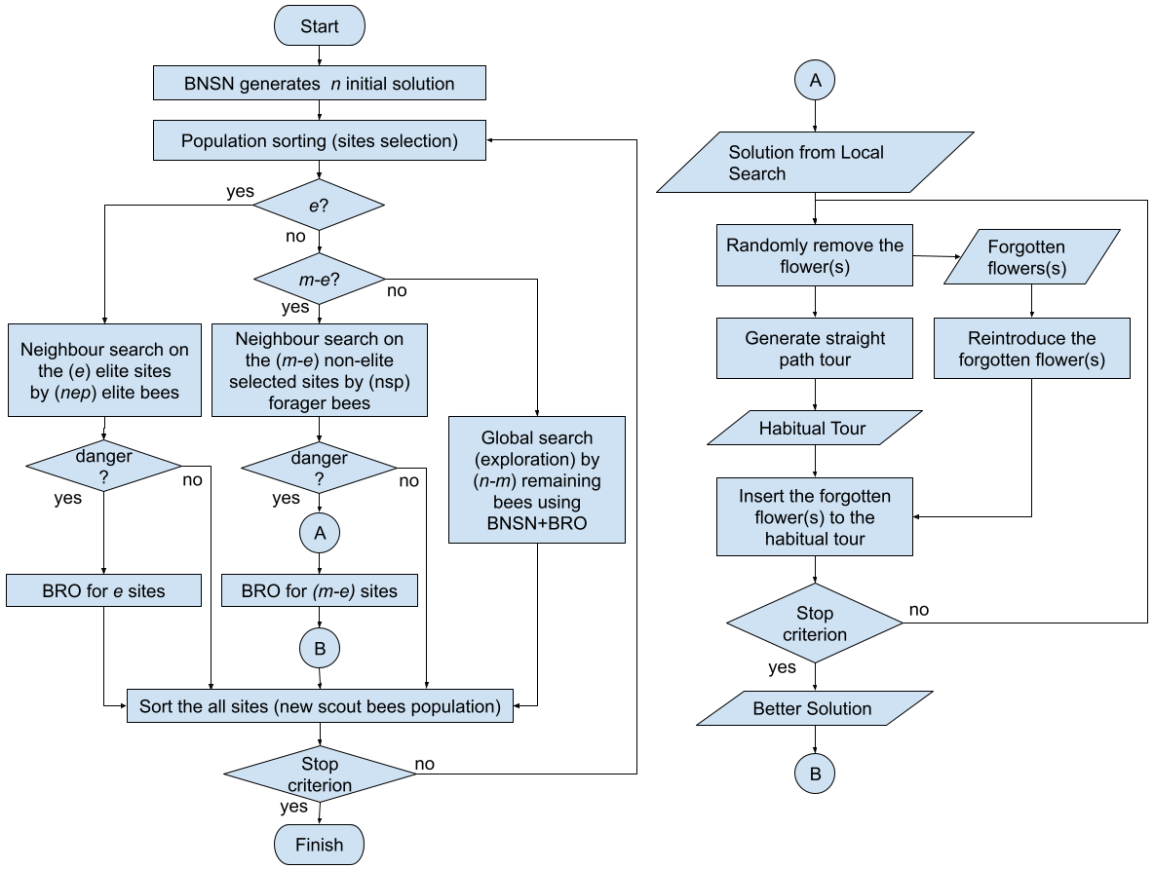


Figure 5.4: The flowchart of The Bees Algorithm with Bee's Nearest Straight Neighbour (BNSN) and Bees Routing Optimiser (BRO)

increases the algorithm's likelihood of escaping from local optima. The insertion will terminate until all forgotten nodes have been reintroduced into the habitual tour. On their next visit, the bees will have a new itinerary that includes all forgotten flower(s). The flowchart of CBA is depicted in Figure 5.4, along with the previously formed BNSN and BRO.

5.4 Experiments, results and discussion

This section consists of two investigations. The first investigation used CBA and BRO to resolve TSPLIB's datasets with [51,200] dimensions. The results would be compared to those obtained from the prior CBA with exploration strategy. The study uses the parameter settings mentioned in the preceding chapter as the most effective. The second investigation involved applying the CBA with both novel exploration and exploitation strategies to 15 TSPLIB datasets

Table 5.1: Parameter setting of the CBA+BNSN+BRO for comparative study

Parameter	CBA
Maximum evaluations	800,000
Colony size $(n + (e * nep) + ((m - e) * nsp))$	40
Population of scout bees (n)	10
Number of elite sites (e)	2
Number of selected sites (m)	5
Elite bees (nep)	9
Non-elite bees (nsp)	4
Number of flowers inside its vision (F)	[1,3]
The additional range of bees' vision (Δr)	[0,1]
The minimum number of forgotten flowers $(fn1)$	1
The minimum number of forgotten flowers $(fn2)$	25%*Dimension

with [51,1577] dimensions and performing comparison assessments between the proposed algorithm and comparable benchmark algorithms. The benchmark algorithms are mostly the bees inspired algorithms, including the artificial bee colony (ABC) and bee colony optimisation (BCO), and other nature-inspired algorithms, like the genetic algorithm (GA) and ant colony system (ACS). All the benchmark algorithms were utilising the nearest neighbour class constructive heuristic and the iterative improvement heuristic. Several datasets from the TSPLIB datasets with [52,1577] dimensions were selected (Reinelt, 1991, 1994) due to their frequent use in numerous TSP publications. The algorithm's performance was evaluated against two distinct metrics: percentage error relative to the best-known solution (Ismail *et al.*, 2020) and the number of function evaluations or tours required to obtain the best candidate solution (Osaba *et al.*, 2018). Several crucial parameters were chosen to be highly similar in order to ensure a fair comparison of CBA and benchmark algorithms. The parameters were established by earlier studies (Karaboga and Gorkemli, 2019; Wong *et al.*, 2010; Dorigo and Gambardella, 1997) as shown in Table 5.1.

5.4.1 Performance studies of exploration vs exploitation strategy

This section compares the results of CBA+BNSN and CBA+BRO. The comparison is made to ascertain the exploitation strategy's effectiveness in comparison to the exploration strategy. According to a previous analysis, the BA and CBA perform best when the colony size is ten

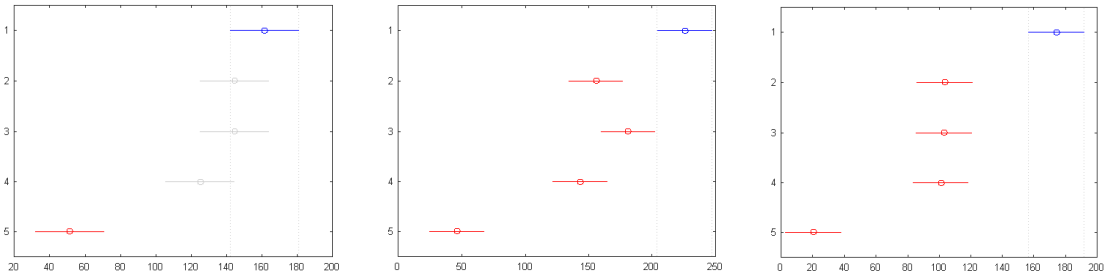
times the dimension of the problem. The CBA+BRO was configured according to Table 4.3, with the minimum and the maximum number of forgotten flowers equivalent to 1 and 25% of the problem's dimension, respectively (the BRO's parameters configuration). This experiment would terminate after 3,000 iterations (or NFE equal to 3,000 multiplied by the colony size). The experiment was repeated ten times, and then the statistical evaluation was performed after the accuracy and convergence results were obtained. The Kruskal Wallis ANOVA test was used because it compared more than two variants of CBA.

The accuracy and convergence of CBA+BRO with a colony size of the ten times problem dimension are presented in Table 5.2. The graph shows that using BRO assisted the CBA to achieve the *Best* per cent error of 0 for all [51,150] datasets except KroB150. For datasets with 200 dimensions, the CBA+BRO could only achieve an average *Best* error of 0.25 per cent. The *Avg* error of ten operating trials is less than 0.20 per cent in the [51,100] datasets. The [150,200] datasets have an error rate of 0.23 to 0.56 per cent. While the convergence rate (as described in Equation 2.10) can be expressed as the ratio of the number of evaluations required to reach the best solution to the total number of evaluations. We discovered a linear relationship between problem dimension and convergence rate ([51,52] = 2-2.1 per cent; [70,76] = 3.2-3.8 per cent; [99,100] = 4.3-5.3 per cent; [150] = 8.9-9.7 per cent; and [200] = 18.5-21.7 per cent). The Eil51's 2% convergence rate indicates that it would take 2% of the maximum number of evaluations to obtain the best solution. This association could serve as a stopping criterion for a dataset (the maximum number of evaluations depends on the dimensions).

The statistical test was performed on the same benchmark datasets used in Chapter 4 on all CBA versions utilising exploration strategy and CBA with exploitation strategy. The findings indicated that the CBA+BRO performed significantly better in terms of both precision and convergence than any other CBA with exploration strategy. Even if the CBA+BRO algorithm used a random number generator as the initial solution generator, the algorithm converged much faster than the CBA+BNSN versions. We assume that the abandonment technique, which repositions the patch, has a diversion capability. This is why integrating the BRO into the abandonment technique results in extremely high levels of accuracy and speed. We discovered that a con-

Table 5.2: Accuracy results of 10 runs of CBA+BRO using ten times dimension of colony size

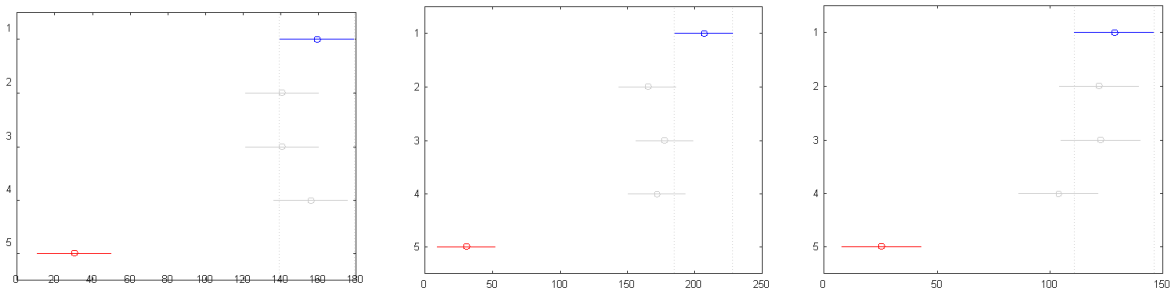
					Accuracy					Evaluation		
No	Datasets	Dims	BKS	Colony	*Best*	(Err.%)	*Avg*	(Err.%)	*StdDev*	*Best*	*Avg*	*StdDev*
F-1	Eil51	51	426	526	426	0	426.50	0.12	0.53	23,290	28,157.70	2,871.97
F-2	Berlin52	52	7,542	532	7,542	0	7,543.20	0.02	1.69	27,498	32,835.40	4,667.98
F-3	St70	70	675	721	676	0	676.20	0.18	0.42	52,632	66,946.70	9,642.41
F-4	Eil76	76	538	784	538	0	538.30	0.06	0.48	63,530	85,431.30	11,156.09
F-5	Pr76	76	108,159	784	108,159	0	108,318.70	0.15	98.91	62,686	74,772.50	7,578.68
F-6	Rat99	99	1,211	1,030	1,211	0	1,212.70	0.14	1.16	130,556	143,206.20	11,165.76
F-7	KroA100	100	21,282	1,036	21,282	0	21,302.90	0.10	25.69	148,705	156,433.00	8,011.91
F-8	KroB100	100	22,141	1,036	22,141	0	22,166.90	0.12	23.04	129,232	148,317.10	13,631.63
F-9	KroC100	100	20,749	1,036	20,749	0	20,759.40	0.05	7.56	135,902	156,369.10	12,766.16
F-10	KroD100	100	21,294	1,036	21,294	0	21,318.80	0.12	21.51	135,967	151,268.60	14,864.93
F-11	KroE100	100	22,068	1,036	22,068	0	22,091.40	0.11	15.44	133,239	160,857.60	21,124.02
F-12	KroA150	150	26,524	1,561	26,524	0	26,584.50	0.23	40.72	322,891	437,403.10	128,792.96
F-13	KroB150	150	26,130	1,561	26,148	0.07	26,192.70	0.24	7.85	319,253	401,142.20	55,776.49
F-14	KroA200	200	29,368	2,086	29,408	0.13	29,474.80	0.36	41.17	675,001	1,111,036.50	366,789.37
F-15	KroB200	200	29,437	2,086	29,548	0.38	29,602.00	0.56	27.54	1,143,203	1,302,217.40	125,092.35



(a) [50,76] (b) [99,100] (c) [150,200]

Figure 5.5: Kruskal Wallis ANOVA test on the accuracy of Exploration Vs Exploitation strategies of CBA for 15 datasets [50, 200] (1 = CBA; 2 = CBA+NNH; 3 = CBA+BNSN(2); 4 = CBA+BNSN(3); 5 = CBA+BRO)

structive heuristic plays a lesser role in all-dimensions datasets than the local search mechanism [51,200]. This result is consistent with Osaba *et al.* (2018). The CBA's detailed output with the exploration strategy version is detailed in Table 4.4 in the preceding chapter.



(a) [50,76] (b) [99,100] (c) [150,200]

Figure 5.6: Kruskal Wallis ANOVA test on the NFE of Exploration Vs Exploitation Strategy of CBA for 15 datasets [50,200] (1=CBA; 2=CBA+NNH; 3=CBA+BNSN(2); 4=CBA+BNSN(3); 5=CBA+BRO)

5.4.2 Comparative analysis of algorithms

This section discusses the CBA outcomes that employ both exploration and exploitation strategies (CBA + BNSN + BRO) and its comparative analysis studies to other nature-inspired algorithms. Table 5.4 and 5.5 discuss the accuracy efficiency and convergence behaviour of CBA + BNSN + BRO with a colony size of 40 bees in depth with varying diversity level (F = [1-3] for the initial solution generator). The accuracy (error) and number function evaluations serve as success metrics for precision and convergence. For each benchmark dataset, the entire CBA+BNSN+BRO scenario was run 30 times for a limit of 800,000 evaluations ($MaxEv$). The

tables demonstrate that no one exercised dominance over the others (the bold font is the better solution between them). It can be concluded that the combination of BNSN + BRO has not a significant effect on the performance of the CBA algorithm as the CBA with only using BRO already has very accurate solutions (see Tabel 5.2).

The CBA with both exploration and exploitation strategies would be compared to other algorithms using the same strategies. As stated previously, the benchmarking algorithms must be comparable not just in terms of population-scale but also in terms of exploration and exploitation mechanisms (Osaba *et al.*, 2018). The metaheuristic algorithm based on the neutral local search operator can not be equivalent to the iterative improvement local search algorithm. All benchmarking algorithms in this comparative studies were based on the same population size and used the same class of constructive heuristics for initialisation and iterative improvement for the neighbour search mechanism.

The first series of benchmark algorithms are BCO by (Wong *et al.*, 2010) and two variants of ABC (CABC and qCABC) by (Karaboga and Gorkemli, 2019) which used a Nearest Neighbour class constructive heuristic and a 2-opt class of iterative improvement techniques. The BCO was initialised with NNH and searched for neighbours using the fixed radius nearest neighbour 2-Opt (FRNN + 2-Opt) with Frequency-Based Pruning Strategy (FBPS). The local search evaluates the closest vertex within the radius of the assigned vertex. The two edges exchange movement is formed by combining two vertex edges (the circle's centre and the closest vertex inside the radius). To improve the efficiency of the FRNN 2-Opt, the author suggested the FBPS, which employs building block (GA) principles to prune the assessments. The ABC used NNH for initialisation and the Greedy Sub-Tour Mutation (GSTM) for neighbour search (Albayrak and Allahverdi, 2011). The GSTM subtracts a (greedy) sub-tour from a full tour, connects the remaining tour, and copies the sub-tour. The sub-tour was created by selecting two vertices (for the start and endpoints) and connecting them to other vertices using the greedy rule. The rolling approach is used to assess the feasibility of reconnecting the remaining tour with the sub-tour. While the CBA initialised using BNSN and searched for neighbours using BRO. The BRO method is comparable to the GSTM procedure in that it has a subtracting and

Table 5.3: A comparison of BCO, ABC, and CBA

Problems		BCO (**)	CABC (*)	qCABC (*)	CBA + BNSN[1-3]+BRO
Berlin52	*Avg* Err.(%)	**0.00**	**0.00**	**0.00**	**0.00**
	Best Err.(%)	**0.00**	**0.00**	**0.00**	**0.00**
KroA100	*Avg* Err.(%)	**0.00**	0.04	0.01	0.20
	Best Err.(%)	**0.00**	**0.00**	**0.00**	**0.00**
Pr144	*Avg* Err.(%)	**0.00**	0.16	0.15	0.09
	Best Err.(%)	**0.00**	**0.00**	**0.00**	**0.00**
Ch150	*Avg* Err.(%)	**0.00**	0.44	0.52	0.50
	Best Err.(%)	**0.00**	0.21	0.25	**0.00**
KroB150	*Avg* Err.(%)	**0.00**	0.69	0.72	0.34
	Best Err.(%)	**0.00**	0.21	0.24	**0.00**
Pr152	*Avg* Err.(%)	**0.00**	0.15	0.22	0.13
	Best Err.(%)	**0.00**	**0.00**	**0.00**	**0.00**
Rat195	*Avg* Err.(%)	**0.09**	1.16	1.18	0.92
	Best Err.(%)	**0.00**	0.60	0.73	0.22
D198	*Avg* Err.(%)	**0.08**	0.47	0.48	0.39
	Best Err.(%)	**0.05**	0.29	0.20	0.08
KroA200	*Avg* Err.(%)	**0.00**	0.51	0.48	0.85
	Best Err.(%)	**0.00**	0.34	0.18	0.05
Ts225	*Avg* Err.(%)	0.62	**0.00**	**0.00**	0.13
	Best Err.(%)	0.42	**0.00**	**0.00**	**0.00**
Pr226	*Avg* Err.(%)	**0.00**	0.78	0.85	0.22
	Best Err.(%)	**0.00**	0.61	0.54	**0.00**
Pr299	*Avg* Err.(%)	**0.03**	0.95	0.88	0.85
	Best Err.(%)	**0.00**	0.58	0.32	0.01
Lin318	*Avg* Err.(%)	**0.09**	2.35	2.28	1.28
	Best Err.(%)	**0.00**	1.73	1.77	0.29
Pcb442	*Avg* Err.(%)	**0.82**	1.50	1.47	1.36
	Best Err.(%)	0.66	1.00	0.91	**0.51**
Fl1577	*Avg* Err.(%)	-	2.44	2.29	1.48
	Best Err.(%)	-	1.87	1.92	0.51
Average	*Avg* Err.(%)	**0.12** (-)	0.66 (0.78)	0.66 (0.77)	0.52 (0.58)
	Best Err.(%)	**0.08** (-)	0.40 (0.50)	0.37 (0.47)	**0.08** (0.11)

**= (Wong *et al.*, 2010)

*= (Karaboga and Gorkemli, 2019)

Table 5.4: Accuracy results of 10 runs of CBA+BNSN(1,2,3)+BRO using a colony size of 40 bees

		CBA + BNSN(1) + BRO					CBA + BNSN(2) + BRO					CBA + BNSN(3) + BRO				
Datasets	BKS	*Best*	(Err-%)	*Avg*	(Err-%)	*StdDev*	*Best*	(Err-%)	*Avg*	(Err-%)	*StdDev*	*Best*	(Err-%)	*Avg*	(Err-%)	*StdDev*
Berlin52	7,542	7,542	**0.00**	7,542.0	**0.00**	0.00	7,542	**0.00**	7,542.0	**0.00**	0.00	7,542	**0.00**	7,542.0	**0.00**	0.00
KroA100	21,282	21,282	**0.00**	21,326.9	0.21	90.37	21,282	**0.00**	21,320.6	**0.18**	19.12	21,282	**0.00**	21,323.1	0.19	108.08
Pr144	58,537	58,537	**0.00**	58,561.8	**0.04**	13.99	58,537	**0.00**	58,617.2	0.14	45.50	58,537	**0.00**	58,585.0	0.08	27.89
Ch150	6,528	6,553	0.38	6,563.5	0.54	14.08	6,528	**0.00**	6,559.4	**0.48**	17.68	6,528	**0.00**	6,559.8	0.49	16.69
KroB150	26,130	26,130	**0.00**	26,188.9	**0.23**	72.55	26,130	**0.00**	26,258.4	0.49	70.16	26,130	**0.00**	26,211.5	0.31	95.77
Pr152	73,682	73,682	**0.00**	73,763.6	0.11	70.23	73,682	**0.00**	73,881.5	0.27	129.62	73,682	**0.00**	73,695.6	**0.02**	43.01
Rat195	2,323	2,331	0.34	2,352.5	1.27	11.37	2,332	0.39	2,340.6	0.76	6.19	2,323	**0.00**	2,337.1	**0.60**	8.14
D198	15,780	15,813	0.21	15,826.7	0.30	13.33	15,842	0.39	15,872.1	0.58	26.36	15,793	**0.08**	15,825.3	**0.29**	20.08
KroA200	29,368	29,486	0.40	29,588.1	0.75	137.78	29,480	0.38	29,591.2	0.76	94.63	29,382	**0.05**	29,671.7	1.03	222.15
Ts225	126,643	126,643	**0.00**	126,713.1	**0.06**	50.95	126,880	0.19	126,967.6	0.26	52.23	126,713	0.06	126,744.9	0.08	52.91
Pr226	80,369	80,373	**0.00**	80,512.4	0.18	172.69	80,459	0.11	80,646.7	0.35	177.49	80,373	**0.00**	80,484.1	**0.14**	201.38
Pr299	48,191	48,297	0.22	48,643.2	0.94	304.33	48,400	0.43	48,701.8	1.06	238.72	48,195	**0.01**	48,451.7	**0.54**	259.60
Lin318	42,029	42,219	0.45	42,390.3	0.86	162.31	42,787	1.80	42,930.6	2.15	142.64	42,149	**0.29**	42,374.0	**0.82**	121.55
Pcb442	50,778	51,036	**0.51**	51,225.5	**0.88**	177.81	51,207	0.84	51,740.3	1.90	368.51	51,130	0.69	51,436.3	1.30	171.92
Fl1577	22,249	22,401	0.68	22,531.9	1.27	115.30	22,363	**0.51**	22,719.7	2.12	143.85	22,363	**0.51**	22,482.1	**1.05**	104.21

Table 5.5: The NFE results of 10 runs of CBA+BNSN(1,2,3)+BRO using a colony size of 40 bees

		CBA	+ BNSN(1) +	BRO	CBA	+ BNSN(2) +	BRO	CBA	+ BNSN(3) +	BRO
Datasets	*MaxEv*	*Best*	*Avg*	*StdDev*	*Best*	*Avg*	*StdDev*	*Best*	*Avg*	*StdDev*
Berlin52	800,000	905	3,490.30	2,307.14	1,832	5,121.60	3,156.06	**811**	**2,628.90**	1,619.33
KroA100	800,000	**2,108**	**77,606.30**	46,451.55	5,700	224,732.20	279,018.60	45,205	108,466.60	85,670.22
Pr144	800,000	3,121	31,864.60	18,057.76	**2,833**	76,879.00	139,740.64	4,994	**30,086.10**	18,928.96
Ch150	800,000	53,923	328,914.80	186,957.86	**25,180**	322,689.90	279,575.00	50,407	**313,915.30**	229,370.28
KroB150	800,000	**49,747**	**225,763.80**	152,421.04	73,580	377,548.10	287,314.29	50,187	300,040.30	233,230.77
Pr152	800,000	46,290	90,871.70	56,781.13	**25,206**	237,343.20	251,494.88	27,319	**80,737.00**	78,498.66
Rat195	800,000	59,963	**293,011.90**	197,259.52	98,215	356,855.00	210,661.03	**59,500**	295,932.00	240,553.09
D198	800,000	48,512	523,363.50	279,124.40	154,441	433,633.90	199,847.17	**48,167**	**430,527.90**	268,157.07
KroA200	800,000	49,465	432,579.10	268,623.77	**42,855**	378,754.10	283,009.83	50,675	**346,721.00**	200,354.90
Ts225	800,000	49,642	67,662.80	16,752.21	**13,225**	**46,957.90**	38,056.74	26,290	86,721.30	77,664.68
Pr226	800,000	**48,299**	474,054.60	267,130.48	160,595	473,419.40	197,334.49	261,708	**461,558.10**	96,234.50
Pr299	800,000	**51,745**	483,115.30	320,312.98	132,364	586,027.80	211,485.30	91,489	**442,020.20**	240,194.46
Lin318	800,000	**48,891**	**416,752.50**	234,363.30	86,638	497,308.20	242,435.47	123,238	467,021.50	201,877.81
Pcb442	800,000	**51,803**	457,395.60	265,385.02	**390,921**	579,141.30	162,122.51	193,702	474,661.00	143,580.94
Fl1577	800,000	499,151	702,593.18	108,241.62	**331,561**	656,924.02	142,723.43	419,001	**653,092.50**	138,171.81

reconnecting step. The local search method between CBA and CABC are quite close, but they differ in three ways. First, the BRO did not subtract a sub-tour in any way other than deleting a vertex or set of vertices. The omitted nodes can result in a sharp turning angle on the remaining tour. Second, the remaining tour's sharp turning angle must be removed using 2-Opt before re-connecting it to the forgotten node(s). Finally, the BRO added the vertices one by one, utilising the vertex-edge distance to determine the most suitable edge to implant.

This comparative analysis worked on fifteen benchmark problems ranging from 52 to 1,577 dimensions (Berlin52 to Fl1577). For all experiments, algorithms were run ten times except the CBA for 30 times. Algorithms were set with colony sizes of 40 and 800,000 evaluations excepts the NFE of BCO equal to Dims*10,000. As seen in Table 5.3, the CBA and BCO have the highest overall accuracy results, with an overall average $Best$ error of 0.08 per cent. The BCO, on the other hand, has higher average accuracy (the average of Avg (mean) error) of all instances, by 0.4 per cent better than the CBA. The detail of the three varying BNSN (F=[1-3]) of CBA+BNSN+BRO could be seen in Table 5.4 and 5.5. At the same time, ABC underperforms behind in both BCO and CBA accuracy metrics. It could be assumed that the amount of maximum evaluation metric utilised by ABC (also CBA) 800,000 maximum evaluations incomparable to some instances. Applying algorithms with 800,000 evaluations to an instance of 299 dimensions (299 factorial of solutions) would likely have a higher percentage of error than the experiment with 2,990,000 evaluations.

However, even though CBA outperforms CABC on the overall accuracy ($Best$ and Avg) errors, it applied differently to the individual (datasets) results. The statistical tests are conducted for all datasets one by one comparison. It used the error bar of the confidence interval test as it assumed that the CBA data on its 30 data points were normal (The data could be accessed in the appendix section). The error bar test outputs of CBA+BNSN+BRO vs CABC Vs qCABC for 15 [52,1577] datasets are depicted in Figure 5.7. The figures show that CBA produces significantly better solutions than CABC and qCABC in seven instances, namely Pr144, KroB150, Rat195, D198, Pr226, Lin318 and Fl1577. Six of the eight instances have a performance similar to CABC and qCABC, namely Berlin52, Ch150, Pr152, Pr299, and Pcb442. Despite the fact



Figure 5.7: Statistical test (95% confidence level) on the accuracy performance of CABC Vs qCABc Vs CBA+BNSN+BRO

Table 5.6: Accuracy comparison of CBA Vs BCO for [1173,1379] datasets

		CBA							BCO	
	BKS	*Best*	*Avg*	*StdDev*	*Avg.Eval*	*EvalRat*	*Best* Err (%)	*Avg* Err (%)	*Best* Err	*Avg* Err
Datasets	(1)	(3)	(4)	(5)	(6)	(6)/*Max Ev*	((3)-(1))/(1)	((4)-(1))/(1)	(%)	(%)
Pcb1173	56892	57972	58485.25	538.89	8,940,935	0.76	**1.90**	**2.80**	2.76	3.05
D1291	50801	51215	51718.8	608.35	10,826,523	0.84	**0.81**	1.81	0.82	**0.93**
Rl1304	252948	257144	259279.2	1367.41	11,908,279	0.91	**1.66**	**2.50**	2.34	2.95
Rl1323	270199	273879	276194.3	1751.33	11,998,696	0.91	**1.36**	2.22	2.15	**2.21**
Nrw1379	56638	57900	58152.33	265.33	13,411,946	0.97	**2.23**	**2.67**	4.05	4.18
Average						0.88	**1.59**	**2.40**	2.42	2.66

Table 5.7: Accuracy comparison of CBA Vs other nature-inspired algorithms

				CBA + BNSN	+ BRO
		Best	*Avg*	*Best*	*Avg*
Algorithms	List of Dims/Total instances	Err(%)	Err(%)	Err(%)	Err(%)
ACS+NN+3-opt	[198,318,783]/ 3 instances	0.045	0.124	0.740	1.167
MMAS+NN+2-opt	[100,198,318,442,783]/ 5	-	0.635	0.545	0.959
MMAS+NN+LK	[318,442,783,1173]/ 4	-	0.002	1.136	1.775
GA+NN+GSTM	[52,100,144,150,152,195,198, 200,225,226,299,318,442]/ 14	0.663	1.565	0.082	0.440

that the CBA's average value for such instances except KroA100 is higher than ABC's. CABC and qCABC outperformed CBA in the other two cases, KroA200 and Ts225. A statistical test of the accuracy could only be performed on ABC and CBA due to ABC's work (Karaboga and Gorkemli, 2019) yielded enough information (especially the standard deviation) for the significant test.

While the BCO provides an excellent accuracy on the overall average solution to the collection of [52,1577] instances, there was evidence that the output disparity between it and the CBA was narrowing in larger instances (see Ts225 and Pcb442). This finding leads to the expansion of the comparison analysis between CBA and BCO to five larger [1173,1379] benchmark problems for each of the five replications in Table 5.6. As the table indicates, CBA reported more accurate values than BCO for [1173,1379] datasets. CBA could reach a better best solution in all cases, while BCO could reach a better solution in an average of two instances (D1291 and Rl1323).

The CBA was also compared to other nature-inspired algorithms that incorporate a constructive and improvement heuristic (see Table 5.7). Three ant-inspired algorithms are presented,

Table 5.8: The convergence comparison of CBA Vs ABC

Problems		CABC	qCABC	CBA+BNSN +BRO
KROA100	Tour Length	**21,282**	**21,282**	**21,282**
	Evaluations needed	61,688	**1,680**	51,674
	Max. Evaluation	800,000	800,000	800,000
D198	Tour Length	15,825	15,820	**15,793**
	Evaluations needed	408,388	**269,680**	749,995
	Max. Evaluation	800,000	800,000	800,000
PCB442	Tour Length	51,309	51,240	**51,036**
	Evaluations needed	**145,440**	339,055	612,182
	Max. Evaluation	800,000	800,000	800,000
FL1577	Tour Length	22,703	22,742	**22,363**
	Evaluations needed	651,242	**328,763**	749,901
	Max. Evaluation	800,000	800,000	800,000

using 2-Opt, 3-Opt, LK, and a GA with GSTM as improvement heuristics. They all applied the constructive heuristic of the nearest neighbour class. As can be shown, CBA can only produce a stronger solution than GA.

In term of convergence performance, the number of function evaluation needed metric was used to identify the convergence efficiency of the proposed version to the benchmark algorithms. According to Osaba *et al.* (2018), it is preferable to compare convergence results using the evaluation number rather than the time measure. Comparing time results from separate machines introduces bias, and the time indicator might be fitted to a comparative study if all benchmark algorithm experiments were conducted on the same machine. However, using the number of function evaluations needed as a convergence indicator has a limitation, as not all works have the necessary details. From the works mentioned previously, only CBA and ABC provide the evaluation needed information. Except for ACS without a local search heuristic, ant-inspired algorithms and GA-based algorithms (Dorigo and Gambardella, 1997; Stutzle and Hoos, 1997; Stützle and Hoos, 2000; Albayrak and Allahverdi, 2011) do not provide the evaluation information required. The comparison of CBA and ABC's convergence performance is seen in Table 5.8. As can be seen from the table, CBA has a larger NFE than ABC. However, except for KroA100, the best solution that CBA could reach better than ABC. That is, CBA found the better local optima within 800,000 evaluations than CABC. The CBA has a better

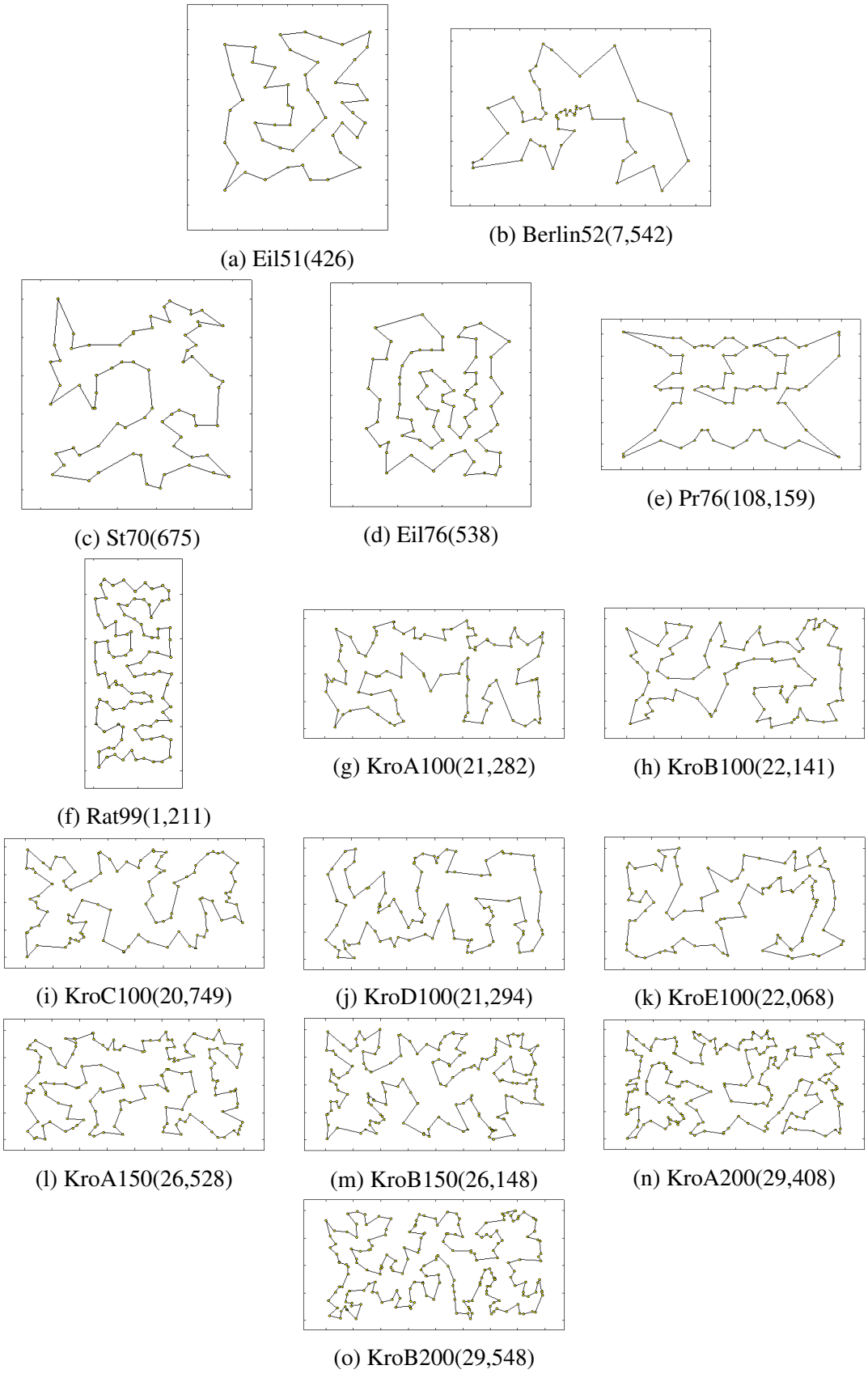
ability to escape from the local optima trap than CABC.

5.5 Summary

This chapter introduces a new intensifier, the Bees Routing Optimiser (BRO), which enhances the accuracy and convergence performance of the Combinatorial Bees Algorithm. The first investigation compared the CBA+BRO algorithm against the previous CBA+BNSN strategy on [51,200] TSPLIB datasets. The finding confirms Osaba *et al.* (2018) and Karaboga and Gorkemli (2019) that a local search strategy contributes more to a metaheuristic algorithm than a global search strategy does.

The second investigation compared the CBA+BNSN+BRO algorithm to bee-inspired algorithms on 15 [51,442] ([51,1577]) TSPLIB datasets. The finding indicates that the proposed algorithm with the 40 bees population had an average of best and average errors of 0.08% (0.11%) and 0.52% (0.58%). If the 1000s dimensions [1173,1379] datasets are included, the overall accuracy performance will degrade with $\overline{Best\ Err} = 0.48\%$ and $\overline{Avg\ Err} = 1.01\%$.

For 15 datasets of [52,1577] dimensions, the algorithm's convergence performance could obtain the best solution ($Best$) for an average of 41% of the maximum evaluations ($MaxEv$), ranging from 10% to 83%. It had a lower evaluation rate performance with 88% on average for 1000-dimensional [1173,1379] datasets (range from 76% to 97%). The rate of evaluations tends to increase in lockstep with the problem's dimensions.



(a) Eil51(426)

(b) Berlin52(7,542)

(c) St70(675)

(d) Eil76(538)

(e) Pr76(108,159)

(f) Rat99(1,211)

(g) KroA100(21,282)

(h) KroB100(22,141)

(i) KroC100(20,749)

(j) KroD100(21,294)

(k) KroE100(22,068)

(l) KroA150(26,528)

(m) KroB150(26,148)

(n) KroA200(29,408)

(o) KroB200(29,548)

Figure 5.8: The best solution tours of CBA+BRO for [51,200] TSPs' instances

Chapter 6

Two engineering applications

The focus of the work presented in this chapter is to implement the new developments of BA in two selected engineering applications. These are PCB assembly sequence optimisation and vehicle routing, two complex problems that are representative of combinatorial optimisation problems in the real world. The optimisation of PCB assembly was accomplished by utilising two distinct types of CBA. The first is the CBA + BNSN with seed (Appendix H). The CBA + BNSN without seed is the second (Appendix I). CBA + Bi-BA + BRO were used to resolve the VRP (Appendix J).

6.1 Printed circuit board assembly sequence optimisation

6.1.1 Preliminaries

Printed circuit boards (PCBs) are an essential part and widely used inside our technology devices (telecommunications, computer, and electronics). The electronic devices market, including its component, is a massive global market and still predicted to grow in the future. According to Castellani *et al.* (2019), PCB assembly is the process of putting electronic parts of various shapes and sizes at explicit areas on the exposed board utilising the surface-mount technology placement machines. These kinds of machines can do high-speed part placement and can handle high and rapid production demand. However, the planning process of assembly perhaps the

most tedious stages in PCB production.

The hardest challenge in PCB assembly is the optimisation of the Moving Board with Time Delay (MBTD) problem, which optimises the component sequence and the feeder arrangement (see Figure 6.1). It combines two complex combinatorial non-deterministic polynomial time (NP)-hard problems,i.e., Travelling Salesman Problem (TSP) and Task Allocation Problem (TAP) for feeder arrangement. It means that it is not easy and time-consuming to find a good solution.

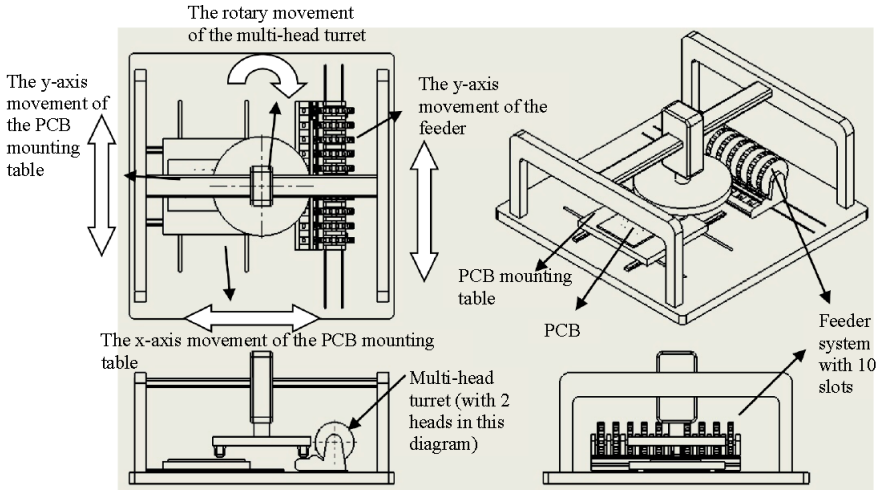


Figure 6.1: PCB assembly machine of the MBTD type (with 2 Rotary Turret Heads, 10 Feeder Slots and a Move-able Assembly Table) (Castellani *et al.*, 2019)

Intelligent optimisation methods are known to provide satisfactory solutions to complex tasks such as NP-complete combinatorial problems (Pham and Karaboga, 2012). There are various attempts to find the optimal solution for the problem since the beginning of the 1980s. They are Kirkpatrick *et al.* (1983) with Simulated Annealing (SA), Nelson and Wille (1995) with Evolutionary Programming (EP), Fogel (2006) with Genetic Algorithms (GA), Pham *et al.* (2007j) with Bees Algorithm (BA), Ang *et al.* (2009b, 2010, 2013b) with CBA using the Teoriya Resheniya Izobretatelskikh Zadatch (TRIZ), and Castellani *et al.* (2019) with CBA using five local search operators (block insertion, single-point insertion, reversion, simple swap, and neighbour swap).

In this study, the MBTD problem will be solved by two versions of CBA using the Bees Nearest Straightest Neighbour (BNSN), with and without seeds.

6.1.2 Problem definition

The MBTD model of planning PCB assembly involves two primary tasks: setup management and process optimisation. The ultimate objective is to significantly reduce assembly cycle time. Three moving parts compose the problem: first, the feeder, which travels along a single axis, moves the required component to the pick-up site, located in the centre of the array of feeders, for assembly. Second, the fixed-axis, two-head rotating turret that picks up components from the feeder array with one head while simultaneously placing another component on the PCB board with the other head. Finally, there is a rotating table that carries the board. The table moves in order to align the PCB with the set component placement position (See Figure 6.1).

Since the suitable feeder and table have reached their assigned locations, component pickup and placement occur concurrently. It is necessary to synchronise the multiple feeders, the two-head turret pick-and-place system, and the assembly table to complete the job. Due to the fact that this type of machine has three moving components (the board, feeder, and turret), one of these three machine elements must wait for the other two to finish their movement before picking up and placing the next component. Thus, the time required to place a single component (τ_k) is equal to the maximum between the times required for board movement (t_1), feeder movement (t_2), and turret indexing (t_3). The cumulative assembly time for all components is equal to the number of times required to place each of the 50 components (Equation 6.1). The primary challenge in reducing the overall assembly sequence time is that two optimisation issues must be solved concurrently: the component positioning sequence and the component arrangement inside the feeders. There would be 50 components of ten different types in this scenario. The components positions are depicted in Figure 6.2.

$$AT_{total} = \sum_{k=1}^{K} \tau_k \tag{6.1}$$

$$\tau_k = max(t_1(Comp_{k-1}, Comp_k), t_2(f_{k+g-1}, f_{k+g}), t_3) \tag{6.2}$$

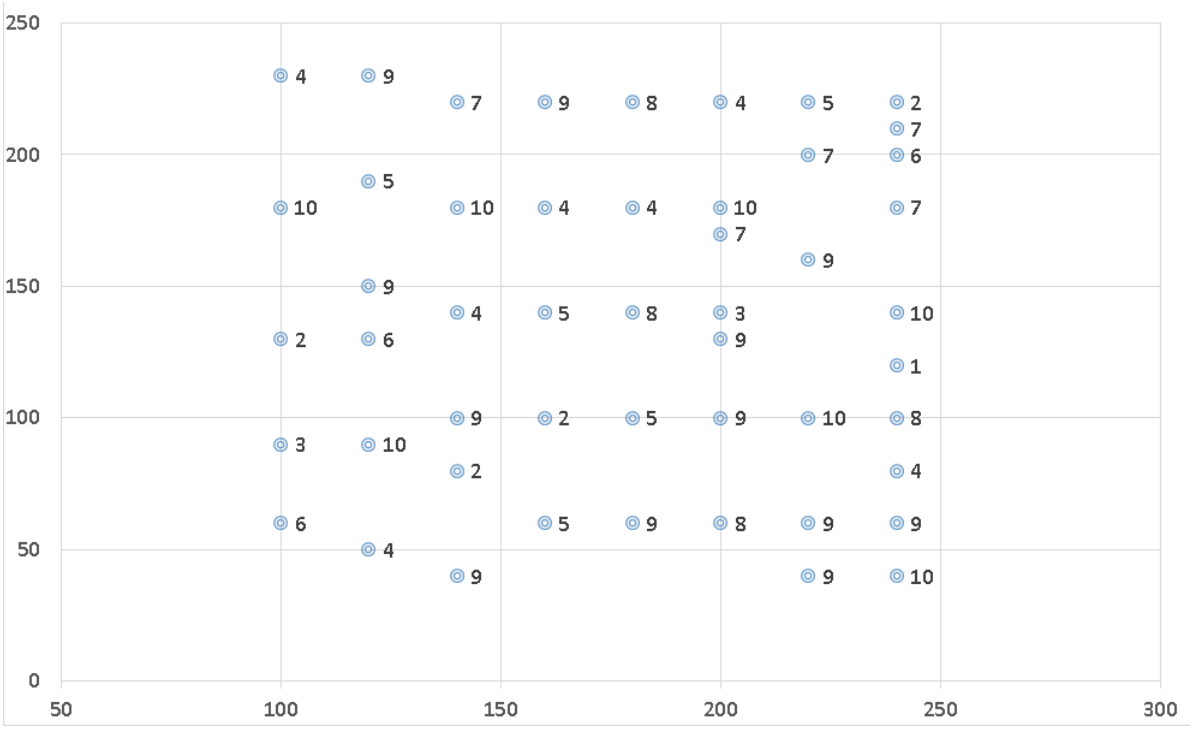


Figure 6.2: The positions of all ten components on the board

$$t_1(Comp_i, Comp_j) = max\left(\frac{|x_j - x_i|}{vc_x}, \frac{|y_j - y_i|}{vc_y}\right) \tag{6.3}$$

$$t_2(f_i, f_j) = \frac{\sqrt{\left|x_j^f - x_i^f\right|^2 + \left|y_j^f - y_i^f\right|^2}}{vc_f} \tag{6.4}$$

Where,

AT_{total} = total assembly time for all components from k =1 to K on to PCB.

K = total components to be inserted to PCB board= 50 components.

τ_k = the time needed to place a component, $Comp_k$. $Comp_{k-1}$ = $Comp_K$ when k=1.

g = gap between turret heads.

$Comp$ = component sequence from $Comp_1$ to $Comp_K$ = $\{Comp_1, .., Comp_i, ..., Comp_{K-1}, Comp_K\}$

$Comp_i$ = the i_{th} component to be placed.

FA = feeder assignment sequence from f_1 to f_R = $\{f_1, ..., f_j, f_{FR-1}, f_{FR}\}$

f_j = the feeder for the j_{th} component type.

FR = Total number of feeders= 10 feeders.

$t_1(Comp_i, Comp_j)$ = time between placement location of component i and component j.

$Comp_i(x_i, y_i)$ = component i with co-ordinate (x_i, y_i).

$Comp_j(x_j, y_j)$ = component j with co-ordinate (x_i, y_j).

vc_x= velocities of the X-Y table in the x-direction = 60 mm/s.

vc_y= velocities of the X-Y table in the y-direction = 60 mm/s.

$t_2(f_i, f_j)$ = travelling time of the feeder carrier between feeder $f_i(x_i^f, y_i^f)$ and $f_i(x_j^f, y_j^f)$.

vc_f = speed of the feeder carrier = 60 mm/s.

$f_i(x_i^f, y_i^f)$ = feeder i.

$f_i(x_j^f, y_j^f)$ = feeder j.

t_3 = turret indexing time = 0.25 (s/step).

6.1.3 The Combinatorial Bees Algorithm for PCB insertion sequence optimisation

The problem's solutions consist of two arrays. The first array denotes the feeder configuration, while the second denotes the component placement sequence. A valid feeder arrangement is any permutation of feeder labels, each label corresponding to a different feeder and component form. The feeder arrangement will be processed and improved on the global search phase, while the sequence of components placements solution on the neighbour search mechanism of CBA.

In this study, we use two approaches, CBA with and without seeding operator, to solve the PCB assembly problem. The Domino Sequence Heuristic (DSH) (Ismail, 2019), a constructive heuristic method, was used to generate the sequence of the feeder arrangement. The main reason for choosing the DSH is to make a Gaussian distribution shape of the arrangement of the components. The smallest frequency of the component will likely lies on the tail of the distribution. The components with large frequency have to put side to side on the feeder as the feeder's movement depends on the gap between the compartments of components. The sequence of the component arrangement (inside the feeder) generated by the DSH generator was based on the minimum distance of the centroid points of each component type. The DSH

starts the sequence from component 9 (12 pieces components) as the most frequent component.

Generally, both methods working similarly, the seeding approach excluded the feeder arrangement mechanism procedure inside the algorithm. The arrangement generated separately before the main algorithm works. The CBA with seeds method will search the components sequences based on the supplied feeder arrangements from the DSH (see Figure6.3). While the CBA without seeds used the DSH at the initialisation phase. The parameter setting followed the previous section (Table 4.3) with colony size ten times the problem's dimension and the DSH using set the two players procedure.

On the initialisation, the n initial solution of component sequences were generated using the BNSN operator (the CBA without seeds generates the component sequence after the feeder arrangement formed). These n valid solutions are evaluated (using Equation 6.1) and ranked. The algorithm then enters its main cycle, which is repeated for a given number of iterations. The e elite bees found the highest solutions, recruiting the nep foragers bees for neighbourhood search. The $m - e$ non-elite bees that found the highest solutions recruit the nsp of worker bees for exploitation. The worker bees (nep and nsp) exploit the local solution (component sequence) using three local search operator (simple swap, insert, and reverse) Ismail *et al.* (2020). The CBA without seed method will search both the components sequence and the feeder arrangement on the searching process of the algorithm.

The remaining bees will explore the global solution. The global search of the CBA with seeds will explore the component sequence, while CBA without seeds the feeder arrangement. The same thing happens at the abandonment stage. The global solution, either the feeder arrangement or component sequence, will be re-arranged when the maximum point of the stagnant limit reach (See Figure 6.4).

6.1.4 The experiment results

In this study, the number of evaluation that meets the best solution did not compare. According to Castellani *et al.* (2019), depending on the number of evaluation of one best lucky run is not fair enough. This study uses a smaller maximum number evaluation, $1.65x10^6$ evaluations, than

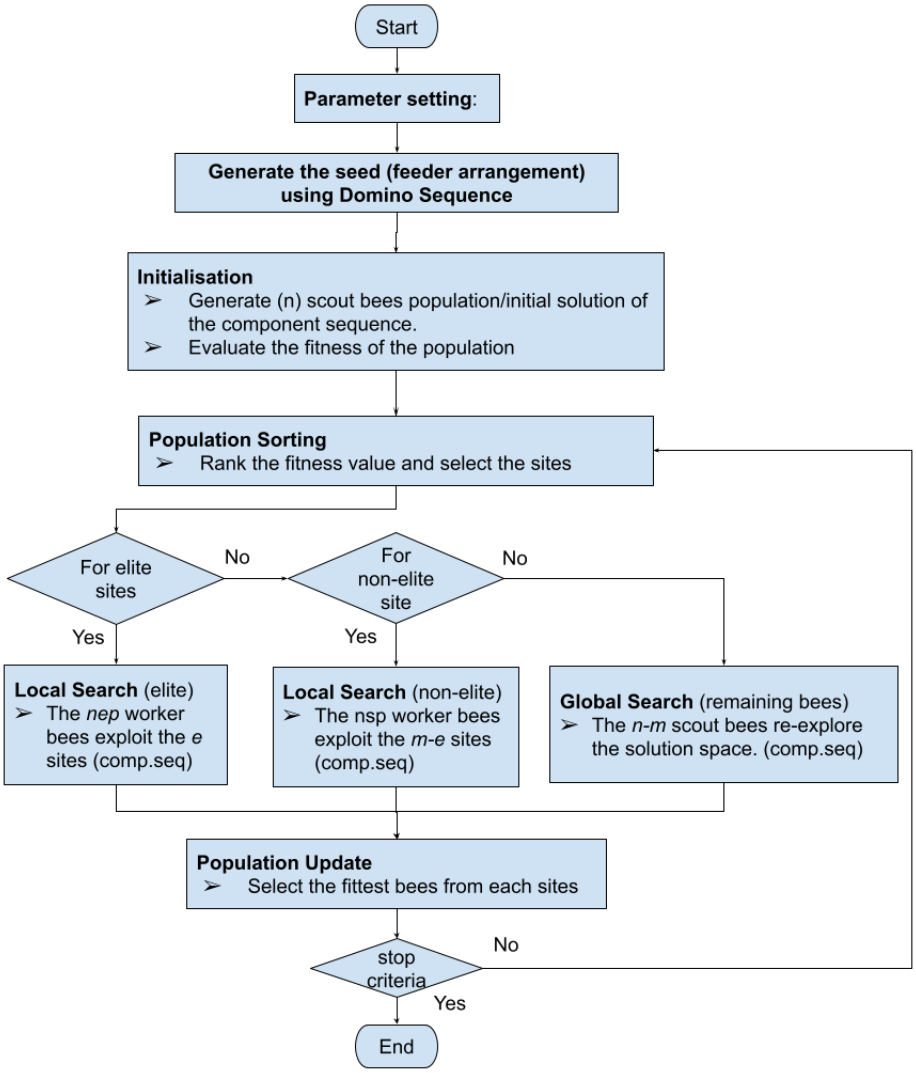


Figure 6.3: Flowchart of CBA with seed for PCB assembly optimisation

previous studies.

It has to remember that this problem considering three-movement and single component placement depends on these movements. So the minimum time assembly will be not equal to the shortest distance of the sequences. The problem is more complex compared to TSP as each feeder arrangement has one matrix of distance. If a 50 dimensions TSP with a single distance matrix has 50! possible solutions, this problem has 10! distance matrices. The complexity of this problem is far beyond the conventional TSP.

The summary of the optimisation results obtained using the CBA+BNSN with or without seed running for 30 runs is presented in Table 6.1. It also compared the proposed version with previous works of CBA. The table reports the best solutions obtained by CBA+BNSN with

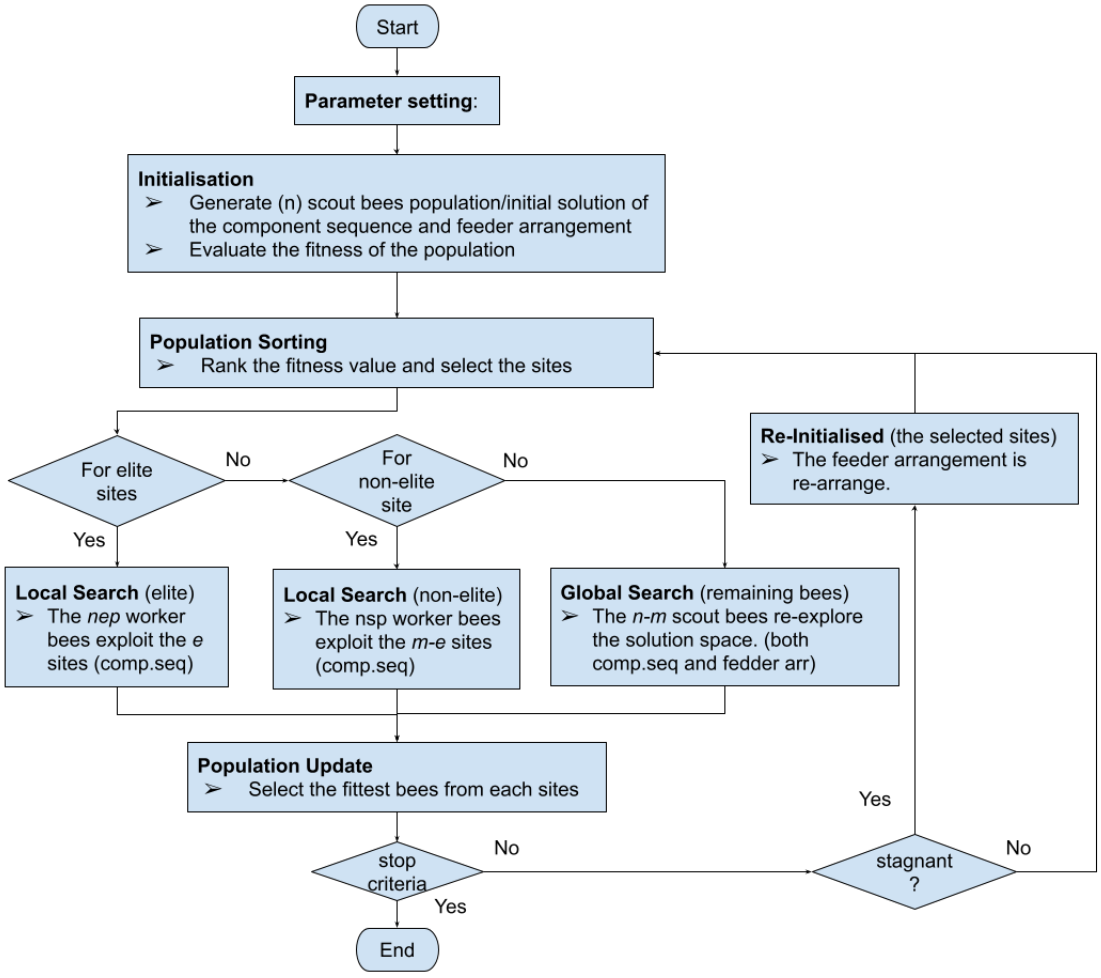


Figure 6.4: Flowchart of CBA without seed for PCB assembly optimisation

seed among all seven benchmark methods. Indeed, the proposed method found the solution of minimum assembly time (23.00 s). Although the maximum evaluation smaller, the best total assembly time could be reached 23 seconds by the CBA+BNSN using seed. The best solution is given in Figure 6.7, and the best solution of the previous study Castellani *et al.* (2019) is given in Figure 6.5.

The customised Bees Algorithm (cBA) used $6x10^6$ evaluations, obtaining a solution of 23.46 seconds total assembly time. In comparison, the CBA+TRIZ used more than $23x10^6$ evaluations and reached 23.58 seconds of total assembly time. Moreover, the cBA Castellani *et al.* (2019) can be considered the best algorithm without seeding procedures. The CBA+BNSN without seeding could only reach the best solution of 24.4167 seconds with an average value of 26.34 seconds of 30 independent running experiments. The overall result shows that cBA

could reach the lowest average value at 24.96 seconds, while CBA+BNSN using seeds is 25.26 seconds.

Table 6.1: MBTD benchmark: the comparison of the results obtained between the CBAs

References	1	2	3	4		
Methods	CBA	CBA+seed	CBA+TRIZ	cBA	CBA+BNSN	CBA+BNSN+seed
Best	25.92	24.08	23.58	23.46	24.42	23
Avg	n/a	n/a	n/a	24.96	26.34	25.26

1=Pham *et al.* (2007j)
2=Pham *et al.* (2007j)
3=Ang *et al.* (2013b)
4=Castellani *et al.* (2019)

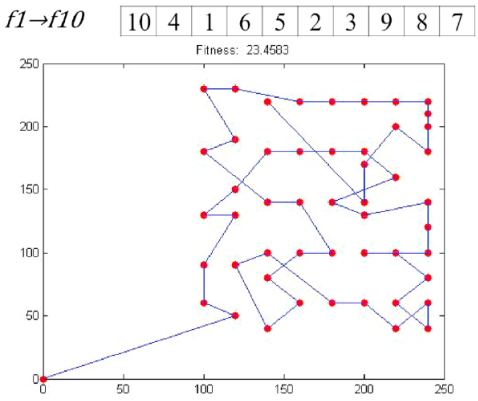


Figure 6.5: The best solution of Castellani *et al.* (2019) (23.46 s assembly time)

After running 30 independent experiments, the best solutions of the two proposed method are validated manually using Equation 6.1 to 6.4 calculation on table 6.2 and 6.3. The validation using simulation also had been conducted for CBA+TRIZ (Ang *et al.*, 2013b). Tabel 6.2 represents CBA's simulation with the seed method, while Table 6.3 is the method without the seed. Those tables comprise 50 rows that pose the sequence of component placement on the board and six columns. These columns include assembly locations, component type, travelling time between the placement of components due to movement of mounting table (t_1), travelling time of feeder carrier between feeders (t_1), turret indexing time (t_3), and the actual assembly time needed (dictated by the longest time needed of either t_1 or t_2 or t_3). The total assembly time is the sum of the actual assembly time needed of 50 component (sum the rows). The result shows that CBA+BNSN using seeds has fixed the previous record time by 0.46 seconds.

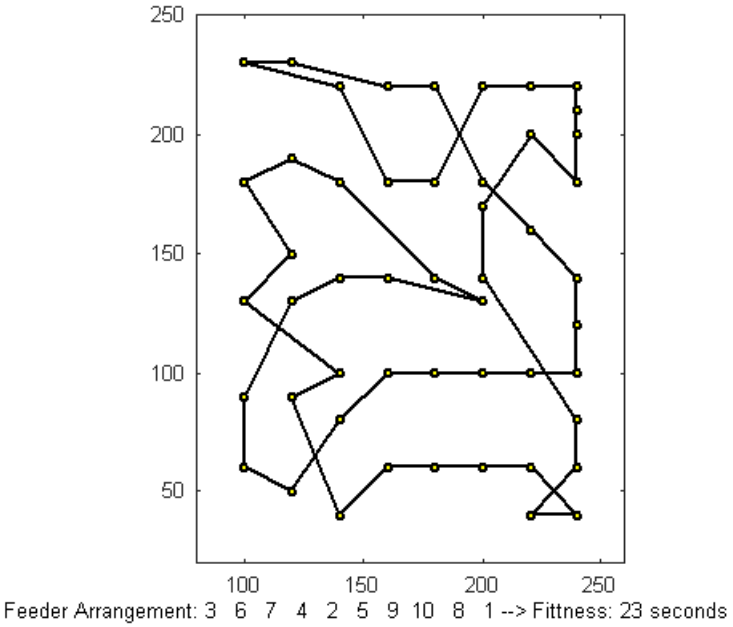


Figure 6.6: The best solution of CBA+BNSN using seeds (23.00 seconds assembly time)

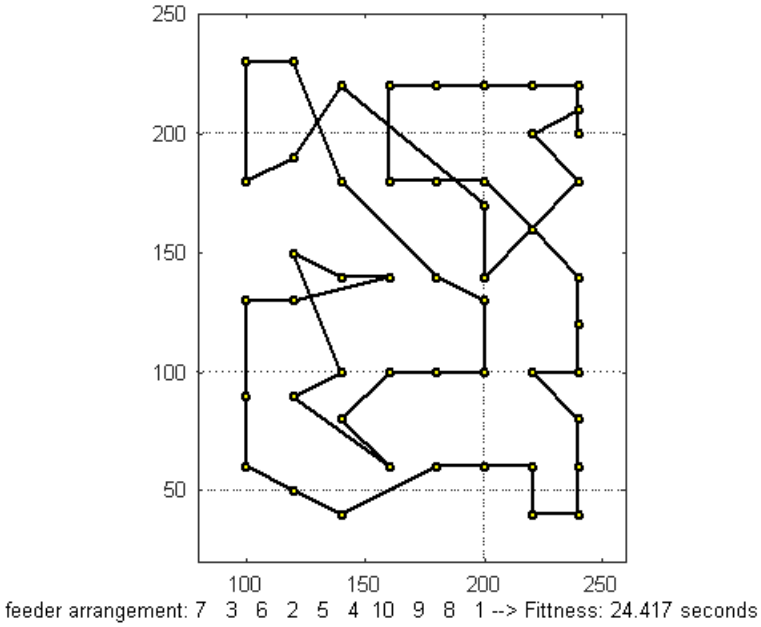


Figure 6.7: The best solution of CBA+BNSN (24.4167 seconds assembly time)

Table 6.2: The simulation data for the PCB assembly of 50 components obtained using the CBA+BNSN using seeds with a total assembly time is 23.00 seconds

x	y	comp.type	t_1	t_2	t_3	t_{max}
140	180	10	0.67	0.50	0.25	0.67
180	140	8	0.33	0.25	0.25	0.33
200	130	9	0.67	0.50	0.25	0.67
160	140	5	0.33	0.50	0.25	0.50
140	140	4	0.33	0.25	0.25	0.33
120	130	6	0.67	0.25	0.25	0.67
100	90	3	0.50	0.50	0.25	0.50
100	60	6	0.33	0.25	0.25	0.33
120	50	4	0.50	0.00	0.25	0.50
140	80	2	0.33	0.25	0.25	0.33
160	100	2	0.33	0.25	0.25	0.33
180	100	5	0.33	0.25	0.25	0.33
200	100	9	0.33	0.25	0.25	0.33
220	100	10	0.33	0.25	0.25	0.33
240	100	8	0.33	0.50	0.25	0.50
240	120	1	0.33	0.25	0.25	0.33
240	140	10	0.33	0.25	0.25	0.33
220	160	9	0.33	0.25	0.25	0.33
200	180	10	0.67	0.50	0.25	0.67
180	220	8	0.33	0.00	0.25	0.33
160	220	9	0.67	0.75	0.25	0.75
120	230	9	0.33	0.25	0.25	0.33
100	230	4	0.67	0.25	0.25	0.67
140	220	7	0.67	0.00	0.25	0.67
160	180	4	0.33	0.00	0.25	0.33
180	180	4	0.67	0.50	0.25	0.67
200	220	4	0.33	0.25	0.25	0.33
220	220	5	0.33	0.50	0.25	0.50
240	220	2	0.17	0.25	0.25	0.25
240	210	7	0.17	0.25	0.25	0.25
240	200	6	0.33	0.00	0.25	0.33
240	180	7	0.33	0.00	0.25	0.33
220	200	7	0.50	0.50	0.25	0.50
200	170	7	0.50	0.75	0.25	0.75
200	140	3	1.00	0.75	0.25	1.00
240	80	4	0.33	0.00	0.25	0.33
240	60	9	0.33	0.25	0.25	0.33
220	40	9	0.33	0.25	0.25	0.33
240	40	10	0.33	0.50	0.25	0.50
220	60	9	0.33	0.50	0.25	0.50
200	60	8	0.33	0.25	0.25	0.33
180	60	9	0.33	0.25	0.25	0.33
160	60	5	0.33	0.25	0.25	0.33
140	40	9	0.83	0.25	0.25	0.83
120	90	10	0.33	0.50	0.25	0.50
140	100	9	0.67	0.50	0.25	0.67
100	130	2	0.33	0.25	0.25	0.33
120	150	9	0.50	0.50	0.25	0.50
100	180	10	0.33	0.50	0.25	0.50
120	190	5	0.33	0.25	0.25	0.33
						23.00 sec

Table 6.3: The simulation data for the PCB assembly of 50 components obtained using the CBA+BNSN without seed with a total assembly time is 24.4167 seconds

x	y	comp.type	t_1	t_2	t_3	t_{max}
240	200	6	0.17	0.00	0.25	0.25
240	210	7	0.33	0.00	0.25	0.33
220	200	7	0.33	0.25	0.25	0.33
240	180	7	0.67	0.25	0.25	0.67
200	140	3	0.50	0.00	0.25	0.50
200	170	7	1.00	1.00	0.25	1.00
140	220	7	0.50	0.50	0.25	0.50
120	190	5	0.33	0.25	0.25	0.33
100	180	10	0.83	0.50	0.25	0.83
100	230	4	0.33	0.25	0.25	0.33
120	230	9	0.83	0.50	0.25	0.83
140	180	10	0.67	0.25	0.25	0.67
180	140	8	0.33	0.00	0.25	0.33
200	130	9	0.50	0.75	0.25	0.75
200	100	9	0.33	0.25	0.25	0.33
180	100	5	0.33	0.00	0.25	0.33
160	100	2	0.33	0.25	0.25	0.33
140	80	2	0.33	0.50	0.25	0.50
160	60	5	0.67	0.25	0.25	0.67
120	90	10	0.33	0.00	0.25	0.33
140	100	9	0.83	0.50	0.25	0.83
120	150	9	0.33	0.25	0.25	0.33
140	140	4	0.33	0.50	0.25	0.50
160	140	5	0.67	0.25	0.25	0.67
120	130	6	0.33	0.50	0.25	0.50
100	130	2	0.67	0.25	0.25	0.67
100	90	3	0.50	0.75	0.25	0.75
100	60	6	0.33	0.50	0.25	0.50
120	50	4	0.33	0.00	0.25	0.33
140	40	9	0.67	0.25	0.25	0.67
180	60	9	0.33	0.25	0.25	0.33
200	60	8	0.33	0.00	0.25	0.33
220	60	9	0.33	0.25	0.25	0.33
220	40	9	0.33	0.25	0.25	0.33
240	40	10	0.33	0.50	0.25	0.50
240	60	9	0.33	0.25	0.25	0.33
240	80	4	0.33	0.50	0.25	0.50
220	100	10	0.33	0.25	0.25	0.33
240	100	8	0.33	0.75	0.25	0.75
240	120	1	0.33	0.25	0.25	0.33
240	140	10	0.33	0.25	0.25	0.33
220	160	9	0.33	0.25	0.25	0.33
200	180	10	0.33	0.00	0.25	0.33
180	180	4	0.33	0.50	0.25	0.50
160	180	4	0.67	0.25	0.25	0.67
160	220	9	0.33	0.75	0.25	0.75
180	220	8	0.33	0.25	0.25	0.33
200	220	4	0.33	0.25	0.25	0.33
220	220	5	0.33	0.25	0.25	0.33
240	220	2	0.33	0.50	0.25	0.50
						24.41667

6.2 Vehicle Routing Problem

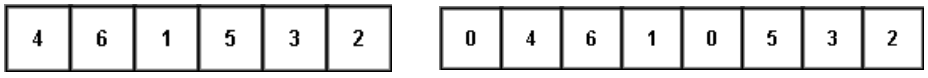
6.2.1 Preliminaries

This section will demonstrate how CBA is applied to the Vehicle Routing Problem (VRP). The basic CBA and the proposed version of CBA are compared to measure the impact of the novel strategy implemented in this problem. All the VRP instances were used from TSPLIB (Reinelt, 1994). The basic BA, which was initially developed in 2005 to solve continuous problems (Pham *et al.*, 2005, 2006c), have been improved to its combinatorial versions to solve TSP, PCB assembly, timetabling, machine scheduling, and others (Hussein *et al.*, 2017b).

As is the case for many other real-world COP applications, VRP is one of the most important real-world COP cases of a TSP extension. The problem has several salesmen, commonly referred to as vehicles with a specific capacity. They depart from one or more depots, travel to cities with specific demands, and then return to the depot (s).

6.2.2 Problem definition

In the Capacitated Vehicle Routing Problem (CVRP), a fleet of identical vehicles located at a central depot has to be optimally routed to supply a set of customers with known demands. The CVRP is described as the graph-theoretic problem: Let $G = (Vx, E)$ be a complete and undirected graph where $\{Vx = 0, \ldots, Dims\}$ is the vertex set and E is the edge set. Vertex set $\{Vx = 1, \ldots, n\}$ corresponds to $Dims$ customers, whereas vertex 0 corresponds to the depot(s).



(a) TSP Solution (b) VRP Solution

Figure 6.8: Solution representation of routing problems

Figure 6.8a means a salesman visits city 4, 6, 1, 5, 3, 2, and returns to starting city. Figure 6.8b means that vehicle-1 visits city 4, 6, 1, back to the depot (0) and vehicle-2 visits city 5,3,2, back to the depot. The principal decision VRP's variable $x_{ijv}(i, j = 0, 1, 2, \ldots, N; v =$

$1,2,\ldots,V; i \neq j$) is 1 if vehicle v travels from customer i to customer j, and 0 otherwise.

CVRP consists of finding tours for all v vehicles, all starting and ending at the depot(s). Each city is visited exactly once, and the total distance of visiting all n cities is minimised without capacity violation. Figure 6.8 illustrates a sample of a VRP solution with $N = 6$ and $v = 2$ and a TSP solution with $N = 6$. For solution x, let $L(x)$ denote the total length of the sequence, and $CV(x)$ denote a total violation of vehicle capacity. The TSP's and VRP's objective function accordingly will be:

$$f(x) = L(x) \tag{6.5}$$

$$f(x) = L(x) + \alpha.CV(x) \tag{6.6}$$

For CVRP's objective function (Equation 6.6), The $Z(x)$ will be equal to $L(x)$ if there is no capacity violation ($CV(x) = 0$). In this study, the α has to be set a larger number ($\alpha = 30$). It is a simple multi-objective problem to minimise distance and capacity violation.

6.2.3 The Combinatorial Bees Algorithm for VRP

In this section, the main element of the proposed CBA's version will be explained. The basic BA used as a benchmark foundation has a similar procedure compared to the CBA for TSP (see Figure 6.9). Two novel strategies implemented on the proposed version. First, the proposed version of CBA used Clustering Bi-Parameters BA in the initialisation phase and the abandonment strategy (global solution). The Clustering method is classified the consumer to the number of the fleet, and every cluster can not exceed the vehicle's capacity. The initialisation serves as the initial solution, while the abandonment phase re-initialises the cluster when stagnated. Moreover, second, the neighbour search mechanism implemented the basic neighbour search mechanism when the neighbour search trap in the local optima, the novel strategy of exploitation, "Bees Routing Optimiser", will try to escape from the trap (see Figure 6.10). The parameter configuration is seen in Table 6.4.

As the random permutation is used on the basic CBA's initialisation, the proposed version

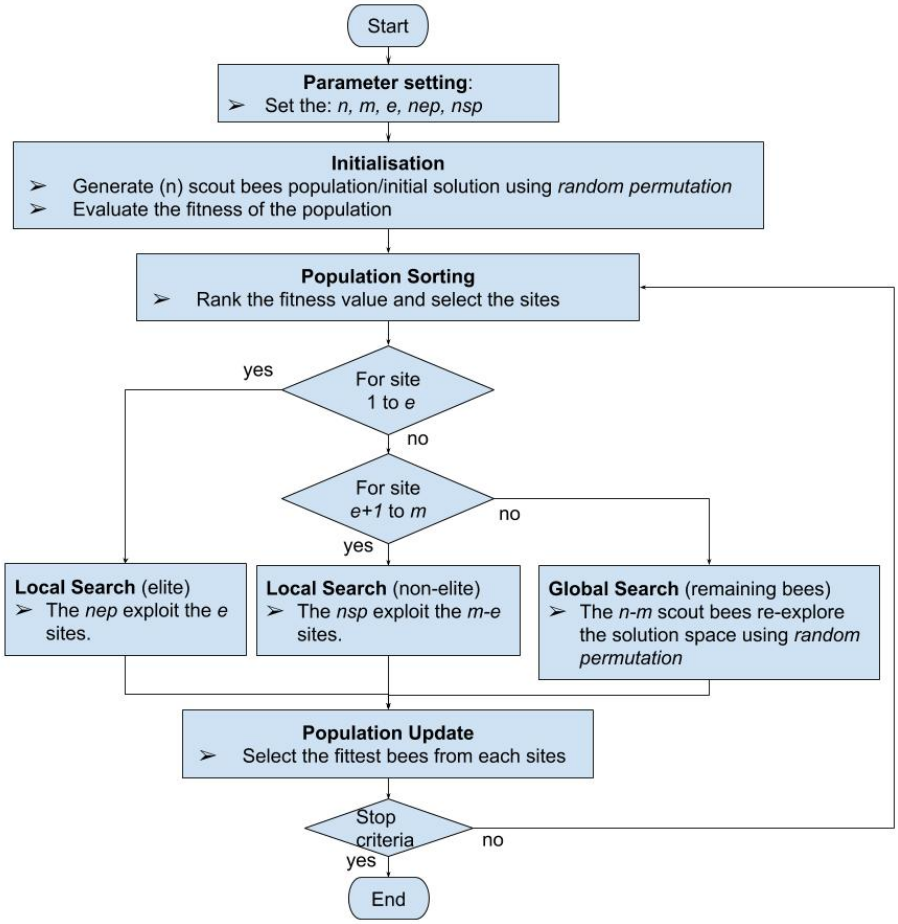


Figure 6.9: CBA for Vehicle Routing Problem

using the customer clustering based on the number of the vehicle using Bi-Parameter BA. Each cluster can not exceed the capacity of the vehicle. The clustering works by minimising the total distance to the centroid and the capacity violation. The clustering starts from n scout bees, and each initial solution comprises the k random centroids. Moreover, the local search will move the centroid, which will produce the minimum fitness.

After constructing the initial solution, the promising initial solution was exploited. The exploitation sites are classified as m selected and e elite. The neighbourhood search operator implements a simplified variant of the local search operator, combining swap-insertion-reversion with a frequency of 33% for the CBA (Ismail *et al.*, 2020). The BRO is placed on the abandonment phase proposed version. The exploration in the global search proposed version keeps using a random permutation generator.

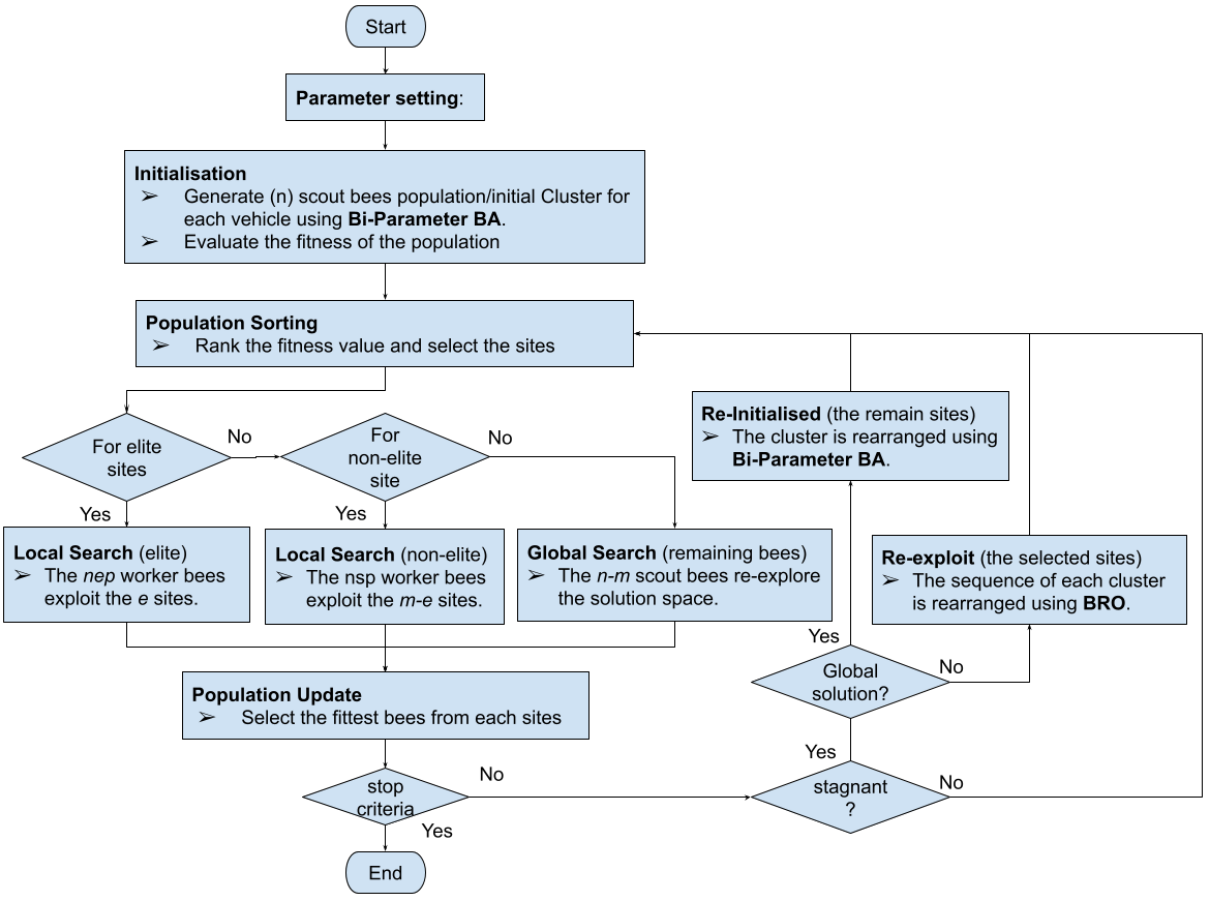


Figure 6.10: Combinatorial Bees Algorithm using Bi Parameter BA Clustering and BRO for Vehicle Routing Problem

Initialisation population

The first step before running the CBA is setting the parameters. The parameter setting followed the previous section (Table 4.3) with colony size ten times the problem's dimension. Moreover, the Bi-BA for clustering used the parameter setting which colony size equal to problem dimensions (example: 50 dimensions means n=7, nep=7), and the BRO setting could be seen in Table 6.5. An Initial population of (n) scout bees as the initial solutions is generated using random permutation for the basic version. The proposed version used Clustering Bi-Parameter BA to generated the cluster nodes for each vehicle without a capacity violation. It means the sum of the customer's demand inside the cluster has to below or equal to the vehicle's capacity. In these cases, all vehicle have a similar capacity.

As it can be seen in Figure 6.11, the Bi-BA could cluster the Eil51-VRP instance by min-

Table 6.4: Parameter setting of CBA for Vehicle Routing Problem

Parameter	Value
Number of scout bees (n)	Dimension
Number of elite bees (nep)	Dimension
Number of best bees (nsp)	50% of (nep)
Number of elite sites (e)	5
Number of selected sites (m)	14

Table 6.5: Parameter setting of Bi-BA and BRO for Vehicle Routing Problem

Parameter	value
Number of scout bees (n) $\rightarrow$ Bi-BA	$Dims^{0.5}$
Number of elite bees (nep) $\rightarrow$ Bi-BA	$Dims^{0.5}$
The value of stagnant ($stlim$) $\rightarrow$ BRO	2* number of vehicle
The minimum number of forgotten flowers/nodes ($fn1$) $\rightarrow$ BRO	1
The minimum number of forgotten flowers/nodes ($fn2$) $\rightarrow$ BRO	1

imising the distance to the centroid and the total demand of the cluster. The red cluster member are node number: [9, 10, 11, 16, 21, 30, 34, 38, 39, 49, 50]. The yellow cluster member are node number: [6, 7, 8, 14, 23, 24, 26, 27, 43, 48]. The green cluster member are node number: [13, 18, 19, 25, 40, 41, 42] The blue cluster member are node number: [4, 5, 12, 15, 17, 33, 37, 44, 45, 46, 47]. The purple cluster member are node number: [1, 2, 3, 20, 22, 28, 29, 31, 32, 35, 36]. The red nodes cluster has total demand of all member of 160. The yellow, green, blue, and purple for 154, 148, 160, and 155 respectively. This cluster information will be converted into the sequence using a basic neighbour search operator and if the solution trapped will be retrieved using the exploitation strategy of BRO.

Exploitation, exploration, and its abandonment strategy

The exploitation sites will be m selected sites out of n scout bees to be improved. There are e elite sites to be primarily exploited as the recruited the worker bees (nep) will be bigger than the worker bees (nsp) on the ($m - e$) non-elite sites. The neighbourhood search operator employs the basic form of the local search operator, a swap-insertion-reversion combination (see Algorithm 8 in Ch.2). When the local solution is trapped, the abandonment strategy using BRO will generate a local solution using BRO.

The remaining ($n - m$) bees will explore the solution space using random permutation for

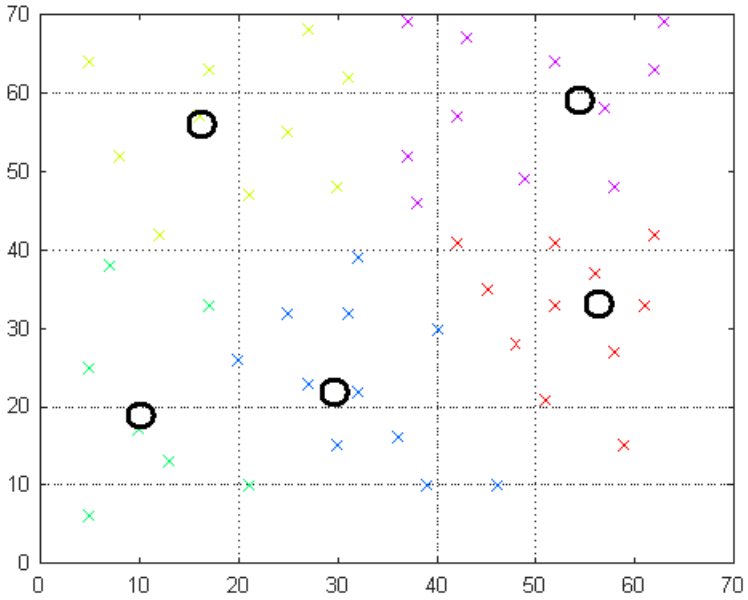


Figure 6.11: Clustering 51 Customer (Eil51) into 5 clusters with total demand below or equal to 160 for each cluster

the basic version of CBA. For the proposed version, the algorithm will explore the solution space using BRO. The BRO will explore for a random cluster. As the method perturbs the cluster, so in this study sets the threshold of stagnant ($stlim$) equal to two times the number of clusters (number of cluster = number of vehicle). It set to two since the algorithm has to perturb each cluster and set to be equal to the number of the cluster does not guarantee to meet all cluster since the algorithm used a random number to choose the cluster. When the global solution trapped, the abandonment strategy will re-initialise all the clusters by using Bi-BA. It used the same method between the clustering method in initialisation and the global search's abandonment strategy phase.

6.2.4 The experiment results

CBA basic version and CBA BRO with clustering strategy have been tested to solve 8 VRPs' datasets of TSPLIB (Reinelt, 1994). All CBA's version for solving TSP in this experiment was coded under Matlab. The code is provided in the appendix section.

The two different versions of CBA, the basic and the proposed version, have been compared to measure the improvement of the proposed algorithm. Both versions used the same neighbour-

Table 6.6: The best result of CBA + Bi-BA + BRO for VRP (Eil33)

Eil33	Sequence															Distance	Capacity
vehicle-1	0	29	28	16	27	26	15	1								247	7950
vehicle-2	0	4	7	9	8	32	11	12	2							167	7850
vehicle-3	0	13	17	25	24	23	22	20	21	19	18	10	6	5	3	265	7770
vehicle-4	0	31	14	30												156	5800
Length																835	
NFE																82391	

Table 6.7: The best result of CBA + Bi-BA + BRO for VRP (Eil51)

Eil51	Sequence												Distance	Capacity
vehicle-1	0	18	13	41	40	19	42	17	4	47			109	157
vehicle-2	0	38	9	30	34	50	16	21	29	2	11		99	159
vehicle-3	0	27	48	23	7	43	24	25	14	6			97	152
vehicle-4	0	32	1	22	20	35	36	3	28	31	26	8	117	149
vehicle-5	0	12	37	44	15	45	33	39	10	49	5	46	99	160
Length													521	
NFE													17,035	

hood search operator using Algorithm 8. Each version was run ten times, and each time was run 2,000 iterations or approximately equal to the number of evaluations of 2,000 multiplied by the colony size.

Before analysed the comparison results, the validation should be conducted. It has to ensure that the model is valid, measuring the total distance of all vehicle, and there is no capacity violation. The feasible solution is the solution without the violation. Although the capacity constraint is a soft constraint, the violation could be avoided by setting the α inside Equation 6.6 to a very large number. The α could be adjusted if we want to have tolerance in the violation.

The table 6.6 to 6.13 demonstrated the simulation of the best solution of every benchmark

Table 6.8: The best result of CBA+ Bi-BA + BRO for VRP (EilA76)

A76	Sequence																Distance	Capacity
vehicle-1	0	53	26	4	25	55	54	24	29	80	68	77	76	12	28		107	192
vehicle-2	0	21	72	75	56	39	67	23	41	22	74	73	40				104	194
vehicle-3	0	7	48	47	46	36	49	64	63	11	19	62					141	171
vehicle-4	0	89	18	82	8	45	17	84	83	60	5	93	59	99	96	6	96	190
vehicle-5	0	13	94	95	92	97	87	42	43	15	57	2	58				83	162
vehicle-6	0	37	98	100	91	44	14	38	86	16	61	85					101	198
vehicle-7	0	3	79	78	34	35	71	65	66	20	51	9	81	33			135	193
vehicle-8	0	52	88	31	10	90	32	30	70	50	1	69	27				94	158
Length																	861	
NFE																	4,497	

Table 6.9: The best result of CBA+ Bi-BA + BRO for VRP (EilB76)

B76	Sequence								Distance	Capacity
vehicle-1	0	40	50	18	24	49			84	100
vehicle-2	0	19	54	13	57	15	29	45	81	99
vehicle-3	0	34	52	27	8	46			45	98
vehicle-4	0	35	14	59	65	38			92	98
vehicle-5	0	36	71	60	70	20	37	5	93	96
vehicle-6	0	75	4	67	26				27	98
vehicle-7	0	10	31	55	25	32			104	100
vehicle-8	0	33	63	23	16	3	44		79	91
vehicle-9	0	53	66	11					74	96
vehicle-10	0	17	51	6	2	30			50	99
vehicle-11	0	7	58	72	39	9	12		66	98
vehicle-12	0	56	41	64	42	43			105	98
vehicle-13	0	62	28	22	1	73	68		73	93
vehicle-14	0	48	47	69	61	21	74		91	100
Length									1064	
NFE									81194	

Table 6.10: The best result of CBA + Bi-BA + BRO for VRP (EilC76)

C76	Sequence														Distance	Capacity
vehicle-1	0	51	3	44	32	50	18	24	49	16	6				93	173
vehicle-2	0	17	40	12	26	67	4								48	147
vehicle-3	0	45	27	52	13	54	19	8	46	34					75	162
vehicle-4	0	35	14	59	66	11	53	7							92	176
vehicle-5	0	2	62	28	22	61	21	74	30	75					90	180
vehicle-6	0	48	47	36	69	71	60	70	20	37	57	15	5	29	120	178
vehicle-7	0	68	73	1	43	42	64	41	56	23	63	33			108	176
vehicle-8	0	58	38	65	10	31	55	25	9	39	72				129	172
Length															755	
NFE															98,034	

Table 6.11: The best result of CBA + Bi-BA + BRO for VRP (EilD76)

D76	Sequence															Distance	Capacity
vehicle-1	0	30	48	21	61	28	74	2	68	75						78	180
vehicle-2	0	46	8	19	54	13	27	52	34	67						66	171
vehicle-3	0	40	9	39	31	10	58	72	12	26						91	185
vehicle-4	0	38	65	11	66	59	14	53	35	7						105	209
vehicle-5	0	57	15	37	20	70	60	71	69	36	47	5	29	45	4	116	209
vehicle-6	0	62	73	1	22	64	42	43	41	56	23	63	33	6		123	215
vehicle-7	0	51	16	49	24	18	55	25	50	32	44	3	17			116	195
Length																695	
NFE																211,439	

Table 6.12: The best result of CBA + Bi-BA + BRO for VRP (A101)

A101	Sequence																Distance	Capacity
vehicle-1	0	31	10	62	11	19	48	7	88	52							87	150
vehicle-2	0	50	33	81	78	34	35	65	71	66	20	9	51	1			133	180
vehicle-3	0	4	55	25	39	67	23	56	75	41	22	74	72				120	192
vehicle-4	0	96	85	16	86	38	14	44	91	100	37	98	92				99	198
vehicle-5	0	28	76	77	3	79	29	24	54	80	68	12	26				87	187
vehicle-6	0	13	97	87	42	43	15	57	2	73	21	40	58	53			89	156
vehicle-7	0	27	69	70	30	32	90	63	64	49	36	47	46	45	8	82	143	197
vehicle-8	0	89	18	83	60	5	84	17	61	93	99	59	95	94	6		83	198
Length																	841	
NFE																	218,628	

Table 6.13: The best result of CBA + Bi-BA + BRO for VRP (B101)

B101	Sequence												Distance	Capacity
vehicle-1	0	69	1	70	30	32	10	31					74	108
vehicle-2	0	53	58	26	28								33	65
vehicle-3	0	14	38	44	86	16	91						99	109
vehicle-4	0	4	39	23	67	25	55						104	112
vehicle-5	0	87	97	92	98	37	100	42	43	15	57	2	87	109
vehicle-6	0	8	45	17	84	5	60	83	89	27			85	105
vehicle-7	0	18	82	46	36	47	48	7	52				91	111
vehicle-8	0	93	85	61	99	96	6						52	99
vehicle-9	0	50	51	9	71	65	66	20					107	108
vehicle-10	0	12	54	24	29	80	68	76					76	104
vehicle-11	0	40	21	72	56	75	22	41	74	7			72	109
vehicle-12	0	3	33	81	35	34	78	79	77				93	112
vehicle-13	0	90	63	11	64	49	19	62	88				116	109
vehicle-14	0	94	59	95	13								38	98
Length													1127	
NFE													172,219	

Table 6.14: Comparison results of VRP instances

Dataset names	k	f	vehicle capacity	BKS	CBA (Ismail *et al.*, 2021) *Best*	*Avg*	*Best* Err.	*Avg* Err.	CBA+Bi-BA+BRO *Best*	*Avg*	*Best* Err.	*Avg* Err.
Eil33	32	4	8,000	835	835	840.00	0.000%	0.599%	835	840.40	0.000%	0.647%
Eil51	50	5	160	521	540	556.80	3.647%	6.871%	521	526.80	0.000%	1.113%
EilA76	75	14	140	826	902	933.60	9.201%	13.027%	861	869.30	4.237%	5.242%
EilB76	75	14	100	1,019	1,111	1,141.30	9.028%	12.002%	1,064	1,092.20	4.416%	7.184%
EilC76	75	8	180	735	770	793.20	4.762%	7.918%	755	764.30	2.721%	3.986%
EilD76	75	7	220	682	702	711.50	2.933%	4.326%	695	710.00	1.906%	4.106%
EilA101	100	8	200	817	864	882.40	5.753%	8.005%	841	849.20	2.938%	3.941%
EilB101	100	14	112	1,077	1,170	1,187.70	8.635%	10.279%	1,127	1,138.60	4.643%	5.720%

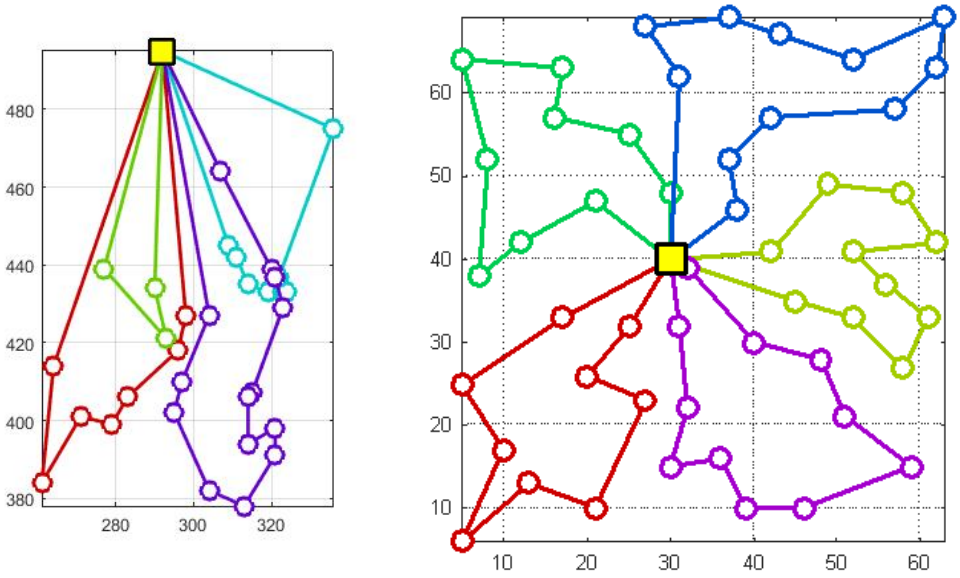
datasets. All the sequences start from vertex-0 (depot), make a tour, and return to the depot. The total length of each dataset is the sum of all vehicles' distance. Those tables show that all vehicles upload goods within their capacity to deliver them to each customer. Every vehicle might have a different number of visited customers. For example, each vehicle on the Eil33 dataset carried no more than its maximum capacity of 8,000. This simulation proves that the built model is valid for solving CVRP.

Table 6.14 reports the accuracy performance of all compared algorithms. The first six columns detail the instance's primary characteristics, including its name, problem type, number of cities, the number of the vehicle (V), vehicle capacity, and best-known solution (BKS). Two serials in the following four columns have a common arrangement. The following four columns detail the best solution obtained ($Best$), the average of all solutions obtained (Avg), the error of the best solution obtained relative to BKS ($Best - Error$), and the average error solution obtained relative to BKS ($Avg - Error$). It can be seen that the accuracy of basic CBA could reach 13.027% up to 0.59% average error toward the best-known solution. While the proposed method 0.64% to 7.184%.

We found that the basic version was effective only in small datasets (below 50 dimensions). The basic CBA has similar performance with the proposed version only at Eil33. The proposed algorithm could perform better both in Average and Best Error on all datasets except Eil33. The

basic CBA has the average of *Best Error*, and *Avg Error* for 5.495% and 7.878%, while the proposed 2.754% and 4.040%.

We found another pattern that influences complexity. The accuracy is depended on the number of the fleet (vehicles). The larger the number of the fleet, the bigger the error will be. The datasets with a larger number of the fleet -like EilA76, EilB76, EilB101 (75 and 100 dimensions with 14 vehicles)- have bigger error rate compared to the datasets with the same dimension but has smaller fleet -EilC76, EilD76, EilA101 (75 and 100 dimensions with 7 or 8 vehicles). The deviation could be noticed when running the clustering phase. The smaller fleet datasets were easier to cluster that meets the non-capacity violation.

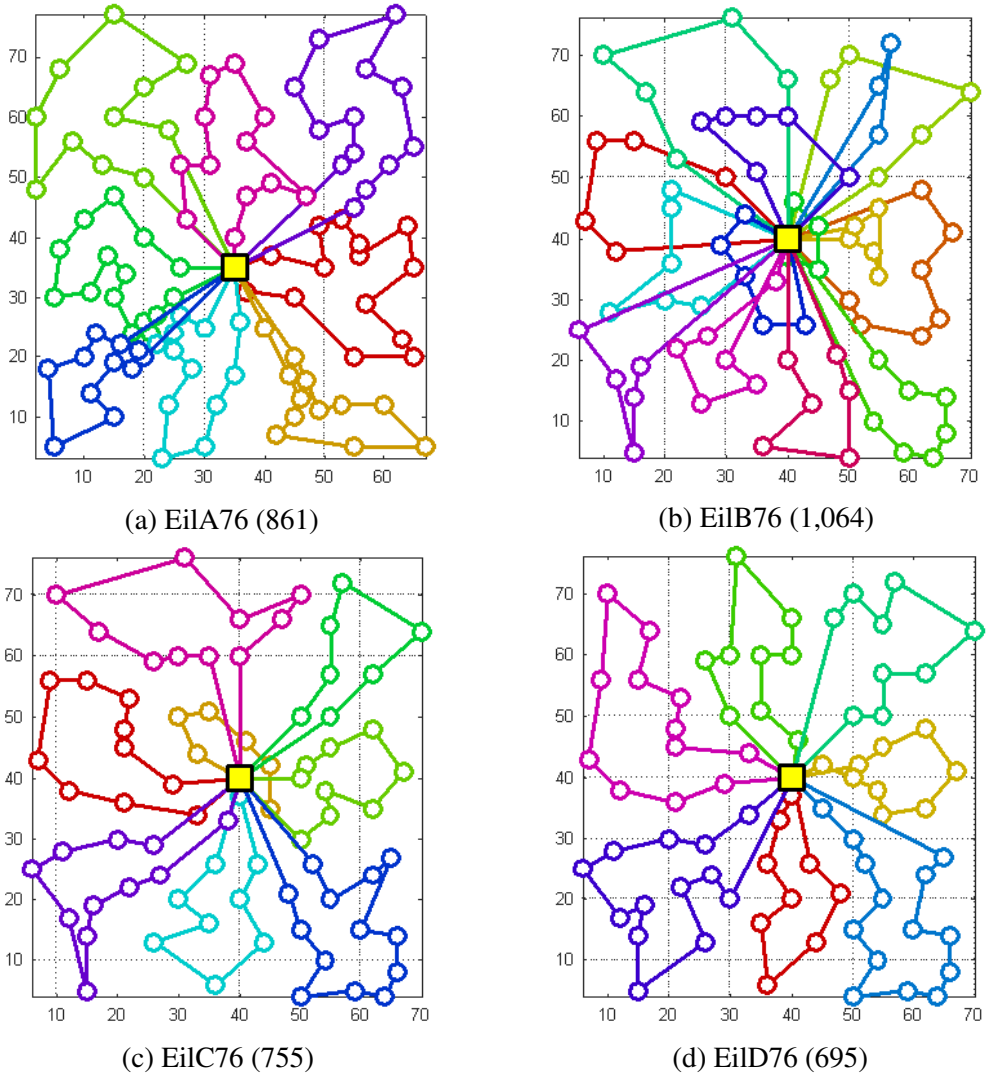


(a) Eil31 (835)

(b) Eil51 (521)

Figure 6.12: Result of the best solution of CBA+Bi-BA+BRO for [33,51] VRPs' instances

Figure 6.12 to 6.14 provide the best-found tour results in graph of the proposed version. For VRP graphical results, these figures used a different line colour to differentiate among the vehicles' paths. The graphical results were the living evidence that provided solutions are the feasible solutions (no capacity violation). The graph of datasets with small error (Eil33, Eil51, EilD76, EilA101) shows that the vehicles' route is clearly separated, or there is no crossing route between them. The datasets with bigger error (EilA76 and EilB76) have more crossing routes than the small ones. So it will be a potential future work if we can develop a clustering approach that can separate datasets with a more extensive fleet without overlapping nodes.



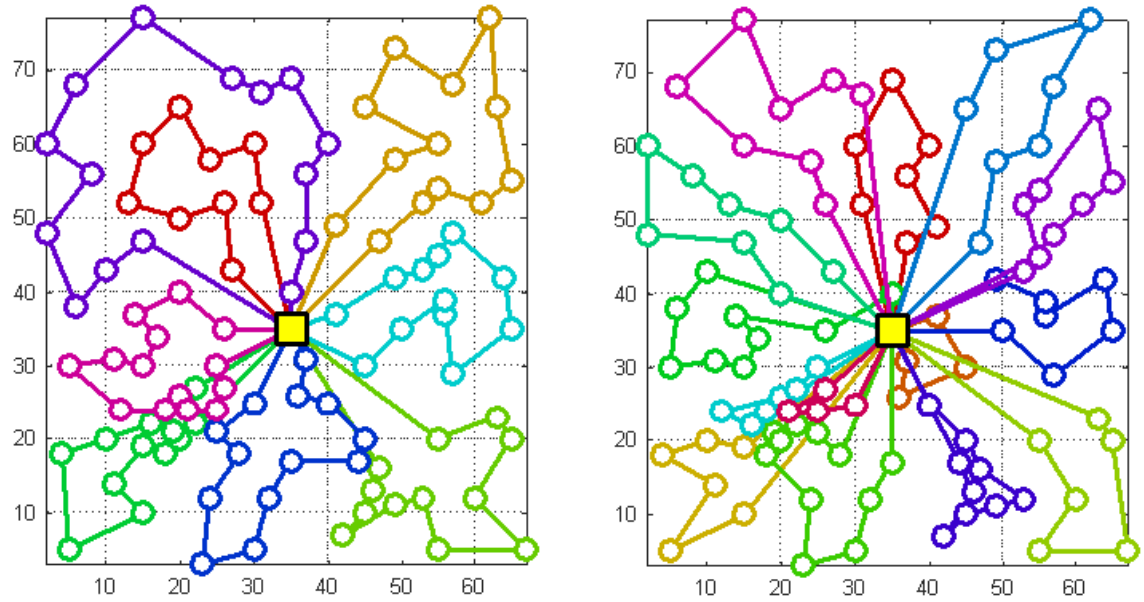
(a) EilA76 (861)

(b) EilB76 (1,064)

(c) EilC76 (755)

(d) EilD76 (695)

Figure 6.13: Result of the best solution of CBA+Bi-BA+BRO for 75 dimension VRPs' instances



(a) EilA101 (841) (b) EilB101 (1,127)

Figure 6.14: Result of the best solution of CBA+Bi-BA+BRO for 100 dimension VRPs' instances

6.3 Summary

The first part of this chapter discussed implementing two versions (seed and non-seed) of the CBA+ BNSN to PCB assembly optimisation (MBTD problem). The proposed technique with seed achieved total assembly time in 23 seconds and an average solution in 25.26 seconds. At the same time, another technique (non-seed) achieved a solution in 24.42 seconds and an average solution in 26.34 seconds. These findings are compared with the current state of the art in the literature. It is worth noting that, in contrast to many recent cases in the literature, the proposed algorithm made use of both seed and non-seed elements of the feeder arrangement. Seeding is likely to accelerate the search process and could also assist the algorithm in discovering better solutions. As a result, it can be inferred that the suggested CBA+BNSN using seeds is capable of finding better solutions at a lower computational expense.

The second section of this chapter presented two versions of CBA for solving CVRP. The basic version of CBA and the new variant of CBA (CBA with Bi-Parameter BA Clustering in the Initialisation phase and the Bees Routing Optimiser on the abandonment phase). The proposed algorithm (CBA+Bi-BA+BRO) could improve the basic version's accuracy by around

2.741% and 3.838% of the mean of $Best - Error$ and $Average - Error$. The convergence performance and the better clustering method could be a critical part that can enrich this study. The clustering method has a critical role in meeting the robust solution of the VRP. A powerful clustering method could improve the solution of VRP.

Chapter 7

Conclusion, contributions and future work

This chapter summarises the findings and contributions of this study. Additionally, it makes recommendations for potential further work.

7.1 Conclusion

Three research objectives have been identified for this study: (i) develop new Bees Algorithm version with better performance -simple, accurate, and faster- using nature as the source of ideas, (ii) evaluate and validate the proposed versions using standard benchmark functions and compare the observed results to those of other algorithms, and (iii) apply the proposed versions to select engineering applications.

The first three chapters following the literature review described three modifications to the Bees Algorithm that enhanced its simplicity, precision, and convergence. The first enhancement to the algorithm, called Bi-parameter BA, is the reduction of initial parameters by integrating exploration and exploitation searches. From the most promising to the least promising, each patch possesses both exploratory and exploitative capabilities with different searching intention. The triangular random number controls the intention itself. The regulator may appoint additional bees to a nearby neighbouring site that appears promising and vice versa. This technique is capable of reducing many parameters, like e, m, nsp, and ngh, since the regulator's assignment differentiates the intention without requiring those parameters. Statistical test revealed

that Bi-BA performs similarly to Basic and Standard Bees Algorithms in terms of precision and convergence for the same standard benchmark functions even though it has fewer initial parameters that need tuning.

The second enhancement to the algorithm, called CBA+BNSN, is a combinatorial BA version with a more suitable initial solution in terms of diverse and initial error balancing. The proposed constructive heuristic mimics the behaviour of bees traplining by taking distance and angle into account while constructing the initial circuit without prior experience. This initial solution generator enables the user to monitor the degree of diversity in the initial solution, ranging from 20% to 90% unique less-error solutions out of n initial solutions. The greater diversity of the initial solution could improve the likelihood of meeting an optimal solution. However, although attached to CBA instances of minimal dimensions (50), the fitness value of CBA with the new generator produces no significantly better result than the random number generator. Additionally, it was discovered that a lower error rate initial solution produced by a constructive heuristic does not guarantee a stronger final solution generated by a metaheuristic. The BNSN with a neutral local search operator is only suitable for small instances up to 150 dimensions. The results showed that CBA needs an intensifier to reach a lower error rate solution for more complex COP (more than 150 dimensions).

The third development, called CBA+BRO, is a combinatorial BA version with a more accurate and convergent final solution. The proposed intensifier mimics bee traplining behaviour by preventing re-visiting such flowers in the context of threats. Threats arise in the form of intruders, rivals, or a lack of resources (nectar depletion). The approach will eliminate threatened flowers from the whole tour and reintroduces them again to the habitual tour's nearest edge. Additionally, since it is located in the abandonment phase of CBA, this intensifier has the potential to escape from local optima traps. The CBA enhanced with this intensifier may be able to compete with other highly effective nature-inspired algorithms. However, since the trial used a lower limit number of evaluations, this latest implementation seems to be less optimal than other algorithms. For 15 datasets with measurements varying from 52 to 1577, the CBA+BNSN+BRO could achieve an error of the best solution obtained of 0.08 per cent.

CBA+BNSN+BRO's success is dependent upon achieving an equilibrium between exploration and exploitation mechanisms. The BNSN could provide a more diverse initial solution to the CBA, while the BRO could search exhaustively for a good initial solution.

The final chapter of the thesis explores how these CBA improvements can be used in engineering applications. The chosen applications are optimisation of the sequence of PCB assembly and the routes for capacitated vehicles. For PCB optimisation, the new CBA version could result in a faster assembly time of 23 seconds, beating the previous study's best solution of 23.46 seconds (Castellani 2019). For VRP, the latest CBA version was introduced to TSPLIB datasets for the first time, and it was found to increase simple CBA accuracy by nearly 4 per cent.

7.2 Contributions

This thesis has detailed three contributions represented by three new versions of the Bees Algorithm (BA). All the versions are inspired by bees' traplining behaviour when solving their own routing problem. The variants are listed below:

- The development of Bi-BA led to the establishment of a simple Bees Algorithm with just two initial parameter settings. This simplicity can assist the novice user in adequately understanding the algorithm and also assist the experienced user in fine-tuning the algorithm.

- The development of CBA+BNSN allows the algorithm to explore a broader solution space, thus increasing the probability of meeting the optimal solution.

- BRO's development enables the algorithm to leverage the good initial solution developed by BNSN more profoundly. It provides an effective combination of diversification and intensification.

- The implementation of CBA for TSP using TSPLIB would add value to the CBA works. The library is used extensively as the standard test function examining the combinatorial

metaheuristic algorithm. Moreover, using TSPLIB for VRP would add value to the CBA work since it is the first VRP to do so.

- The programmes in this study are created using Matlab and Python, and they are all available on GitHub: www.github.com/asrulharunismail, so the reader could easily replicate the work.

7.3 Future work

In the future, it will be important to improve the algorithm's performance and practical purpose of the algorithm using broader datasets (more than 3,000 cities). These datasets will accurately represent the real-world complexities of the problem. Other combinatorial applications would also enhance the CBA's stability, such as VRP and its derivatives, assembly and disassembly sequences, and scheduling. Other well-known constructive heuristics, such as the nearest insertion, the farthest insertion, and the cheapest insertion, may be used to round out the analysis.

Although the basic BA is implemented for numerical problems, the underlying theory applies as well to combinatorial problems. As a result, it can boost the continuous BA as well by adapting to these combinatorial developments. Numerous studies demonstrate that CBA has a competitive performance compared to several population-based metaheuristic algorithms, especially in terms of exploration and exploitation strategies, due to its unique mechanism. It would be exciting to adjust the Bi-parameters BA to a combinatorial form and the CBA+BNSN+BRO to a continuous form.

Developing a mechanism for unique neighbour search in the combinatorial bees algorithm that does not rely primarily on genetic operations may prove challenging in the future. Indeed, the bees' angle-sensitive traplining behaviour has the potential to result in the creation of a novel local search operator.

Additionally, we can explore more from this rich behaviour of foraging bees that involves many species, including bees, flowers, and competitors or predators. This relationship may lead to another conceptual model of cooperative and competitive foraging behaviour in bees.

Appendix A

The data of Chapter 2

Table A.1: The Applications of Bees Algorithm

Applications	Types of Bees Algorithms used and References
1. Aerospace Engineering	
- Aircraft Landing Problem	**CBA**: Abdul-Razaq and Ali (2015)
2. Autonomous System, Control Engineering, and Robotics	
- Control Strategy Testing (Inverted Pendulum)	**BA**: Sen and Kalyoncu (2016), Bilgic *et al.* (2016), Dagher Al-Khwarizi and Ibraheem Abdulkareem (2016); **NN+BA**: Metni and Lahoud (2013); **BA+GA**: Sabah Al-Araji (2016); **BA+CA**: Al-Araji (2019)
- Controller Optimisation	**BA**: Farhang and Mazlumi (2014); **NN+BA**: Satheesh and Manigandan (2013); **BA+Steepest-Descent**: Alfi and Khosravi (2012).
- Fuzzy Controller in the Robotic System	**BA**: Pham *et al.* (2007e), Pham and Kalyoncu (2009), Pham *et al.* (2009a), Zaeri *et al.* (2011), Chamazi and Motameni (2019); **BA+Kalman-Filter**: Pham and Darwish (2009)
- Mapping or Perception	**CBA**: Mazitov *et al.* (2016)
- Motion, Trajectory, Path Planning	**CBA**: Darwish *et al.* (2018), Sabri *et al.* (2018), Ang *et al.* (2009a), Masajedi *et al.* (2013)
- Optimal Control Problem	**BA**: Konstantinov *et al.* (2019a)
- PI/PID Controller	**BA**: Jones and Bouffet (2008), Pham *et al.* (2008f), Amirinejad *et al.* (2014), Danaei and Khajezadeh (2015), Sen and Kalyoncu (2015), Bakırcıoğlu *et al.* (2016), Toloei *et al.* (2014), Ercin and Coban (2011), Arif Şen *et al.* (2016); **MOBA**: ÇOBAN and ERÇİN (2012); **NN+BA**: Aalizadeh and Asnafi (2016); **BA+PSO**: Al-Araji (2017); **BA+FA**: Hameed *et al.* (2019)
- Robotic Arm, Manipulator and Control	**BA**: Eldukhri and Kamil (2013), Hadi *et al.* (2014), Pham *et al.* (2014), Hadi *et al.* (2015), Pham *et al.* (2018); **NN+BA**: Pham *et al.* (2008d), Fahmy *et al.* (2012)
- Biped Robotic Control	**BA**: Yazdi *et al.* (2011), Yazdi *et al.* (2010); **BA+DE**: Massah *et al.* (2013)
- Swarm Robots	**BA**: Jevtic *et al.* (2010); **Distributed BA**: Jevtic *et al.* (2011)
- Three Tank System Control	**BA**: Sarailoo *et al.* (2015)

Table A.1 continued from previous page

Applications	Types of Bees Algorithms used and References
3. Bio Informatics	
- Metabolite Engineering	**CBA+Flux-Balance-Analysis**: Choon *et al.* (2012), Yin *et al.* (2013), Yin *et al.* (2014), Koo *et al.* (2014); **CBA+Hill-Climbing+FBA**: Choon *et al.* (2013b), Choon *et al.* (2013a); **CBA+Hill-Climbing+FBA+OptKnock**: Choon *et al.* (2014b), Choon *et al.* (2014c), Choon *et al.* (2014a), Choon *et al.* (2015).
- Case Prediction	**Mutation based NN+BA**: Saif *et al.* (2021)
- DNA Sequence	**CBA**: Pourkamalianaraki and Sadeghi (2016)
- Gene Regulatory Network	**CBA**: Ruz and Goles (2012), Ruz and Goles (2013), Ruz *et al.* (2014)
- Production of Essential Amino Acids	**CBA+HS**: Aw *et al.* (2018)
- Protein Conformation Search	**CBA**: Bahamish *et al.* (2008), Jana *et al.* (2015)
4. Biomedical Engineering	
- Brain-Computer Interfaces Channel Selection	**Binary CBA**: Martínez-Cagigal and Hornero (2017)
- Cancer Detection	**BA**: Abirami *et al.* (2018); **NN+BA**: Khosravi *et al.* (2011)
- Fermentation Pathway	**BA**: Leong *et al.* (2012)
5. Chemical Engineering	
- Chemical Process Optimisation	**BA**: Pham *et al.* (2008a)
- Dynamic Optimisation	**BA**: Castellani *et al.* (2012)
6. Civil Engineering	
- Ready Mixed Concrete Problem	**CBA**: Wongthatsanekorn and Matheekrieangkrai (2014), Mayteekrieangkrai and Wongthatsanekorn (2015), Mayteekriengkrai and Wongthatsanekorn (2017)
- Structural Control System	**BA**: Arif Şen *et al.* (2018)
- Structural Health Monitoring	**BA**: Krainyukov *et al.* (2014), Dey *et al.* (2019)
- Structure Optimisation	**MOBA**: Moradi *et al.* (2015), **BA+Multi-Setup-Stochastic-Subspace**: Khademi-Zahedi and Alimouri (2019)

Table A.1 continued from previous page

Applications	Types of Bees Algorithms used and References
7. Computer science	
- Image Processing	**BA**: Nebti (2013), Abdelhakim *et al.* (2016), Hussain and Surendran (2020); **BA+Otsu**: Shatnawi *et al.* (2013a); **NN+BA**: Farhan and Bilal (2011), Sabzi *et al.* (2020); **Modified BA**: Hussein *et al.* (2016); **MOBA**: Lee *et al.* (2014); **BA+SVM** Shatnawi (2018)
- Intrusion Detection System	**CBA+ID3**: Eesa *et al.* (2015a)
- Pattern Classification	**NN+BA+Kalman-Filter**: Pham and Darwish (2010)
- Rough Set	**CBA**: Nagy *et al.* (2019)
- Software Effort Estimation	**BA**: Azzeh (2011a), Azzeh (2011b)
- Software Testing	**CBA**: Zabil *et al.* (2012), Zabil *et al.* (2018); **CBA+Interval-based**: Wang *et al.* (2012), Zabil and Zamli (2013), Wang *et al.* (2013)
8. Data Science, Machine Learning, and Deep Learning	
- Data Clustering, and Mining	**BA**: Pham *et al.* (2007i), AbdelHamid *et al.* (2013), Mohammed and Al-Khafaji (2017), Nemmich *et al.* (2018a), **NN+BA**: Pham *et al.* (2007g); BA+Fuzzy-c-mean: Pham *et al.* (2008b); **BA+k-prototypes**: Pham *et al.* (2011); **BA+k-Means**: Mahmuddin and Yusof (2010); **BA+GA+K-means**: Shafia *et al.* (2011), **Pollen based BA**: Bradford Jr and Hung (2012); **BA+PSO**: Dhote *et al.* (2013); **BA+k-Means+HS**: Bonab and Hashim (2014); **B4M Bee for Mining**: Packianather *et al.* (2019); **BA+k-Means+DE**: Bonab *et al.* (2015); **BA Miner**: Tapkan *et al.* (2016b); **BA+Levy**: Shatnawi (2017); **BA+FA**: Songmuang and Luantangsrisuk (2016); **BA+k-Means+Ward**: Kataria and Rupal (2012)
- Data Missing Problem	**CBA**: Sadiq *et al.* (2012)
- Feature Selection	**BA**: Ramlie *et al.* (2016), Mahmuddin and Al-dawoodi (2017), Al-dawoodi and Mahmuddin (2017); **Modified BA**: Ramlie *et al.* (2020); **NN+BA**: Ahmed and Brifcani (2015)
- Regression Machine learning	**BA**: Baronti *et al.* (2020b)
- SVM Optimisation	**BA**: Pham *et al.* (2007h), Samadzadegan and Ferdosi (2012), Samadzadegan and Hasani (2015)

Table A.1 continued from previous page

Applications	Types of Bees Algorithms used and References
- Training Neural Network	**BA**: Ghanbarzadeh (2010), Khanmirzaei (2010); **Enhanced BA**: Marzi *et al.* (2017)
9. Education	
- Automated Test Assembly	**BA**: Luantangsrisuk *et al.* (2017)
10. Electrical, and Electronic Engineering	
- Antenna Design	**BA**: Guney and Onay (2007a), Guney and Onay (2007b), Guney and Onay (2008), Guney and Onay (2010), Guney and Onay (2011), Guney and Onay (2013)
- Device Allocation	**CBA**: Idris *et al.* (2009b) Idris *et al.* (2009a), Idris *et al.* (2010b), Idris *et al.* (2010c), Azman *et al.* (2020); **Paralel BA**: Idris *et al.* (2010a); **MOBA**: Mohamad Idris *et al.* (2009), Mugiira and Ndolo (2016).
- Economic Dispatch	**BA**: Tiacharoen *et al.* (2010), Trongwanichnam *et al.* (2019), Sharma *et al.* (2015), Leitao *et al.* (2018); **MOBA**: Pham *et al.* (2008g), Lee and Darwish (2008), Lee and Oh (2009), Pham *et al.* (2010),Leeprechanon and Polratanasak (2010), Lee and Kim (2010).
- Electronic Device and Circuit Design	**BA**: Pham and Koç (2010), Akkar (2010), Mollabakhshi and Eshghi (2013); **NN+BA**: Akkar (2010); **BA+Fuzzy**: Tiacharoen and Chatchanayuenyong (2019)
- Multilevel-Inverter	**BA**: Kavousi *et al.* (2012); **MOBA**: Falehi (2019)
- Optimal Power Flow	**BA**: Rajagopalan *et al.* (2018), Lenin *et al.* (2014), Manusov *et al.* (2019); **BA+Newton**: Anantasate *et al.* (2010); **Parallel BA**: Polratanasuk *et al.* (2010), Othman *et al.* (2010); **MOBA**: Anantasate and Bhasaputra (2011), Bhasaputra *et al.* (2011); **BA+CA**: Srikun and Sawetsakulanond (2017); **Two Hive BA**: Leeprechanon and Phonrattanasak (2013); **MOBA+Fuzzy**: Tolabi *et al.* (2014a)
- Power System	**BA**: Rashtchi *et al.* (2010), Arzeha *et al.* (2015), Khalid *et al.* (2015), Arzeha *et al.* (2018), Firouzjah (2018); **Fuzzy BA**: Girdenis *et al.* (2019); **NN+BA**: Akamine (2019), Khodsuz and Seyyedbarzegar (2019b), Khodsuz and Seyyedbarzegar (2019a), Ghiasi *et al.* (2019).
- Power Transmission	**CBA+Benders**: MacRae *et al.* (2019)

Table A.1 continued from previous page

Applications	Types of Bees Algorithms used and References
- Renewable Hybrid Energy System Optimisation	**BA**: Tudu *et al.* (2011), Tudu *et al.* (2014), Maleki (2018), Fahmy (2012), Falehi and Rafiee (2018); **MOBA**: Phonrattanasak (2011), Phonrattanasak *et al.* (2013); **NN+BA**: Assareh and Biglari (2016)
- Three Phase Power Transformer	**BA**: Rodríguez *et al.* (2019)
11. Environment Health and Safety	
- Contamination and Toxicology	**BA**: Zarei *et al.* (2014), Ghaedi *et al.* (2015a); **NN+BA**: Zarei *et al.* (2013), Ghaedi *et al.* (2015b), Farajvand *et al.* (2018), Ebrahimpoor *et al.* (2019); **Cross validation BA**: Zarei *et al.* (2017)
- Flood Susceptibility Modeling	**NN+BA**: Tien Bui *et al.* (2018)
- Relief Center Allocation	**CBA**: Saeidian *et al.* (2016)
12. Hydrology	
- Fluid Flow Simulation	**BA**: Mehdinejadiani (2017); **NN+BA**: Zargartalebi *et al.* (2012), Mehdinejadiani *et al.* (2013)
13. Industrial or Manufacturing Engineering	
- Assembly/Disassembly Sequence Planning	**CBA**: Pham *et al.* (2007j),Liu *et al.* (2018a), Castellani *et al.* (2019), Liu *et al.* (2019), Li *et al.* (2019), Liu *et al.* (2020a); **MOBA** Liu *et al.* (2020c), Xu *et al.* (2020), Liu *et al.* (2020b); **CBA-TRIZ**: Ang *et al.* (2009b), Ang *et al.* (2010), Ang *et al.* (2013b), **Ternary CBA**: Laili *et al.* (2019)
- Assignment and Line Balancing	**CBA**: Baykasoğlu *et al.* (2009), Sadiq and Hamad (2010), Bernardino *et al.* (2010), Daoud *et al.* (2012), Tapkan *et al.* (2012a), Akpinar and Baykasoğlu (2014a), Chmiel and Szwed (2016), Tapkan *et al.* (2016a), Çil *et al.* (2020), Özbakir *et al.* (2010); **Modified CBA**: Ozbakir and Tapkan (2011); **Fuzzy MOCBA**: Ozbakir and Tapkan (2010), Tapkan *et al.* (2012b); **Multiple colony CBA**: Tapkan *et al.* (2013b); **MOCBA**: Liu *et al.* (2018b), Xu *et al.* (2011a)
- Cloud Manufacturing	**CBA**: Xu *et al.* (2016b), Tian *et al.* (2013), Xu *et al.* (2016a); **CBA+Forager-adjustment**: Xie *et al.* (2015a)
- Container Loading Problem	**CBA**: Luangpaiboon (2011), Mongkolkosol and Luangpaiboon (2011); **CBA+Heuristic-Filling**: Dereli and Das (2011)

Table A.1 continued from previous page

Applications	Types of Bees Algorithms used and References
- Control Chart	**NN+BA**: Pham *et al.* (2006d), Pham *et al.* (2006a), Pham *et al.* (2006b), Ebrahimzadeh *et al.* (2013), Addeh *et al.* (2018), De la Torre Gutiérrez and Pham (2018), Wong and Chua (2019); **Fuzzy NN+BA**: Addeh and Ebrahimzadeh (2013)
- Facility Layout	**CBA**: Fon and Wong (2010), Li *et al.* (2010a); **CBA+PSO**: Lien and Cheng (2012), Cheng and Lien (2012),Lien and Cheng (2014)
- Fault and Crack Detection	**BA**: Packianather and Kapoor (2015), Packianather *et al.* (2018),González-Islas *et al.* (2011), Moradi *et al.* (2011b), Moradi and Kargozarfard (2013), Hashem *et al.* (2013), Zahedi *et al.* (2017), Almansob *et al.* (2017), Eesa *et al.* (2015b), Rufai *et al.* (2014), Sagayam and Akilandeswari (2012); **BA+SVM**: Xie *et al.* (2018), Alomari and Othman (2012); **NN+BA**: Pham *et al.* (2006e), Pham *et al.* (2007b), Ali and Jantan (2011), Attaran *et al.* (2011), Attaran *et al.* (2012), Attaran and Ghanbarzadeh (2015), Kalami (2014)
- Lot Sizing Problem	**NN+BA**: Şenyiğgit *et al.* (2013); **CBA+Fix-and-Optimise (BFO)**: Furlan and Santos (2017)
- Material Handling Equipment Planning	**CBA**: Sayarshad (2010)
- Multi-Zone Dispatching Systems	**CBA**: Triwate and Luangpaiboon (2010)
- Operational/Production Scheduling	**CBA**: Pham *et al.* (2007c), Pham *et al.* (2007f), Teimoury and Haddad (2013a), Teimoury and Haddad (2013b), Aungkulanon (2016); **CBA+GA**: Packianather *et al.* (2014), Yuce *et al.* (2017); **CBA+SA** Almaneea and Hosny (2018); **CBA+Slope-Angle-Computation+Hill-Climbing**: Yuce *et al.* (2015)
- Project Management (Schedulling)	**CBA**: Sadeghi *et al.* (2011), Oztemel and Selam (2017),Imannezhad and Avakh Darestani (2018), Nemmich *et al.* (2019); **MOCBA**: Sadeghi and Alahyari (2013), Ghasemi *et al.* (2015); **CBA+Forward-Backward-Interchange**: Ziarati *et al.* (2011).
- Quality of Service in Manufacturing	**CBA**: Xu *et al.* (2012)

Table A.1 continued from previous page

Applications	Types of Bees Algorithms used and References
- Supply Chain Network or Facility Location Problem	**CBA**: Xu *et al.* (2010b), Lambiase *et al.* (2016), Gharaei and Jolai (2018), Gharaei and Jolai (2019); **CBA+Mix-Integer-Programming**: Cabrera G. *et al.* (2012), **MO GA+CBA**: Gharaei and Jolai (2021); **MOCBA**: Mastrocinque *et al.* (2013); **CBA+Adaptive-Neighbour-Search**: Yuce *et al.* (2014); **CBA+TS** Martino *et al.* (2016)
14. Information and Communication Technology Engineering	
- Cloud Environment and Data Center	**CBA**: Firdhous *et al.* (2011a), Firdhous *et al.* (2011b), Scionti *et al.* (2019), Keshavarznejad *et al.* (2021); **CBA+ML**: Yuan *et al.* (2020)
- Cloud Gaming Environment	**CBA**: Aboutorabi and Rezvani (2020)
- Cryptanalysis	**CBA**: Ali (2013a), Ali (2013b); **BA+SA**: Ali and Mahmod (2015), Ali *et al.* (2018)
- E-Testing	**BA**: Songmuang and Ueno (2011), Songmuang *et al.* (2012)
- Load Balancing	**CBA**: Bernardino *et al.* (2011)
- Mobile and Adhoc Networks, Peer to Peer Searching	**BA**: Dhurandher *et al.* (2009), Dhurandher *et al.* (2011); **MOCBA**: Sayadi *et al.* (2009)
- Modulation Identification and Classification	**BA**: Sherme (2012), Yang *et al.* (2015), Hakimi and Ebrahimzadeh (2015)
- Multi Input Multi Output Radar	**BA**: Malekzadeh *et al.* (2012)
- Multicast Routing in TCP/IP Communication	**CBA**: Taher and Masoudrahmani (2012)
- Network Optimisation	**CBA**: Saad *et al.* (2008), Moussa and El-Sheimy (2010), Osamy *et al.* (2018), Khalaf *et al.* (2020); **Modified Distributed CBA**: Tkach *et al.* (2018); **CBA+Grasshopper**: Deghbouch and Debbat (2021); **NN+CBA**: Ananthi and Ranganathan (2016)
- Optical Telecommunication Networks Design	**CBA**: Bernardino *et al.* (2012)
- Path or Trajectory Tracking	**NN+BA+PSO**: Al-Araji and Yousif (2017)
- Satellite Based Navigation System	**NN+BA**: Azarbad *et al.* (2014)
- Security Attack	**CBA**: Ramesh (2018)
- Signal Recognition and Separation	**BA**: Ebrahimzadeh and Mavaddati (2014); **NN+BA**: Shrme (2011)

Table A.1 continued from previous page

Applications	Types of Bees Algorithms used and References
- Spectrum Allocation	**BA**: Lu *et al.* (2015)
15. Material Science	
- Electrochemical Discharge Machining	**BA**: Antil *et al.* (2019)
- Material Design Optimisation	**NN**+**BA**: Düenci *et al.* (2015), Ahangarpour *et al.* (2018)
- Metal Forming	**BA**: Ramirez *et al.* (2010), Yaghoubi and Fereshteh-Saniee (2020)
- Welding	**BA**: Vejdannik and Sadr (2017), Hasanvand (2019); **NN+BA**: Vejdannik and Sadr (2018)
16. Mathematical Optimisation	
- Chaotic and Non Chaotic System	**BA**: Gholipour *et al.* (2013), Gholipour *et al.* (2012); **MOBA**: Gholipour *et al.* (2015)
- Dynamic Environment	**Cellular Learning BA**: Khosravy Far and Aghazadeh (2015)
- Fuzzy Measure	**BA**: Wang *et al.* (2011)
- Four Colour Map Problem	**CBA+SA**: Sadiq and Hamad (2010);
- Inverse Parabolic System	**BA**: Mazraeh *et al.* (2013)

Table A.1 continued from previous page

Applications	Types of Bees Algorithms used and References
- Numerical Function Optimisation	**BA**:Karaboga and Akay (2009), Li *et al.* (2010b), Chai-ead *et al.* (2011), Assareh *et al.* (2011), Yuce *et al.* (2013), Pham and Castellani (2014), Hussein *et al.* (2015), Pham and Castellani (2015), Zhou *et al.* (2016); BA-Fuzzy selection Pham and Darwish (2008b); **BA+Levy-Flight**: Hussein *et al.* (2013), Shatnawi *et al.* (2013c), Hussein *et al.* (2014), Hussein *et al.* (2017a); **BA+ABC**: Tsai (2014a); **Multiple colony BA**: Akpinar and Baykasoğlu (2014b); **Parallel BA**: Luo *et al.* (2014), Najm and Hammash (2015); **BA+Self-Adaptive-Neighbour**: Tsai (2014b), Azfanizam *et al.* (2014); **SFL BA**: Nguyen (2015); **Grouped BA**: Nasrinpour *et al.* (2017); **BA+FA**: Gholami and Mohammadi (2018), Nemmich *et al.* (2020b), Nemmich *et al.* (2020a); **BA+Grey-Wolf**: Konstantinov *et al.* (2019b); **BA+Nelder-Mead**: Mahmuddin and Yusof (2009), Kamaruddin *et al.* (2019); **Standard BA**: Pham and Castellani (2009); **Pheromone BA**: Packianather *et al.* (2009), Shirasaki *et al.* (2010), Shirasaki *et al.* (2011); **BA+Dynamic-Cellular-Learning**: Khosravy Far and Aghazadeh (2015); **BA+ LS-Manoeuvre-Recruitment**: Muhamad *et al.* (2011); **BA+FA+VNS**: Aungkulanon and Luangpaiboon (2012), Nemmich *et al.* (2018b), Nemmich *et al.* (2020a); **BA+HS**: Gao *et al.* (2012)
- Parameter Setting	**BA**: Zhang and Cheng (2016), Phan *et al.* (2020)
- Statistical Analysis	**BA**: Baronti *et al.* (2020a)
17. Mechanical Engineering	
- Energy Conversion System	**NN+BA**: Uysal *et al.* (2017)
- Energy and Environment	**BA+ADVISOR**: LONG and NHAN (2012); **NN+BA**: Behrang *et al.* (2011a), Behrang *et al.* (2011b), Xu *et al.* (2011b), Tolabi *et al.* (2013), Naderian (2014); Tolabi *et al.* (2014b), Pham *et al.* (2013), Ahmad and Sunthiram (2018); **MOBA**: Long (2015).

Table A.1 continued from previous page

Applications	Types of Bees Algorithms used and References
- Engineering Design or Mechanical Component	**BA**: Pham *et al.* (2007d), Pham *et al.* (2008e), Pham *et al.* (2009b), Moradi *et al.* (2011a), Aydogdu and AKIN (2011), Nafchi *et al.* (2012), Braiwish *et al.* (2014), Zarea *et al.* (2014), Mirshekari *et al.* (2016), Kashkooli and Nasir (2016), Osman *et al.* (2018), Kamaruddin and Abd Latif (2019), Kazemi *et al.* (2012), Banooni *et al.* (2014), Moradi *et al.* (2014), Karimi *et al.* (2016), Ilka *et al.* (2015), Tandis and Assareh (2017), Dat *et al.* (2020a); **MOBA**: Pham and Ghanbarzadeh (2007), Nafchi and Moradi (2011), Nafchi *et al.* (2011),Zarchi and Attaran (2017), Zarchi and Attaran (2019), Salamat and Ghanbarzadeh (2012), Attaran *et al.* (2017); **BA+Fuzzy**: Attaran and Ghanbarzadeh (2012), Zarchi and Attaran (2019), Zarea *et al.* (2018); **BA+TRIZ**: Ahmad *et al.* (2012); **Memory based BA**: Shatnawi *et al.* (2013b); **BA+GA**: Braiwish *et al.* (2015); **BA+PSO**: Khazaei *et al.* (2015); **BA+HS**: Acar *et al.* (2019)
- Hydraulic Motor	**BA**: Pham *et al.* (2018)
- Deep Drawing	**BA**: Yaghoubi and Fereshteh-Saniee (2020)
- Piping System	**BA**: Moradi *et al.* (2010)
- Transfer Heat Conduction	**BA**: Mizanadl and Ardakani (2012)
- Vibration Analysis	**BA**: Alimouri *et al.* (2017), Alimouri *et al.* (2018), Dat *et al.* (2020b), Anh *et al.* (2021)
18. Meteorology	
- Weather Prediction	**BA+SVM**: Karunakaran *et al.* (2019); **NN+BA+Gradient-Descent**: Khanmirzaei (2010),Khanmirzaei and Teshnehlab (2010).
19. Operations Research	
- Minimum Spanning Tree	**CBA**: Malik (2012)
- Resource or Task Allocation	**CBA**: Archana and Rejith (2014a), Archana and Rejith (2014b), Xie *et al.* (2015b), Tkach and Amador (2021); **Modified Distributed CBA**: Tkach *et al.* (2013); **CBA+ACO**: Phonrattanasak and Leeprechanon (2016), Sharma *et al.* (2017)

Table A.1 continued from previous page

Applications	Types of Bees Algorithms used and References
- Routing Plan (TSP or VRP)	**CBA**: Pham *et al.* (2008h) Masmoudi *et al.* (2016), Alzaqebah *et al.* (2018), Ali and Al Masud (2018), Ismail *et al.* (2020), Jamhuri *et al.* (2020), Leong *et al.* (2020), Rabbani *et al.* (2020), Ismail *et al.* (2021), Zeybek *et al.* (2021); **Mutation based CBA**: Akram Chaweshly (2010), **Exploration balance CBA**: Sadiq AlObaidi and Hamad (2012), **Scatter+CBA**: Sagheer *et al.* (2012), **CBA+VPT**: Zeybek and Koç (2015), Zeybek *et al.* (2021)
- Timetabling	**CBA**: Lara *et al.* (2008), Khang *et al.* (2011), Al-Negheimish *et al.* (2018); **CBA+GA**: Phuc *et al.* (2011a); **CBA+HS**: Nguyen *et al.* (2012); **CBA+SA+Hill-Climbing**: Abdullah and Alzaqebah (2013); **CBA+SA**: Alhuwaishel and Manar (2015)
20. Languange	
- Handwriting	**NN+BA**:Nebti and Boukerram (2010),Nebti and Boukerram (2013)
21. Petroleum Engineering	
- Oil Recovery	**BA**: Siavashi *et al.* (2017)
22. Social Science and Management	
- e-Government	**BA**: Ghodousi *et al.* (2019)
- Management Decision Making Model	**BA**: Paul *et al.* (2014)
- Product Concept, Design, Branding, and Test	**BA**: Pham *et al.* (2008c), Ang *et al.* (2013a), Parsa *et al.* (2013)

Appendix B

The results of Chapter 3

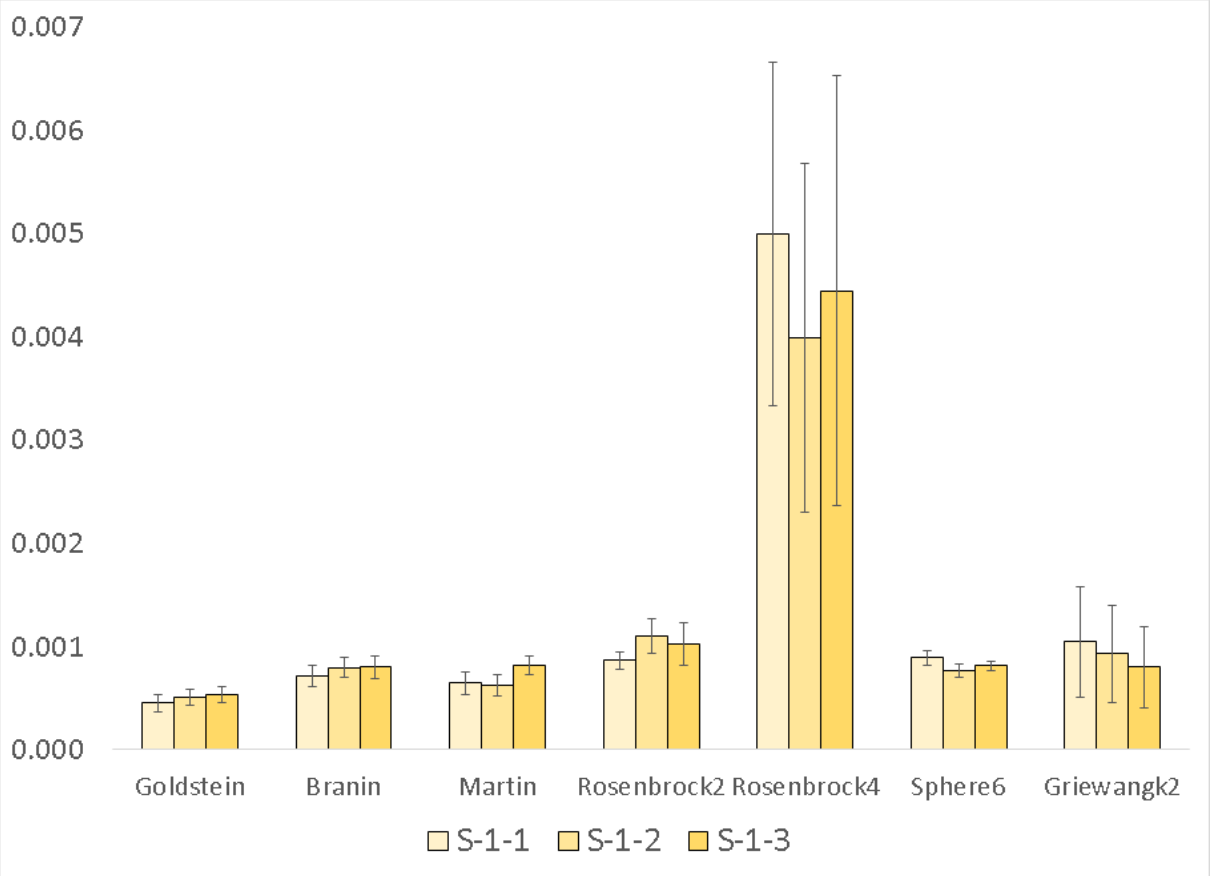


Figure B.1: Global view of Statistical test (95% confidence level) on the accuracy of Bi-BA

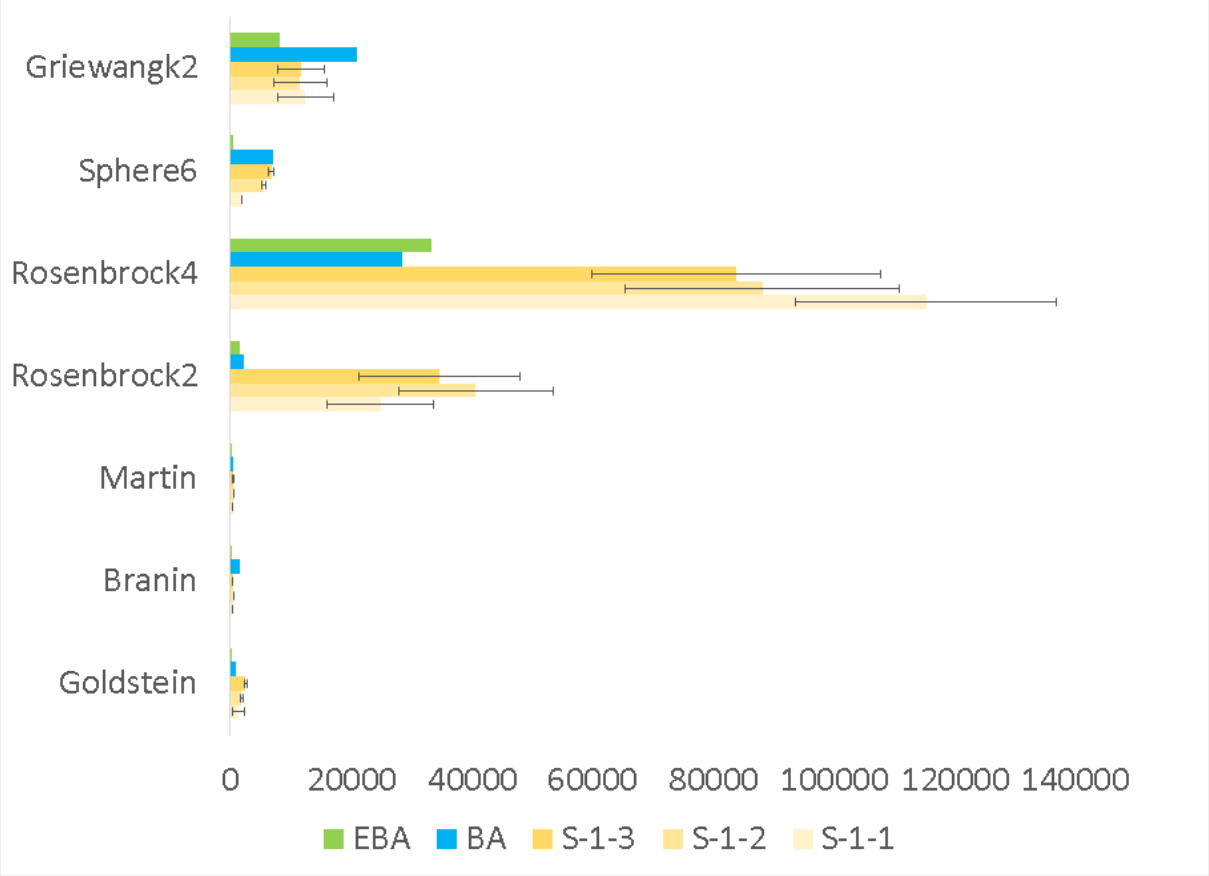


Figure B.2: Global view of Statistical test (95% confidence level) on the number of function evaluations (NFE) of Bi-BA

Table B.1: The result of Bi-BA with Scenario 1-1 (Bi-BA-1)

Function	**Dim**	Solution (Mean)	Solution (StdDev)	Difference (Mean)	Difference (StdDev)	Evaluation (Mean)	Evaluation (StdDev)	Num.of Succed
Goldstein	2	3.000455	0.000288	0.000455	0.000288	1323.56	3717.40	50
Branin	2	0.398447	0.000365	0.000720	0.000365	325.62	88.16	50
Martin	2	0.000648	0.000386	0.000648	0.000386	377.98	137.98	50
Griewangk2	2	0.001048	0.001891	0.001048	0.001891	12445.86	16222.70	46
Rosenbrock2	2	0.000866	0.000303	0.000866	0.000303	24866.32	31070.98	44
Rosenbrock4	4	0.004998	0.005839	0.004998	0.005839	115283.46	76246.16	18
Sphere6	6	0.000892	0.000250	0.000892	0.000250	1863.38	215.89	50

Table B.2: The result of Bi-BA with Scenario 1-2 (Bi-BA-2)

Function	**Dim**	Solution (Mean)	Solution (StdDev)	Difference (Mean)	Difference (StdDev)	Evaluation (Mean)	Evaluation (StdDev)	Num.of Succed
Goldstein	2	3.000510	0.000286	0.000511	0.000286	1868.34	735.48	50
Branin	2	0.398525	0.000330	0.000798	0.000330	559.54	122.6475	50
Martin	2	0.000625	0.000363	0.000625	0.000363	589.82	196.7071	50
Griewangk2	2	0.000932	0.001660	0.000932	0.001660	11547.12	15418.40	47
Rosenbrock2	2	0.001099	0.000595	0.001099	0.000595	40691.92	44807.25	34
Rosenbrock4	4	0.003995	0.005952	0.003995	0.005952	88154.14	79810.93	25
Sphere6	6	0.000765	0.000213	0.000765	0.000213	5518.22	1420.20	50

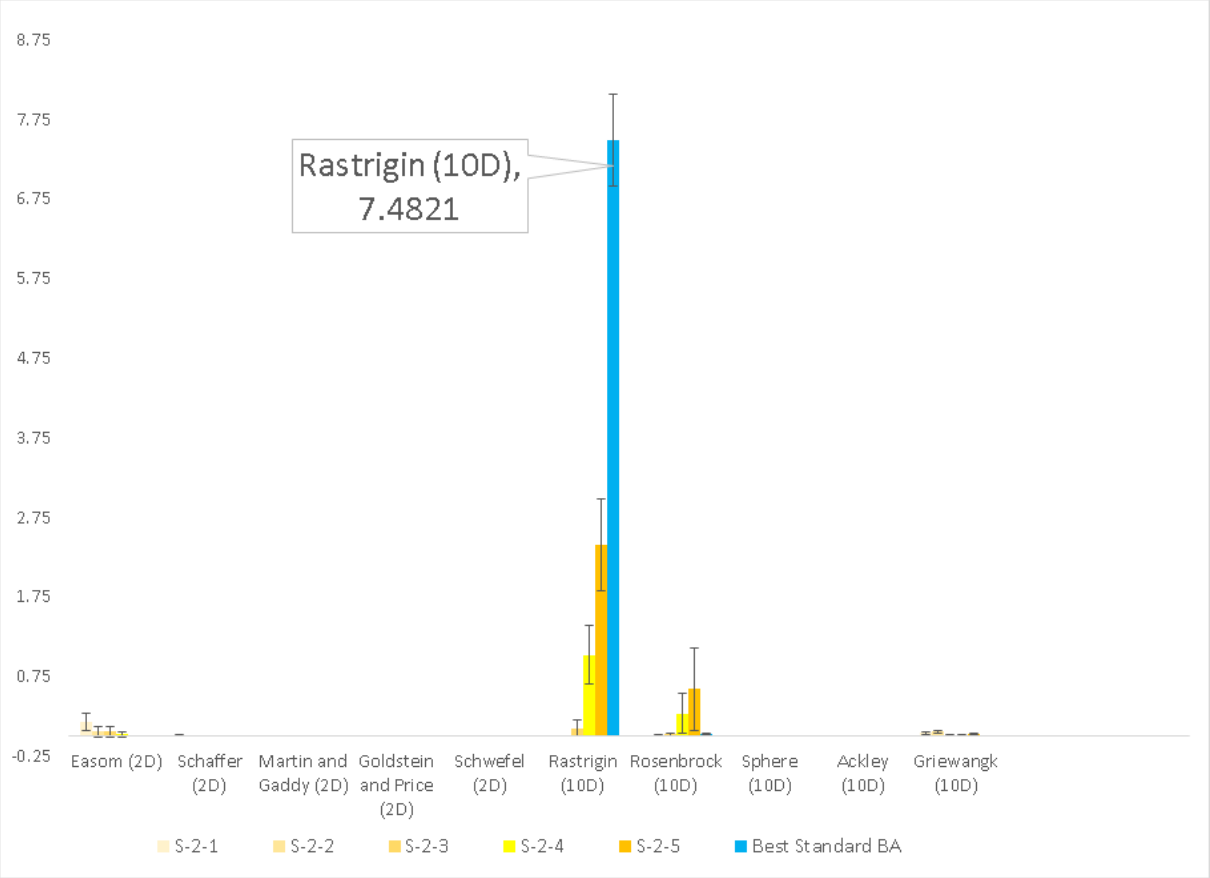


Figure B.3: Comparison of best performance of Bi-SBA Vs SBA (confidence interval (C.I.) = 95%)

Table B.3: The result of Bi-BA with Scenario 1-3 (Bi-BA-3)

Function	**Dim**	Solution (Mean)	Solution (StdDev)	Difference (Mean)	Difference (StdDev)	Evaluation (Mean)	Evaluation (StdDev)	Num.of Succed
Goldstein	2	3.000533	0.000288	0.000533	0.000288	2538.06	1085.74	50
Branin	2	0.398528	0.000377	0.000801	0.000377	350.32	85.31786	50
Martin	2	0.000817	0.000310	0.000817	0.000310	410.78	176.26	50
Griewangk2	2	0.000801	0.001379	0.000802	0.001379	11682.32	13887.66	48
Rosenbrock2	2	0.001024	0.000717	0.001024	0.000717	34643.70	46626.50	37
Rosenbrock4	4	0.004448	0.007323	0.004448	0.007323	83872.70	84208.48	28
Sphere6	6	0.000813	0.000163	0.000813	0.000163	6756.52	1296.06	50

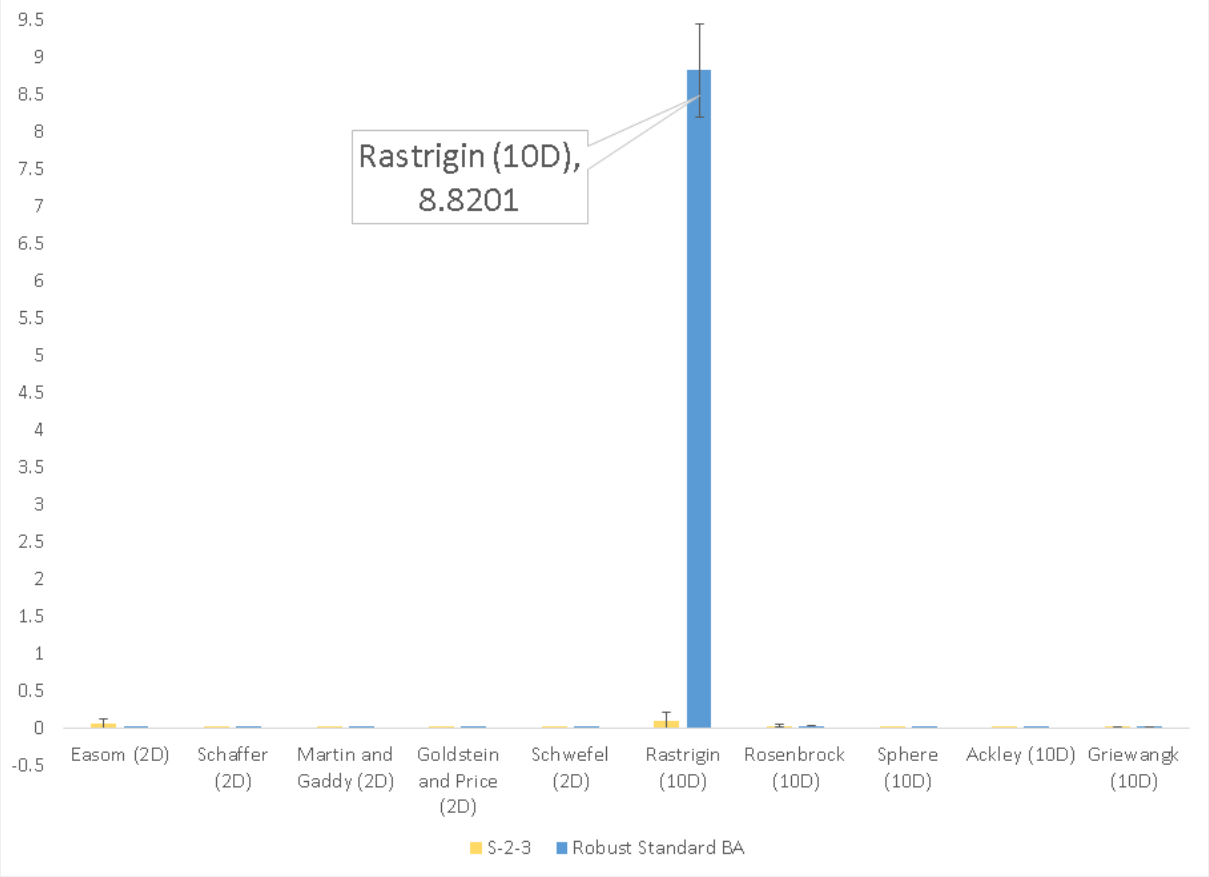


Figure B.4: The Comparison of Robust Performance - Global view (Bi-SBA Vs SBA)

Table B.4: The result of Bi-SBA with Scenario 2-1 (Bi-SBA-1)

Function	Solution (Mean)	Solution (StdDev)	Difference (Mean)	Difference (StdDev)	Evaluations (Mean)	Evaluations (StdDev)	Num.of. Succes
Easom (2D)	-0.819643	0.384016	0.180357	0.384016	12614.38	69608.25	41
Schaffer (2D)	0.008686	0.037225	0.008686	0.037225	33342.48	117969.22	47
Camel six hump (2D)	-1.031170	0.000292	0.000430	0.000292	1283.54	399.66	50
Martin and Gaddy (2D)	0.000518	0.000298	0.000518	0.000298	1360.16	538.59	50
Goldstein and Price (2D)	3.000536	0.000301	0.000536	0.000301	1746.16	455.69	50
Schwefel (2D)	0.000546	0.000278	0.000521	0.000278	3929.12	3625.27	50
Michalewicz (5D)	-4.686996	0.000251	0.000662	0.000251	11110.38	16782.93	50
Trid (6D)	-49.999091	0.000076	0.000909	0.000076	10955.30	2606.54	50
MovedHyper (10D)	0.000802	0.000192	0.000802	0.000192	7885.60	539.84	50
Rastrigin (10D)	0.000733	0.000234	0.000733	0.000234	113016.76	95050.17	50
Rosenbrock (10D)	0.001862	0.001996	0.001862	0.001996	202692.02	206429.79	38
Sphere (10D)	0.000816	0.000155	0.000816	0.000155	5561.06	500.86	50
Ackley (10D)	0.000639	0.000248	0.000639	0.000248	5840.38	645.98	50
Griewangk (10D)	0.041374	0.066952	0.041374	0.066952	245396.56	217774.44	31

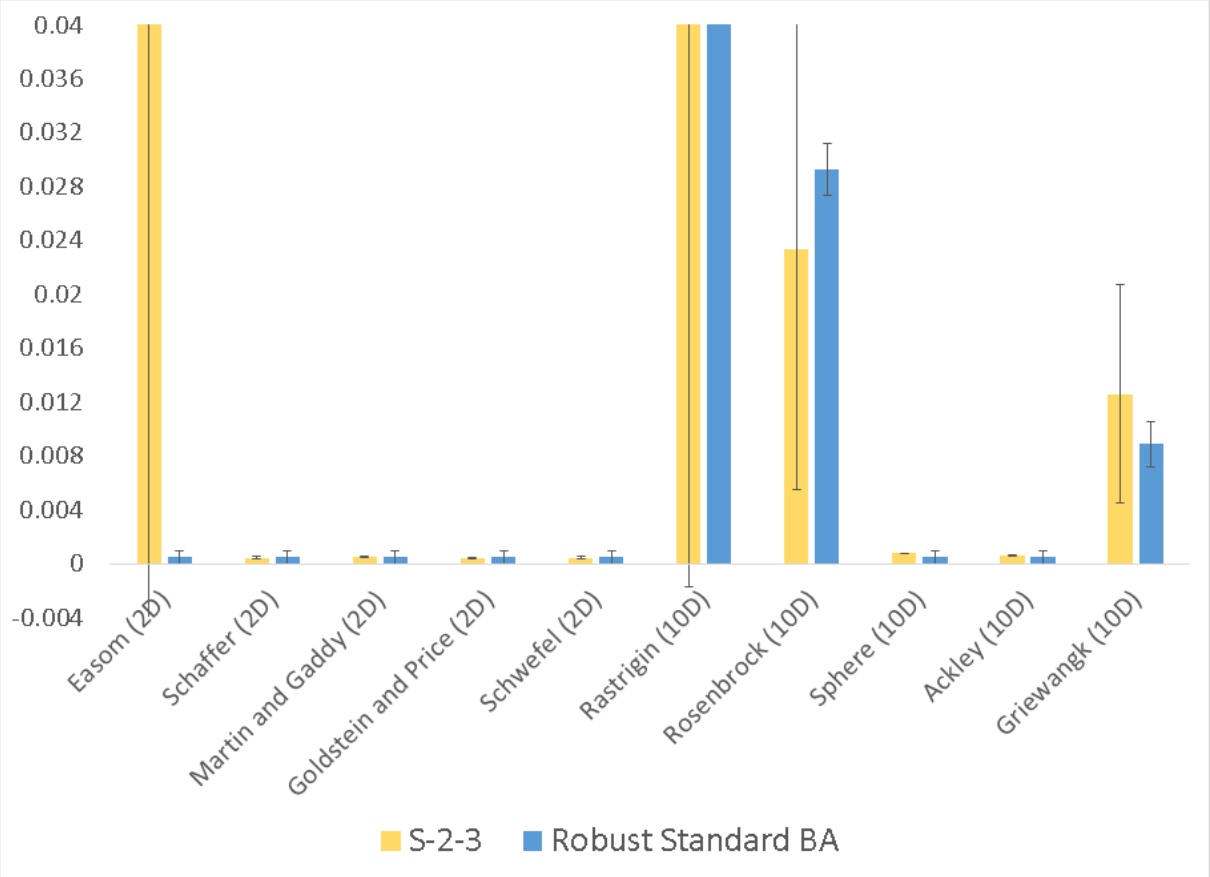


Figure B.5: Global view of the comparison of robust accuracy performance with C.I. 95% (Bi-SBA Vs SBA)

Table B.5: The result of Bi-SBA with Scenario 2-2 (Bi-SBA-2)

Function	Solution (Mean)	Solution (StdDev)	Difference (Mean)	Difference (StdDev)	Evaluations (Mean)	Evaluations (StdDev)	Num.of. Succes
Easom (2D)	-0.939532	0.237369	0.060468	0.237369	2858.96	901.27	50
Schaffer (2D)	0.002259	0.011842	0.002259	0.011842	12988.06	69622.83	50
Camel six hump (2D)	-1.031129	0.000340	0.000471	0.000340	1386.42	376.56	50
Martin and Gaddy (2D)	-4.686971	0.000254	0.000687	0.000254	9534.04	16217.35	50
Goldstein and Price (2D)	-49.998990	0.000075	0.001010	0.000075	10711.50	2477.97	50
Schwefel (2D)	0.000626	0.000278	0.000626	0.000278	1402.42	573.63	50
Michalewicz (5D)	3.000528	0.000285	0.000528	0.000285	1635.34	471.82	50
Trid (6D)	0.000483	0.000277	0.000457	0.000277	3605.84	2620.65	50
MovedHyper (10D)	0.000885	0.000190	0.000885	0.000190	7675.28	491.50	50
Rastrigin (10D)	0.000772	0.000269	0.000772	0.000269	206416.28	119726.54	50
Rosenbrock (10D)	0.008044	0.015737	0.008044	0.015737	304183.80	227445.03	23
Sphere (10D)	0.000775	0.000166	0.000775	0.000166	5519.52	368.24	50
Ackley (10D)	0.000649	0.000276	0.000649	0.000276	5605.94	425.03	50
Griewangk (10D)	0.052704	0.068871	0.052704	0.068871	281353.06	220769.59	26

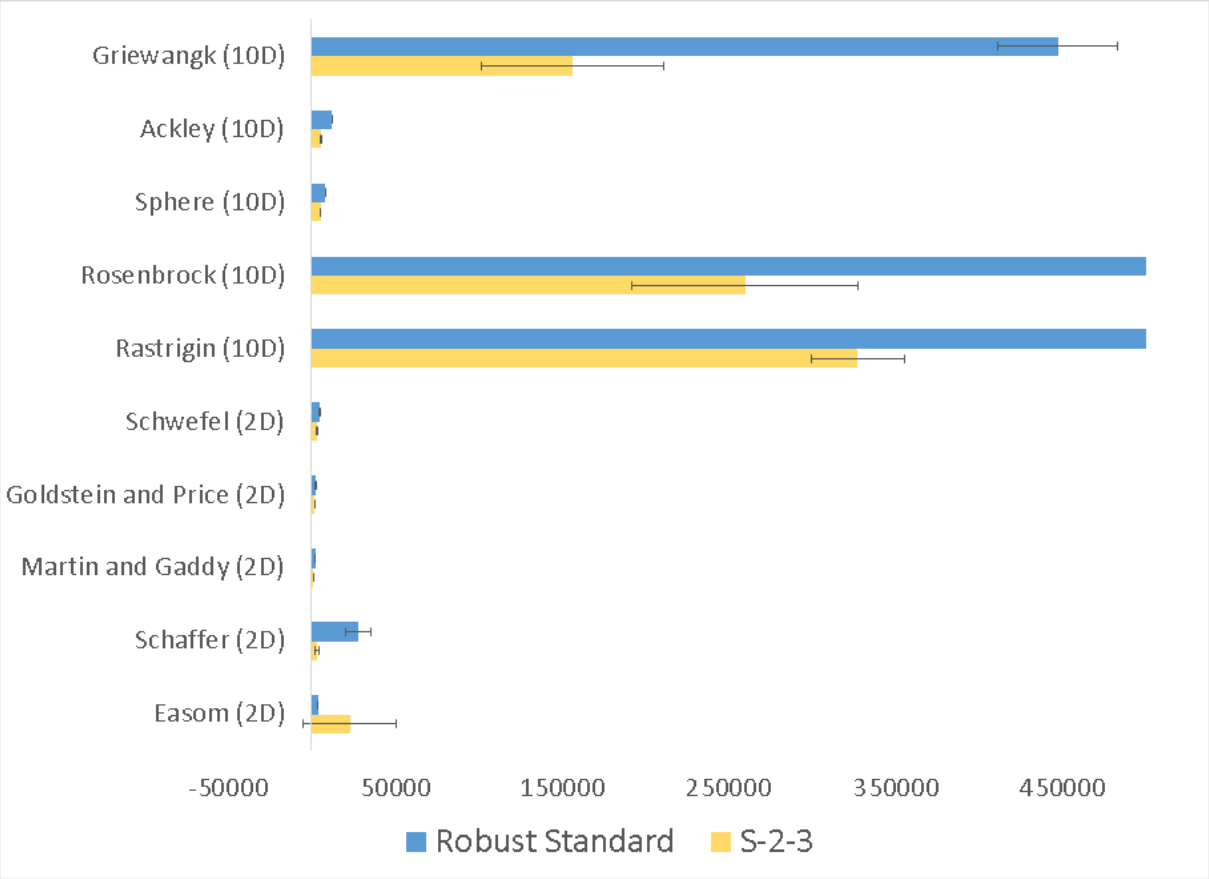


Figure B.6: Global view of the NFE robust performance with C.I. 95% (Bi-SBA Vs SBA)

Table B.6: The result of Bi-SBA with Scenario 2-3 (Bi-SBA-3)

Function	Solution (Mean)	Solution (StdDev)	Difference (Mean)	Difference (StdDev)	Evaluations (Mean)	Evaluations (StdDev)	Num.of. Succes
Easom (2D)	-0.939582	0.237368	0.060418	0.237368	22948.18	97237.09	50
Schaffer (2D)	0.000465	0.000335	0.000465	0.000335	3216.36	4616.67	50
Camel six hump (2D)	-1.031066	0.000303	0.000534	0.000303	1551.88	368.77	50
Martin and Gaddy (2D)	-4.687039	0.000275	0.000619	0.000275	19437.26	34311.84	50
Goldstein and Price (2D)	-49.999092	0.000088	0.000908	0.000088	11487.42	2979.14	50
Schwefel (2D)	0.000512	0.000274	0.000512	0.000274	1358.18	559.85	50
Michalewicz (5D)	3.000453	0.000246	0.000453	0.000246	2003.66	461.05	50
Trid (6D)	0.000499	0.000260	0.000473	0.000260	3313.76	651.01	50
MovedHyper (10D)	0.000868	0.000194	0.000868	0.000194	7863.98	359.28	50
Rastrigin (10D)	0.100154	0.358555	0.100154	0.358555	327228.88	98461.66	46
Rosenbrock (10D)	0.023410	0.062963	0.023410	0.062963	259679.78	239212.26	26
Sphere (10D)	0.000773	0.000168	0.000773	0.000168	5602.36	341.74	50
Ackley (10D)	0.000636	0.000284	0.000636	0.000284	5774.40	520.61	50
Griewangk (10D)	0.012633	0.028404	0.012633	0.028404	156570.44	191467.03	41

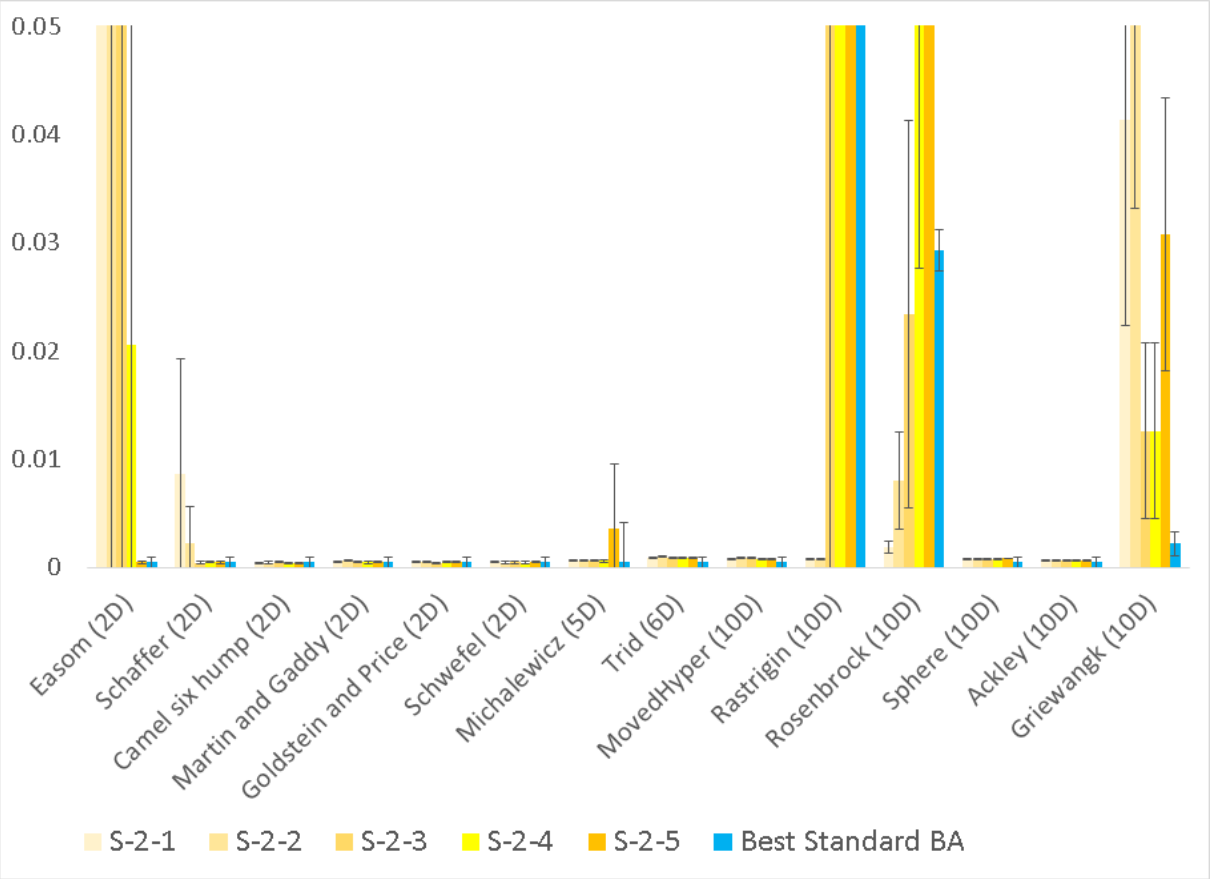


Figure B.7: Global view of the statistical test (95% confidence level) on the accuracy of Bi-SBA

Table B.7: The result of Bi-SBA with Scenario 2-4 (Bi-SBA-4)

Function	Solution (Mean)	Solution (StdDev)	Difference (Mean)	Difference (StdDev)	Evaluations (Mean)	Evaluations (StdDev)	Num.of. Succes
Easom (2D)	-0.979441	0.139920	0.020559	0.139920	3188.04	711.65	50
Schaffer (2D)	0.000489	0.000246	0.000489	0.000246	3702.94	5397.59	50
Camel six hump (2D)	-1.031191	0.000291	0.000409	0.000291	1563.22	496.01	50
Martin and Gaddy (2D)	0.000482	0.000307	0.000482	0.000307	1548.48	402.38	50
Goldstein and Price (2D)	3.000503	0.000300	0.000503	0.000300	1948.58	559.06	50
Schwefel (2D)	0.000485	0.000267	0.000459	0.000267	4347.94	3239.37	50
Michalewicz (5D)	-4.687060	0.000322	0.000598	0.000322	21434.74	46028.12	50
Trid (6D)	-49.999087	0.000112	0.000913	0.000112	12772.86	3815.46	50
MovedHyper (10D)	0.000774	0.000176	0.000774	0.000176	8142.34	454.44	50
Rastrigin (10D)	1.017497	1.296011	1.017497	1.296011	446117.60	70818.53	27
Rosenbrock (10D)	0.283803	0.901374	0.283803	0.901374	304928.84	238810.20	20
Sphere (10D)	0.000773	0.000168	0.000773	0.000168	5602.36	341.74	50
Ackley (10D)	0.000636	0.000284	0.000636	0.000284	5774.40	520.61	50
Griewangk (10D)	0.012633	0.028404	0.012633	0.028404	156570.44	191467.03	41

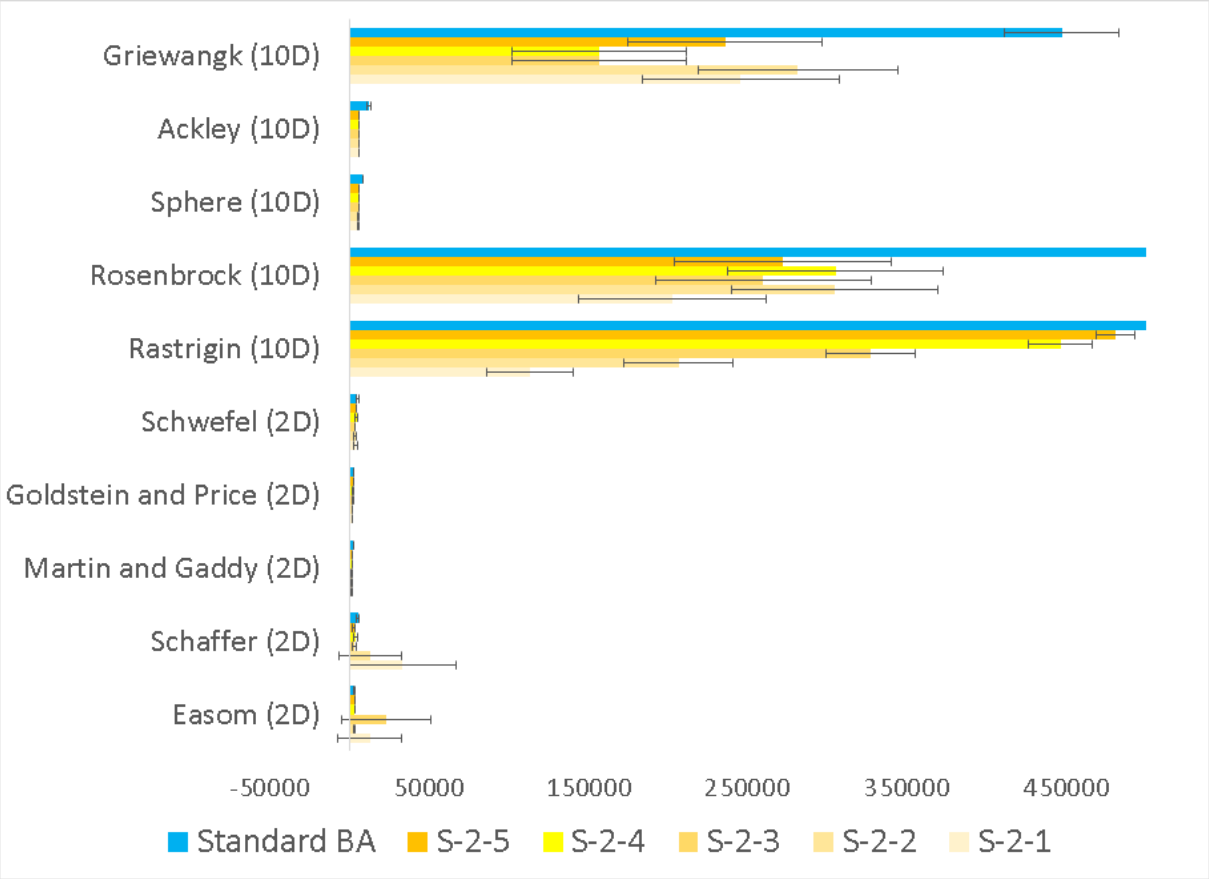


Figure B.8: Global View of the statistical test (95% confidence level) on the NFE of Bi-SBA

Table B.8: The result of Bi-SBA with Scenario 2-5 (Bi-SBA-5)

Function	Solution (Mean)	Solution (StdDev)	Difference (Mean)	Difference (StdDev)	Evaluations (Mean)	Evaluations (StdDev)	Num.of. Succes
Easom (2D)	-0.999513	0.000308	0.000487	0.000308	3485.48	646.64	50
Schaffer (2D)	0.000476	0.000293	0.000476	0.000293	2570.42	1903.11	50
Camel six hump (2D)	-1.031170	0.000280	0.000430	0.000280	1817.20	533.32	50
Martin and Gaddy (2D)	0.000500	0.000304	0.000500	0.000304	1415.40	458.94	50
Goldstein and Price (2D)	3.000508	0.000315	0.000508	0.000315	2406.38	695.69	50
Schwefel (2D)	0.000576	0.000264	0.000551	0.000264	3976.10	558.97	50
Michalewicz (5D)	-4.684063	0.020917	0.003595	0.020917	41307.68	93773.94	49
Trid (6D)	-49.999090	0.000112	0.000910	0.000112	14326.14	4257.49	50
MovedHyper (10D)	0.000805	0.000186	0.000805	0.000186	9308.78	428.64	50
Rastrigin (10D)	2.404653	2.040606	2.404653	2.040606	480293.14	42270.37	11
Rosenbrock (10D)	0.588430	1.803264	0.588430	1.803264	271579.90	239782.05	24
Sphere (10D)	0.000858	0.000137	0.000858	0.000137	5772.38	398.57	50
Ackley (10D)	0.000605	0.000240	0.000605	0.000240	5858.78	552.02	50
Griewangk (10D)	0.030781	0.044415	0.030781	0.044415	235696.20	213798.80	31

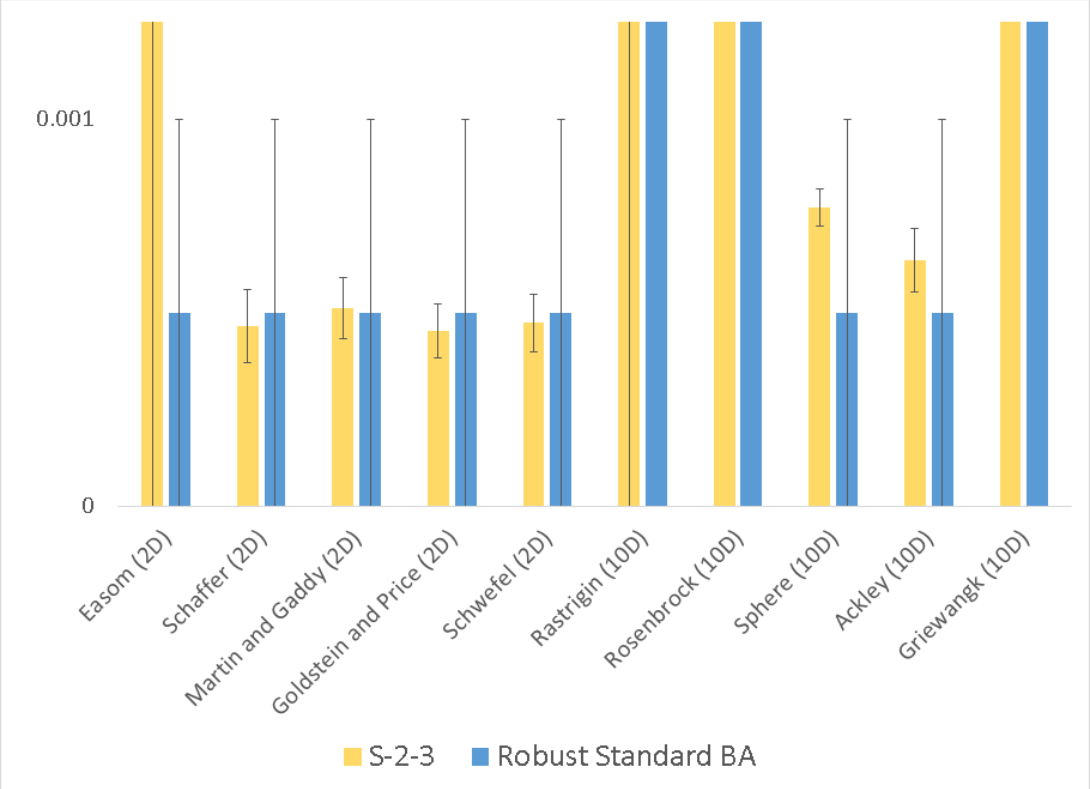


Figure B.9: Comparison of robust accuracy performance with C.I. 95% (Bi-SBA Vs SBA)

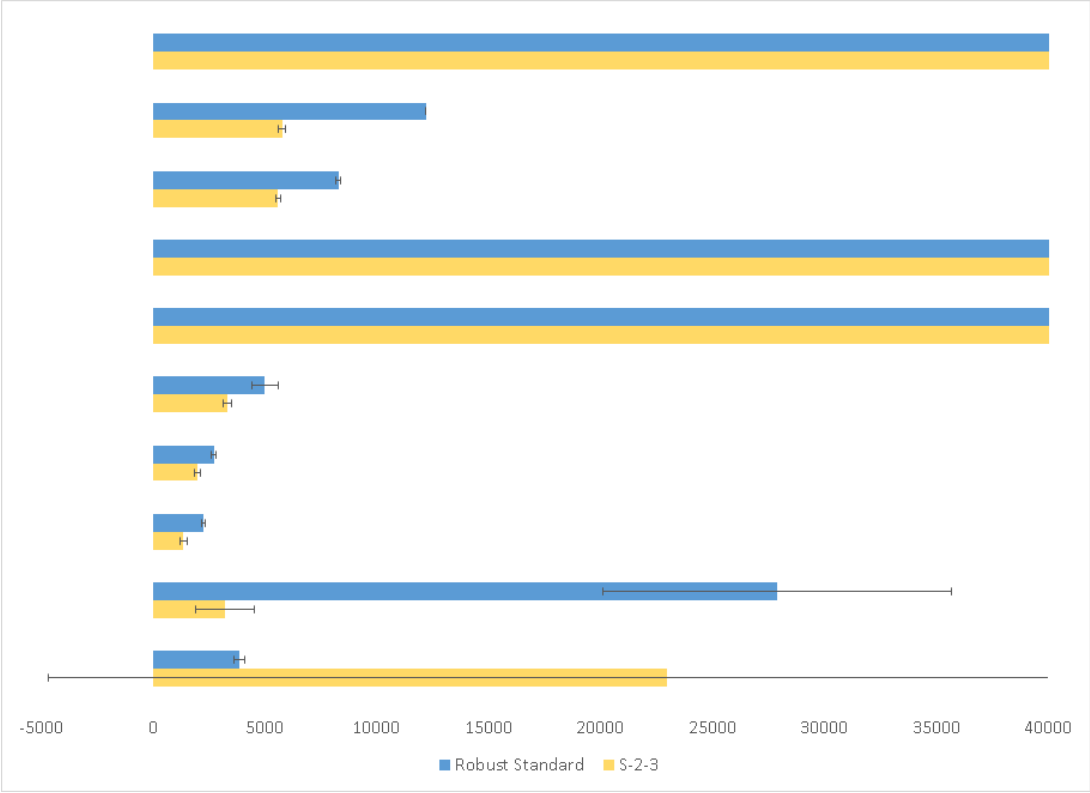


Figure B.10: The NFE robust performance with C.I. 95% (Bi-SBA Vs SBA)

Appendix C

The results of Chapter 4

Table C.1: The diversity of RNG

Run	1	2	3	4	5	6	7	8	9	10	Mean
Eil51	88	86	88	87	90	86	89	86	88	87	**87.5**
Berlin52	99	99	96	100	99	99	99	99	99	99	**98.8**
St70	94	95	93	94	89	92	92	93	91	89	**92.2**
Eil76	93	87	85	90	86	86	88	86	86	86	**87.3**
Pr76	100	100	100	100	100	100	100	100	100	100	**100**
Rat99	96	99	98	99	99	97	98	97	98	97	**97.8**
KroA100	100	100	100	99	100	99	99	99	99	99	**99.4**
KroB100	100	100	100	100	100	100	100	100	100	100	**100**
KroC100	100	100	100	100	99	99	99	99	99	99	**99.4**
KroD100	100	100	100	100	100	100	100	100	100	100	**100**
KroE100	100	100	100	99	100	99	99	99	99	99	**99.4**
KroA150	100	100	100	100	100	100	100	100	100	100	**100**
KroB150	100	100	100	100	100	100	100	100	100	100	**100**
KroA200	100	100	100	100	100	100	100	100	100	100	**100**
KroB200	100	100	100	100	100	100	100	100	100	100	**100**

Table C.2: The diversity of BNSN(F=1;Δr = [0,1])

Run	1	2	3	4	5	6	7	8	9	10	Mean
Eil51	24	31	30	25	29	25	30	26	29	26	**27.5**
Berlin52	43	42	41	44	43	41	41	43	41	42	**42.1**
St70	40	40	34	38	41	36	34	39	38	35	**37.5**
Eil76	45	45	45	50	46	47	46	47	46	46	**46.3**
Pr76	55	54	61	60	54	57	58	55	54	57	**56.5**
Rat99	52	46	49	58	54	51	50	53	56	51	**52**
KroA100	60	61	60	63	57	57	62	57	62	57	**59.6**
KroB100	59	63	63	67	61	62	63	65	66	61	**63**
KroC100	60	61	64	55	61	58	57	56	58	60	**59**
KroD100	62	63	62	57	59	60	58	61	59	58	**59.9**
KroE100	66	56	63	64	64	59	57	58	62	58	**60.7**
KroA150	70	70	69	71	75	73	71	73	73	73	**71.8**
KroB150	71	70	68	66	73	72	66	71	69	72	**69.8**
KroA200	81	73	77	82	78	75	77	76	77	76	**77.2**
KroB200	76	78	81	77	85	80	79	84	83	81	**80.4**

Table C.3: The diversity of BNSN(F=2;Δr = [0,1])

Run	1	2	3	4	5	6	7	8	9	10	Mean
Eil51	36	39	42	36	39	40	36	39	39	37	**38.3**
Berlin52	41	69	72	57	69	52	58	56	57	66	**59.7**
St70	44	48	48	42	51	44	49	46	49	44	**46.5**
Eil76	35	48	45	51	45	40	43	40	50	44	**44.1**
Pr76	54	60	78	78	69	62	64	76	67	74	**68.2**
Rat99	50	48	60	63	54	62	52	59	54	60	**56.2**
KroA100	62	75	63	69	63	72	66	65	63	63	**66.1**
KroB100	62	72	72	72	78	71	72	75	77	73	**72.4**
KroC100	63	60	78	66	63	72	70	71	63	70	**67.6**
KroD100	60	66	69	57	78	76	71	72	57	64	**67**
KroE100	62	60	75	66	75	71	60	70	64	61	**66.4**
KroA150	70	75	75	63	60	67	72	69	70	66	**68.7**
KroB150	71	60	60	63	60	63	61	61	60	62	**62.1**
KroA200	83	69	75	69	69	73	71	72	70	69	**72**
KroB200	81	63	57	75	63	78	66	63	68	68	**68.2**

Table C.4: The diversity of BNSN(F=3;Δr = [0,1])

Run	1	2	3	4	5	6	7	8	9	10	Mean
Eil51	68	68	74	66	66	71	66	67	68	67	**68.1**
Berlin52	92	90	93	91	88	88	91	92	90	88	**90.3**
St70	68	75	67	74	72	68	67	69	73	69	**70.2**
Eil76	69	65	66	66	67	68	65	67	67	66	**66.6**
Pr76	97	96	93	95	97	96	95	93	93	96	**95.1**
Rat99	80	87	81	83	83	82	85	81	81	81	**82.4**
KroA100	96	95	94	91	96	95	91	93	93	91	**93.5**
KroB100	99	96	93	89	96	97	96	96	89	91	**94.2**
KroC100	96	90	91	94	97	90	95	94	90	94	**93.1**
KroD100	95	93	96	89	95	91	93	90	90	93	**92.5**
KroE100	96	97	93	94	96	93	93	93	94	95	**94.4**
KroA150	99	95	94	89	95	93	90	89	92	93	**92.9**
KroB150	95	96	92	97	93	95	95	95	93	94	**94.5**
KroA200	98	96	96	98	98	97	96	96	96	97	**96.8**
KroB200	96	96	96	94	96	95	94	94	95	95	**95.1**

Table C.5: The error result (%) of RNG, BNSN(1), BNSN(2), BNSN(3) on 15 datasets TSPLIB for 10 independent runs

No	RNG	BNSN(1)	BNSN(2)	BNSN(3)
F-1	287.79	23.47	32.16	32.86
F-2	296.94	23.83	25.06	27.84
F-3	441.78	24.74	28.59	28.44
F-4	369.52	23.79	27.70	28.44
F-5	433.75	36.15	35.80	36.76
F-6	594.47	26.92	33.20	33.11
F-7	705.38	27.16	36.03	35.34
F-8	663.52	25.72	35.90	34.04
F-9	720.46	25.90	39.34	36.93
F-10	660.66	27.99	28.63	30.72
F-11	683.65	24.32	31.00	31.37
F-12	869.41	27.12	33.72	33.90
F-13	883.12	35.32	34.18	38.04
F-14	1058.58	27.89	37.38	34.90
F-15	1029.90	26.24	31.27	31.92
Mean	**646.59**	**27.10**	**32.66**	**32.97**

Table C.6: The result of basic BA on 15 datasets TSPLIB for 10 independent runs

Run	1	2	3	4	5	6	7	8	9	10
F-1	433	435	428	430	432	432	428	428	433	429
F-2	7542	7728	7542	7542	7577	7542	7775	7542	7566	7741
F-3	675	687	688	681	685	687	694	682	684	680
F-4	551	554	552	552	558	549	555	551	549	558
F-5	109046	109523	108234	111114	110647	109749	109221	109555	110040	109836
F-6	1260	1231	1227	1257	1271	1261	1236	1270	1274	1252
F-7	21450	21709	21647	22110	21641	22035	21315	21673	21814	21711
F-8	22622	22612	22711	22540	22601	22660	22498	22678	22520	22334
F-9	20983	21136	21028	21010	21086	21213	21564	20897	20892	21093
F-10	22316	21567	21838	21982	21795	22288	21726	22045	21621	21765
F-11	22600	22354	22561	22441	22388	22528	22416	22795	22557	22344
F-12	27792	27980	28182	28199	28055	27841	27948	28196	28094	28130
F-13	27146	27677	27524	26642	27253	27623	27033	27508	27605	27463
F-14	32622	32905	32217	31546	31866	31916	32173	31597	31742	31754
F-15	31769	32927	32751	32227	32199	32196	32290	32320	32318	32297

Table C.7: the error results of 10 runs of basic CBA

No	1	2	3	4	5	6	7	8	9	10
F-1	0.01643	0.02113	0.00469	0.00939	0.01408	0.01408	0.00469	0.00469	0.01643	0.00704
F-2	0.00000	0.02466	0.00000	0.00000	0.00464	0.00000	0.03089	0.00000	0.00318	0.02639
F-3	0.00000	0.01778	0.01926	0.00889	0.01481	0.01778	0.02815	0.01037	0.01333	0.00741
F-4	0.02416	0.02974	0.02602	0.02602	0.03717	0.02045	0.03160	0.02416	0.02045	0.03717
F-5	0.00820	0.01261	0.00069	0.02732	0.02300	0.01470	0.00982	0.01291	0.01739	0.01550
F-6	0.04046	0.01652	0.01321	0.03799	0.04955	0.04129	0.02064	0.04872	0.05202	0.03386
F-7	0.00789	0.02006	0.01715	0.03891	0.01687	0.03538	0.00155	0.01837	0.02500	0.02016
F-8	0.02172	0.02127	0.02574	0.01802	0.02078	0.02344	0.01612	0.02425	0.01712	0.00872
F-9	0.01128	0.01865	0.01345	0.01258	0.01624	0.02236	0.03928	0.00713	0.00689	0.01658
F-10	0.04799	0.01282	0.02555	0.03231	0.02353	0.04668	0.02029	0.03527	0.01536	0.02212
F-11	0.02411	0.01296	0.02234	0.01690	0.01450	0.02084	0.01577	0.03294	0.02216	0.01251
F-12	0.04781	0.05489	0.06251	0.06315	0.05772	0.04965	0.05369	0.06304	0.05919	0.06055
F-13	0.03888	0.05920	0.05335	0.01959	0.04298	0.05714	0.03456	0.05274	0.05645	0.05101
F-14	0.11080	0.12044	0.09701	0.07416	0.08506	0.08676	0.09551	0.07590	0.08084	0.08124
F-15	0.07922	0.11856	0.11258	0.09478	0.09383	0.09373	0.09692	0.09794	0.09787	0.09716

Table C.8: The result of CBA+BNSN(1) on 15 datasets TSPLIB for 10 independent runs

Run	1	2	3	4	5	6	7	8	9	10
F-1	427	429	430	428	427	430	427	430	430	430
F-2	7542	7542	7542	7565	7542	7542	7596	7542	7547	7566
F-3	684	684	684	686	686	688	684	684	685	684
F-4	547	545	541	549	548	554	549	551	543	550
F-5	109653	109938	109728	110040	110040	109938	110767	110051	110430	110565
F-6	1225	1229	1243	1223	1212	1253	1223	1254	1253	1253
F-7	21373	21282	21282	21282	21292	21358	21393	21360	21308	21379
F-8	22362	22282	22387	22311	22179	22320	22365	22301	22608	22361
F-9	20880	20852	20965	21166	21000	20946	20929	20955	20785	21000
F-10	21778	21855	21761	21713	21990	21704	21987	22273	21792	21770
F-11	22306	22369	22458	22116	22408	22286	22283	22244	22236	22244
F-12	27114	27348	27518	26916	27062	27487	27922	27510	27564	27767
F-13	26601	26684	26690	27029	26748	26394	26727	26594	26459	26728
F-14	29982	29915	29887	29679	29883	29804	29688	29656	29845	29712
F-15	30777	31271	30598	31032	30816	30549	30424	30379	30589	30321

Table C.9: the error results of 10 runs of CBA+BNSN(1)

No	1	2	3	4	5	6	7	8	9	10
F-1	0.00235	0.00704	0.00939	0.00469	0.00235	0.00939	0.00235	0.00939	0.00939	0.00939
F-2	0.00000	0.00000	0.00000	0.00305	0.00000	0.00000	0.00716	0.00000	0.00066	0.00318
F-3	0.01333	0.01333	0.01333	0.01630	0.01630	0.01926	0.01333	0.01333	0.01481	0.01333
F-4	0.01673	0.01301	0.00558	0.02045	0.01859	0.02974	0.02045	0.02416	0.00929	0.02230
F-5	0.01381	0.01645	0.01451	0.01739	0.01739	0.01645	0.02411	0.01749	0.02100	0.02225
F-6	0.01156	0.01486	0.02642	0.00991	0.00083	0.03468	0.00991	0.03551	0.03468	0.03468
F-7	0.00428	0.00000	0.00000	0.00000	0.00047	0.00357	0.00522	0.00367	0.00122	0.00456
F-8	0.00998	0.00637	0.01111	0.00768	0.00172	0.00808	0.01012	0.00723	0.02109	0.00994
F-9	0.00631	0.00496	0.01041	0.02010	0.01210	0.00949	0.00868	0.00993	0.00174	0.01210
F-10	0.02273	0.02635	0.02193	0.01968	0.03269	0.01925	0.03254	0.04598	0.02339	0.02235
F-11	0.01078	0.01364	0.01767	0.00218	0.01541	0.00988	0.00974	0.00798	0.00761	0.00798
F-12	0.02224	0.03107	0.03748	0.01478	0.02028	0.03631	0.05271	0.03717	0.03921	0.04686
F-13	0.01803	0.02120	0.02143	0.03440	0.02365	0.01010	0.02285	0.01776	0.01259	0.02289
F-14	0.02091	0.01863	0.01767	0.01059	0.01754	0.01485	0.01090	0.00981	0.01624	0.01171
F-15	0.04552	0.06230	0.03944	0.05418	0.04685	0.03778	0.03353	0.03200	0.03913	0.03003

Table C.10: The result of CBA+BNSN(2) on 15 datasets TSPLIB for 10 independent runs

Run	1	2	3	4	5	6	7	8	9	10
F-1	429	427	428	427	427	428	427	428	427	427
F-2	7542	7542	7902	7542	7542	7542	7902	7902	7749	7902
F-3	682	685	685	685	677	677	683	682	682	682
F-4	547	546	546	544	543	542	547	543	541	543
F-5	109653	110684	110684	109653	110495	109694	112037	109523	111349	110891
F-6	1240	1234	1234	1231	1233	1230	1232	1222	1231	1236
F-7	21446	21282	21282	21411	21410	21494	21373	22026	21877	21825
F-8	22300	22269	22269	22392	22621	22563	22257	22179	22681	22257
F-9	21181	20915	20915	20965	20852	21465	21571	21465	21410	21546
F-10	21856	21839	21839	21935	21820	21940	21789	21607	21505	21501
F-11	22348	22107	22107	22286	22352	22433	22362	22266	22449	22355
F-12	27140	27127	27127	26954	27106	27208	27271	27072	26986	27407
F-13	26549	26381	26381	26664	26788	26995	27422	27048	27086	27019
F-14	29942	29784	29784	29880	30213	29679	29854	29827	29623	29852
F-15	30925	30898	30898	31430	31253	30391	30205	30754	30888	30366

Table C.11: the error results of 10 runs of CBA+BNSN(2)

No	1	2	3	4	5	6	7	8	9	10
F-1	0.00704	0.00235	0.00469	0.00235	0.00235	0.00469	0.00235	0.00469	0.00235	0.00235
F-2	0.00000	0.00000	0.04773	0.00000	0.00000	0.00000	0.04773	0.04773	0.02745	0.04773
F-3	0.01037	0.01481	0.01481	0.01481	0.00296	0.00296	0.01185	0.01037	0.01037	0.01037
F-4	0.01673	0.01487	0.01487	0.01115	0.00929	0.00743	0.01673	0.00929	0.00558	0.00929
F-5	0.01381	0.02335	0.02335	0.01381	0.02160	0.01419	0.03585	0.01261	0.02949	0.02526
F-6	0.02395	0.01899	0.01899	0.01652	0.01817	0.01569	0.01734	0.00908	0.01652	0.02064
F-7	0.00771	0.00000	0.00000	0.00606	0.00601	0.00996	0.00428	0.03496	0.02796	0.02551
F-8	0.00718	0.00578	0.00578	0.01134	0.02168	0.01906	0.00524	0.00172	0.02439	0.00524
F-9	0.02082	0.00800	0.00800	0.01041	0.00496	0.03451	0.03962	0.03451	0.03186	0.03841
F-10	0.02639	0.02559	0.02559	0.03010	0.02470	0.03034	0.02325	0.01470	0.00991	0.00972
F-11	0.01269	0.00177	0.00177	0.00988	0.01287	0.01654	0.01332	0.00897	0.01726	0.01301
F-12	0.02322	0.02273	0.02273	0.01621	0.02194	0.02579	0.02816	0.02066	0.01742	0.03329
F-13	0.01604	0.00961	0.00961	0.02044	0.02518	0.03310	0.04945	0.03513	0.03659	0.03402
F-14	0.01955	0.01417	0.01417	0.01743	0.02877	0.01059	0.01655	0.01563	0.00868	0.01648
F-15	0.05055	0.04963	0.04963	0.06770	0.06169	0.03241	0.02609	0.04474	0.04929	0.03156

Table C.12: The result of CBA+BNSN(3) on 15 datasets TSPLIB for 10 independent runs

Run	1	2	3	4	5	6	7	8	9	10
F-1	427	427	428	427	429	427	428	427	428	428
F-2	7685	7542	7542	7542	7542	7542	7542	7542	7542	7542
F-3	687	685	685	685	682	685	685	682	684	684
F-4	541	548	549	543	543	547	544	550	549	540
F-5	109653	108234	109653	109535	109500	109938	110051	110001	109046	109632
F-6	1221	1221	1218	1226	1236	1225	1239	1213	1227	1228
F-7	21282	21282	21636	21282	21282	21292	21292	21320	21282	21292
F-8	22246	22275	22264	22258	22179	22297	22428	22459	22306	22566
F-9	21010	21096	21016	21165	20749	20880	20880	21152	21081	20937
F-10	21908	21514	21604	21839	21885	21868	21629	21852	21611	21879
F-11	22308	22236	22165	22298	22396	22412	22357	22310	22401	22107
F-12	27265	27346	27183	27008	27453	27332	27036	27388	26965	27105
F-13	26773	26656	26813	26791	26631	26382	26859	26600	26602	26681
F-14	30025	29838	29805	29788	29654	29675	29701	29849	29693	29974
F-15	30245	30564	31023	31336	30622	30730	30546	31043	30864	30891

Table C.13: the error results of 10 runs of CBA+BNSN(3)

No	1	2	3	4	5	6	7	8	9	10
F-1	0.00235	0.00235	0.00469	0.00235	0.00704	0.00235	0.00469	0.00235	0.00469	0.00469
F-2	0.01896	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
F-3	0.01778	0.01481	0.01481	0.01481	0.01037	0.01481	0.01481	0.01037	0.01333	0.01333
F-4	0.00558	0.01859	0.02045	0.00929	0.00929	0.01673	0.01115	0.02230	0.02045	0.00372
F-5	0.01381	0.00069	0.01381	0.01272	0.01240	0.01645	0.01749	0.01703	0.00820	0.01362
F-6	0.00826	0.00826	0.00578	0.01239	0.02064	0.01156	0.02312	0.00165	0.01321	0.01404
F-7	0.00000	0.00000	0.01663	0.00000	0.00000	0.00047	0.00047	0.00179	0.00000	0.00047
F-8	0.00474	0.00605	0.00556	0.00528	0.00172	0.00705	0.01296	0.01436	0.00745	0.01920
F-9	0.01258	0.01672	0.01287	0.02005	0.00000	0.00631	0.00631	0.01942	0.01600	0.00906
F-10	0.02883	0.01033	0.01456	0.02559	0.02775	0.02696	0.01573	0.02620	0.01489	0.02747
F-11	0.01088	0.00761	0.00440	0.01042	0.01486	0.01559	0.01310	0.01097	0.01509	0.00177
F-12	0.02794	0.03099	0.02485	0.01825	0.03502	0.03046	0.01930	0.03257	0.01663	0.02190
F-13	0.02461	0.02013	0.02614	0.02530	0.01917	0.00964	0.02790	0.01799	0.01806	0.02109
F-14	0.02237	0.01600	0.01488	0.01430	0.00974	0.01045	0.01134	0.01638	0.01107	0.02063
F-15	0.02745	0.03829	0.05388	0.06451	0.04026	0.04392	0.03767	0.05456	0.04848	0.04939

Table C.14: The evaluation (NFE) result of basic CBA on 15 datasets TSPLIB for 10 independent runs

Run	1	2	3	4	5	6	7	8	9	10
F-1	532099	1335028	1838657	1741768	1870056	295694	156316	264191	337804	216844
F-2	236165	1799750	1861278	2285778	1062627	196904	162000	141081	421947	300664
F-3	578374	1728562	2174396	1722543	1759798	743499	1059300	488940	481114	234398
F-4	771585	2151183	1717403	1333499	1554383	744163	1749213	926801	628913	333338
F-5	1073717	1789260	1167049	1521930	2354837	493243	517586	388978	679075	1227963
F-6	1144389	1564362	2434858	1376953	1477326	2253885	1512185	852010	741795	777804
F-7	1564943	1096195	2412295	1006947	2257408	443547	907665	2081432	582429	1184289
F-8	896731	2271322	1749894	1894317	1312608	1384256	945013	988513	934639	2586028
F-9	990151	1723183	1725552	2373378	1302164	921194	991645	2533232	847645	1727277
F-10	690970	2668773	1933733	1497275	1986206	1885676	1633895	2581910	768878	1090021
F-11	842265	2306697	1564822	1408284	2009189	1522005	1451658	651788	1501275	1352117
F-12	1734835	1622932	1822401	1779493	2457518	1821572	1287102	2146404	2330204	1914812
F-13	2647852	1745825	1665794	1813697	1799222	1892341	1805000	1828942	1819741	1860192
F-14	1738337	2739448	1737168	1741733	2675107	2682320	2565594	1946639	1900502	2487754
F-15	1732511	1702911	2784129	1732491	2700028	2558065	1984993	2647650	2508456	2200044

Table C.15: The evaluation (NFE) result of CBA+BNSN(1) on 15 datasets TSPLIB for 10 independent runs

Run	1	2	3	4	5	6	7	8	9	10
F-1	235577	2127662	1288309	1846495	1585190	100536	67434	248392	80020	125793
F-2	243679	1557315	1897841	1738869	1584630	112881	51163	52748	210232	49580
F-3	1105621	922785	1602535	1722653	1864745	230189	607953	410344	303628	283672
F-4	344668	2806098	600876	3050362	868147	842916	1093218	495622	279235	247870
F-5	531345	1586368	2128023	1544652	1545691	501127	211007	791203	961265	316183
F-6	958852	2096089	1432095	2220266	1016768	190788	868433	861250	327805	657295
F-7	331823	2237663	2635573	1257699	1282200	247894	234249	426987	386565	611535
F-8	1172836	1512077	2257398	1861847	1861847	278840	960474	221899	642491	482035
F-9	1391499	1102643	2273737	1227803	1920947	241573	601053	172155	365875	769899
F-10	1157729	1335201	1470924	2288746	2009138	904586	1877336	861061	759587	1147006
F-11	1387412	915902	2058420	1996972	2062556	274736	248809	1585387	183595	551270
F-12	1154841	2195359	1816007	1436113	1349699	1878172	3530311	1177189	1470735	1935823
F-13	1596231	1718004	1440754	2187890	1748112	1679912	1452148	3045767	4069796	3718725
F-14	1482015	1858304	1796319	1658838	1754507	2176014	4368314	2069631	3780110	4541782
F-15	1716319	1056162	2225549	1793944	1869404	3717719	3091807	4577072	5476380	4032529

Table C.16: The evaluation (NFE) result of CBA+BNSN(2) on 15 datasets TSPLIB for 10 independent runs

Run	1	2	3	4	5	6	7	8	9	10
F-1	214092	1831087	1592128	1760957	2222584	68470	84304	75312	152655	103149
F-2	62706	1932051	1612468	1691847	119265	34647	265197	510280	145823	1031378
F-3	929247	1131654	1788808	1911596	699477	572558	1145370	539517	351963	570459
F-4	733162	930008	1234085	1903603	287044	352987	516046	149064	652418	903295
F-5	667107	1486524	2078764	1322477	577141	200836	619490	367856	718260	856225
F-6	394510	1717958	2557276	1468005	1441930	776805	600609	1695553	1330980	409202
F-7	1464072	1396626	1654181	1541697	888006	890112	841476	973971	281980	1228932
F-8	578884	1950559	1509837	1720992	545291	1427758	594794	1214541	480865	1191598
F-9	1563767	1801511	1818889	2100772	892196	2619272	909729	998989	605158	869307
F-10	1625856	2048006	2060732	1600848	1192592	803017	661257	796792	514012	290274
F-11	1040564	1910583	894440	2050284	1335566	384623	895270	1097304	1294153	1017624
F-12	1648982	1674500	1779524	1757446	2864592	1918842	3119103	1387964	2329175	4443186
F-13	1625264	1693057	1986559	1810223	2213779	655881	1786232	1693938	3008566	3280055
F-14	1714598	1841029	1640813	5079744	1979951	2186642	1741192	1517927	5443256	3982419
F-15	1490096	1804428	1559051	1941914	1606671	2007126	5390620	1411177	5161459	3089657

Table C.17: The evaluation (NFE) result of CBA+BNSN(3) on 15 datasets TSPLIB for 10 independent runs

Run	1	2	3	4	5	6	7	8	9	10
F-1	118376	1573153	797902	1491223	1699494	1766120	1551017	192578	472730	146806
F-2	277913	1610657	1448256	1605469	850248	1617052	1733680	132022	55922	272437
F-3	216425	953566	1230421	821008	1766144	1427964	1653110	793279	514877	496235
F-4	1147905	965733	1632730	1213930	1454089	1525345	1529912	540309	306671	184419
F-5	322170	1392654	1182618	1656052	1533946	649650	2375763	270614	942478	414090
F-6	852648	1656537	959525	800323	904892	1577085	1520181	374064	961143	748948
F-7	483883	1679613	959768	1186659	1294577	1677424	1815417	880728	1168751	354496
F-8	454835	1640254	1828247	2027709	1167804	1473352	632747	659064	219786	414842
F-9	921157	1412921	907305	1818293	1758049	1936233	881060	2151003	1026784	922201
F-10	1305808	1793793	1799794	1886231	963980	1179200	990671	413530	793704	399022
F-11	658343	788156	1682048	1063621	1253408	1578217	1667019	487120	1096268	525382
F-12	1671714	1667455	1712763	1705296	1184953	1378133	1541060	4029352	1433270	1896791
F-13	1336398	1541696	2316044	1016149	2050295	1510396	2026526	2269900	2224661	3581478
F-14	1664134	1775935	1644363	1693582	1788745	1661798	1452883	5991528	3484015	2463773
F-15	1725564	1742980	1542807	1849691	1413465	1437287	1266758	2411697	4810707	2862254

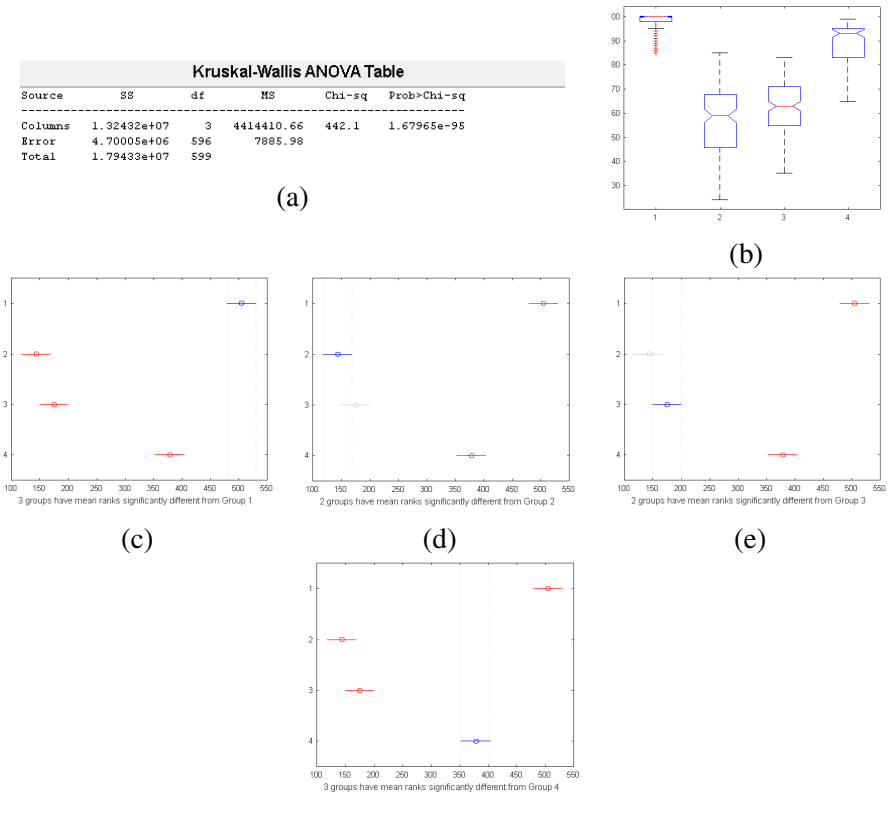


Figure C.1: Statistic test Constructive Heuristic (1= RNG; 2= BNSN(1); 3= BNSN(2); 4= BNSN(3))

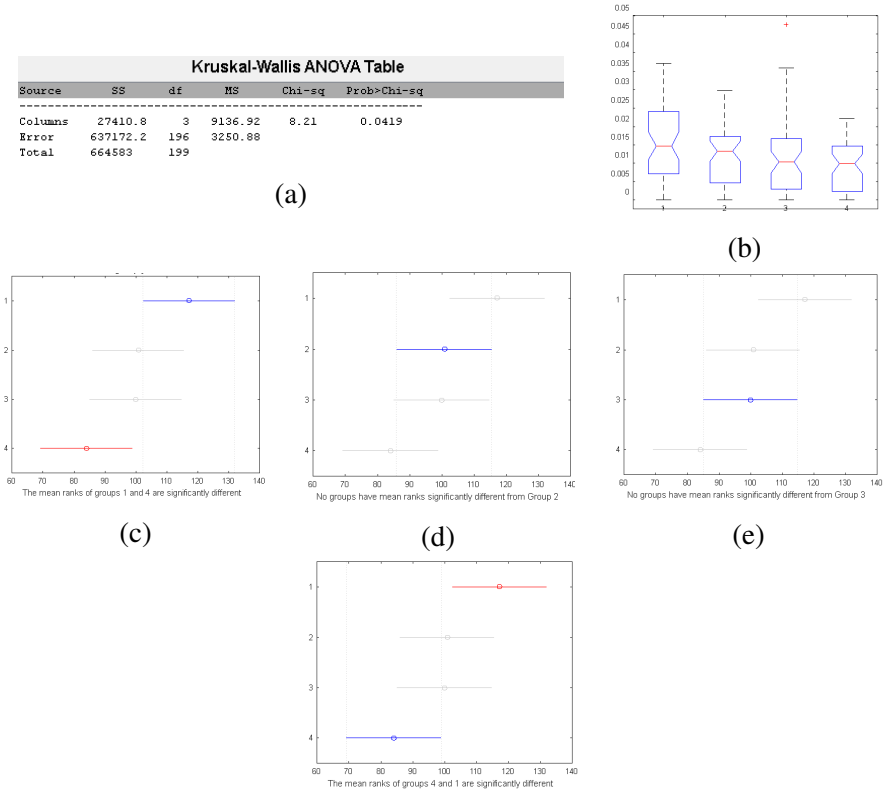


Figure C.2: Statistic test TSPLIB [50,76] dimensions, (1= RNG; 2= BNSN(1); 3= BNSN(2); 4= BNSN(3))

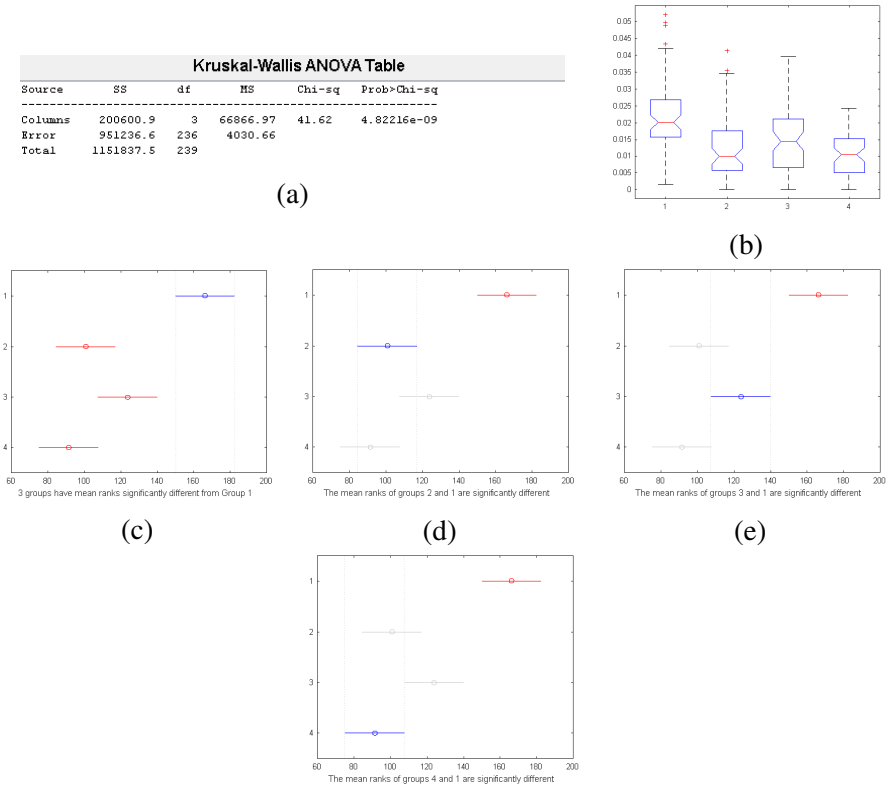


Figure C.3: Statistic test TSPLIB [99,100] dimensions, (1= RNG; 2= BNSN(1); 3= BNSN(2); 4= BNSN(3))

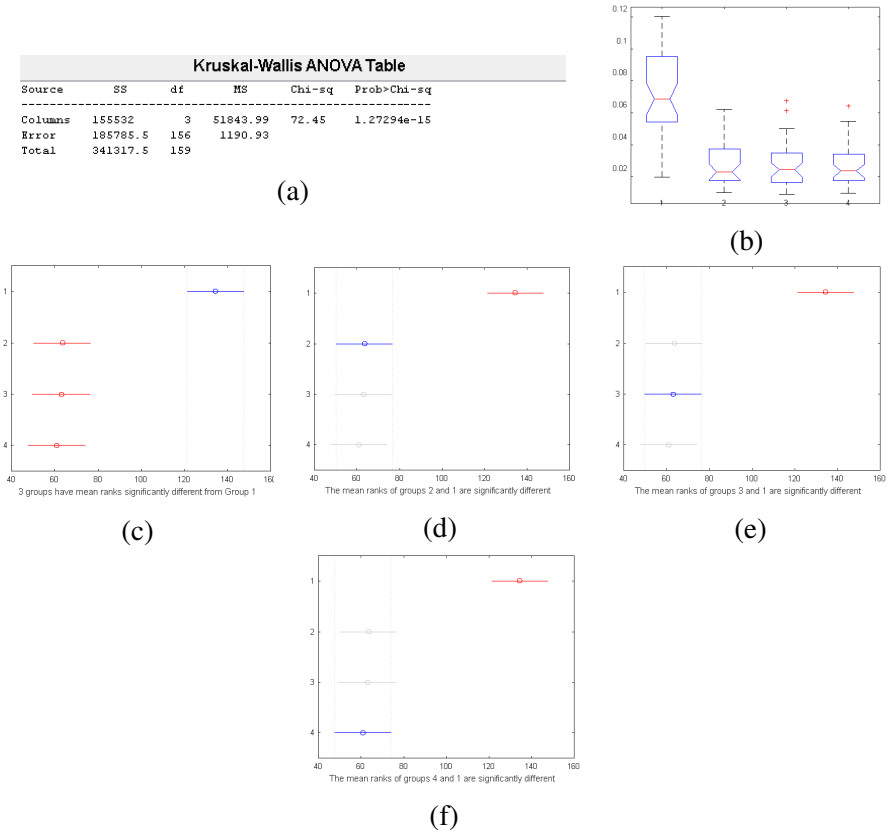


Figure C.4: Statistic test TSPLIB [150,200] dimensions, (1= RNG; 2= BNSN(1); 3= BNSN(2); 4= BNSN(3))

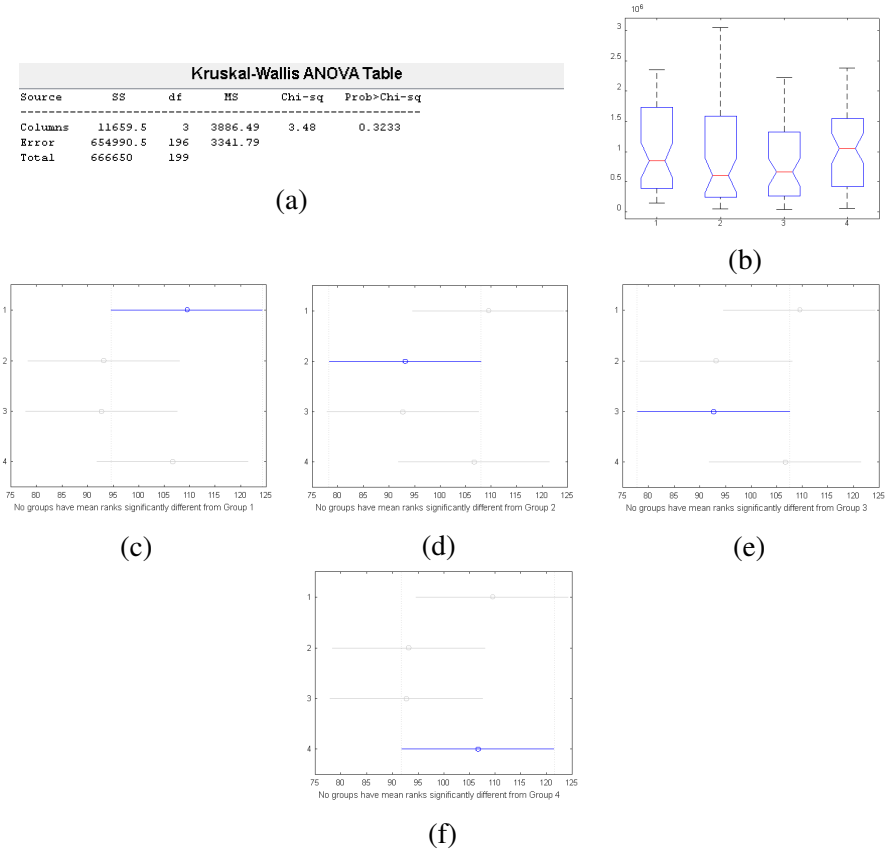


Figure C.5: NFE Statistic test TSPLIB [50,76] dimensions, (1= RNG; 2= BNSN(1); 3= BNSN(2); 4= BNSN(3))

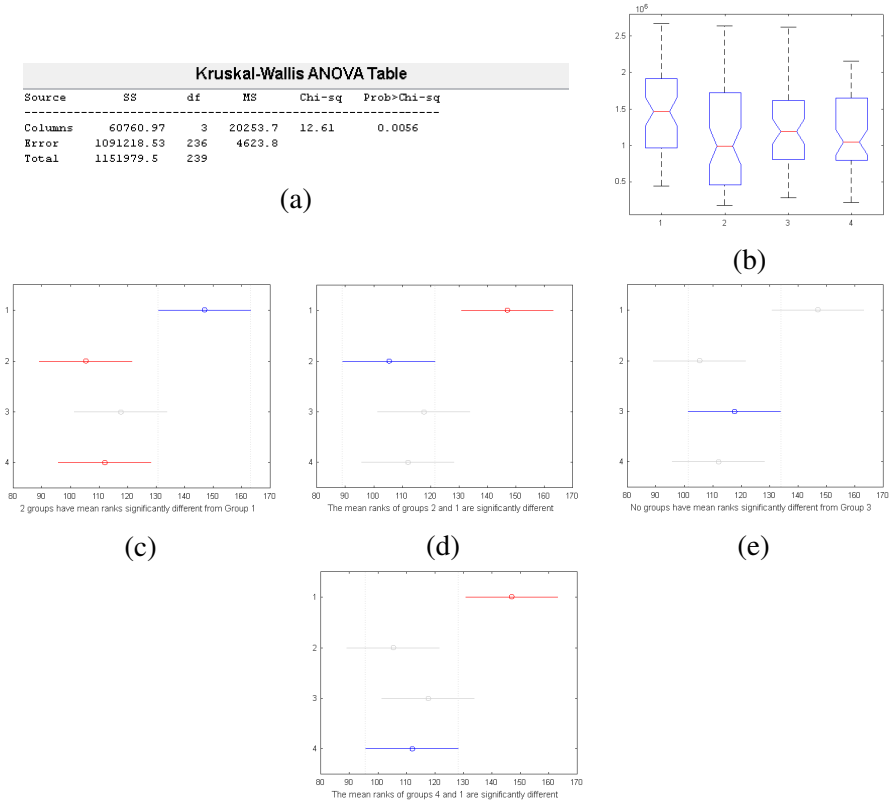


Figure C.6: NFE Statistic test TSPLIB [99,100] dimensions, (1= RNG; 2= BNSN(1); 3= BNSN(2); 4= BNSN(3))

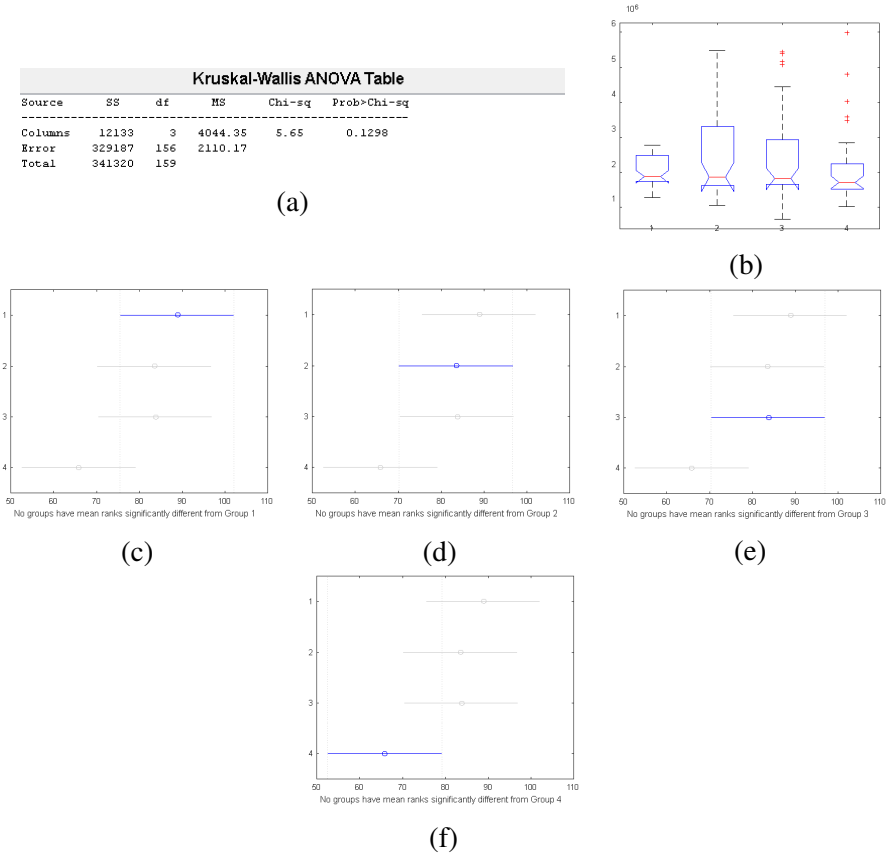


Figure C.7: NFE Statistic test TSPLIB [150,200] dimensions, (1= RNG; 2= BNSN(1); 3= BNSN(2); 4= BNSN(3))

Appendix D

The results of Chapter 5

Table D.1: The results of 10 runs of CBA+BRO

No	1	2	3	4	5	6	7	8	9	10
F-1	426	427	427	427	427	426	427	426	426	426
F-2	7542	7542	7542	7542	7547	7543	7544	7545	7543	7542
F-3	676	677	676	677	676	676	676	676	676	676
F-4	538	538	539	539	539	538	538	538	538	538
F-5	108304	108159	108333	108425	108274	108423	108159	108365	108420	108325
F-6	1214	1211	1214	1214	1211	1213	1213	1213	1212	1212
F-7	21343	21331	21343	21282	21282	21285	21282	21297	21296	21288
F-8	22141	22193	22146	22141	22199	22183	22156	22155	22161	22194
F-9	20749	20769	20769	20769	20754	20753	20762	20756	20760	20753
F-10	21294	21294	21351	21294	21314	21329	21308	21348	21331	21325
F-11	22094	22105	22084	22068	22110	22080	22102	22099	22104	22068
F-12	26563	26524	26681	26584	26597	26597	26565	26599	26569	26566
F-13	26196	26195	26148	26196	26206	26205	26199	26186	26198	26198
F-14	29450	29550	29441	29408	29452	29519	29529	29447	29493	29459
F-15	29645	29595	29548	29602	29634	29619	29580	29603	29588	29606

Table D.2: The error results of 10 runs of CBA+BRO

No	1	2	3	4	5	6	7	8	9	10
F-1	0.00000	0.00235	0.00235	0.00235	0.00235	0.00000	0.00235	0.00000	0.00000	0.00000
F-2	0.00000	0.00000	0.00000	0.00000	0.00066	0.00013	0.00027	0.00040	0.00013	0.00000
F-3	0.00148	0.00296	0.00148	0.00296	0.00148	0.00148	0.00148	0.00148	0.00148	0.00148
F-4	0.00000	0.00000	0.00186	0.00186	0.00186	0.00000	0.00000	0.00000	0.00000	0.00000
F-5	0.00134	0.00000	0.00161	0.00246	0.00106	0.00244	0.00000	0.00190	0.00241	0.00153
F-6	0.00248	0.00000	0.00248	0.00248	0.00000	0.00165	0.00165	0.00165	0.00083	0.00083
F-7	0.00287	0.00230	0.00287	0.00000	0.00000	0.00014	0.00000	0.00070	0.00066	0.00028
F-8	0.00000	0.00235	0.00023	0.00000	0.00262	0.00190	0.00068	0.00063	0.00090	0.00239
F-9	0.00000	0.00096	0.00096	0.00096	0.00024	0.00019	0.00063	0.00034	0.00053	0.00019
F-10	0.00000	0.00000	0.00268	0.00000	0.00094	0.00164	0.00066	0.00254	0.00174	0.00146
F-11	0.00118	0.00168	0.00073	0.00000	0.00190	0.00054	0.00154	0.00140	0.00163	0.00000
F-12	0.00147	0.00000	0.00592	0.00226	0.00275	0.00275	0.00155	0.00283	0.00170	0.00158
F-13	0.00253	0.00249	0.00068	0.00253	0.00291	0.00287	0.00264	0.00214	0.00260	0.00260
F-14	0.00279	0.00620	0.00249	0.00136	0.00286	0.00514	0.00548	0.00269	0.00426	0.00310
F-15	0.00707	0.00537	0.00377	0.00561	0.00669	0.00618	0.00486	0.00564	0.00513	0.00574

Table D.3: The NFE results of 10 runs of CBA+BRO

No	1	2	3	4	5	6	7	8	9	10
F-1	29242	23290	31649	29389	31504	27459	24246	26067	28534	30197
F-2	40099	37145	27498	27983	38645	30807	28762	35170	32989	29256
F-3	60732	66083	82137	52632	68477	78791	75046	65988	54781	64800
F-4	63530	70462	79698	98290	83881	93959	89988	89156	93992	91357
F-5	77650	62686	81869	76236	83003	66945	67672	80564	82474	68626
F-6	130556	165826	159545	136131	143965	133248	139001	142567	139594	141629
F-7	176557	151963	162312	157181	148705	151316	154617	156349	152538	152792
F-8	148210	149535	132501	129232	165086	162856	158816	129635	158539	148761
F-9	155037	135902	174373	160556	175532	144108	143890	157710	161605	154978
F-10	138111	184909	164570	153846	135967	137157	152253	152561	149962	143350
F-11	174092	148428	133239	164039	197989	141030	139341	154033	182133	174252
F-12	392160	666016	322891	390555	358309	346832	610003	328074	579363	379828
F-13	319253	370097	479013	339685	435049	346416	393132	466330	441727	420720
F-14	768306	743312	1118386	675001	1649677	992214	1570618	927760	1578262	1086829
F-15	1143203	1452021	1158921	1367947	1342956	1448888	1444527	1168478	1225489	1269744

Table D.4: the results of 10 runs of CBA+BNSN(1)+BRO using colony size of 40 bees

Datasets	Colony	1	2	3	4	5	6	7	8	9	10
Berlin52	40	7542	7542	7542	7542	7542	7542	7542	7542	7542	7542
KroA100	40	21353	21315	21282	21282	21292	21292	21577	21292	21292	21292
Pr144	40	58570	58570	58537	58537	58570	58554	58570	58570	58570	58570
Ch150	40	6559	6553	6556	6569	6555	6559	6569	6600	6553	6562
KroB150	40	26268	26147	26130	26339	26143	26143	26198	26248	26130	26143
Pr152	40	73818	73818	73818	73818	73682	73818	73818	73682	73682	73682
Rat195	40	2349	2352	2373	2352	2361	2331	2354	2358	2355	2340
D198	40	15815	15829	15814	15813	15830	15831	15837	15816	15856	15826
KroA200	40	29873	29805	29492	29508	29555	29489	29581	29486	29578	29514
Ts225	40	126726	126713	126713	126726	126828	126643	126713	126643	126713	126713
Pr226	40	80414	80426	80373	80823	80377	80765	80397	80426	80679	80444
Pr299	40	48633	48979	48812	49062	48318	48386	48667	48304	48297	48974
Lin318	40	42288	42544	42219	42263	42371	42454	42256	42253	42670	42585
Pcb442	40	51589	51258	51316	51402	51036	51136	51207	51046	51037	51228
Fl1577	40	22671	22524	22524	22410	22725	22648	22401	22482	22524	22410

Table D.5: The results of 10 runs of CBA+BNSN(2)+BRO using colony size of 40 bees

Datasets	Colony	1	2	3	4	5	6	7	8	9	10
Berlin52	40	7542	7542	7542	7542	7542	7542	7542	7542	7542	7542
KroA100	40	21343	21317	21320	21305	21343	21343	21315	21320	21318	21282
Pr144	40	58702	58587	58607	58607	58656	58590	58623	58656	58607	58537
Ch150	40	6549	6574	6592	6549	6558	6550	6556	6564	6574	6528
KroB150	40	26381	26273	26204	26217	26231	26265	26299	26332	26252	26130
Pr152	40	73871	74153	73818	73822	73898	73902	73822	73818	74029	73682
Rat195	40	2345	2332	2343	2336	2344	2339	2343	2352	2340	2332
D198	40	15862	15865	15887	15850	15887	15906	15918	15862	15842	15842
KroA200	40	29579	29664	29538	29565	29589	29480	29574	29801	29642	29480
Ts225	40	126880	126977	127014	127020	126937	126977	126977	127007	127007	126880
Pr226	40	80946	80567	80549	80459	80604	80700	80549	80684	80950	80459
Pr299	40	48792	49152	48804	48938	48611	48691	48400	48492	48738	48400
Lin318	40	42837	42787	43089	42905	42807	43157	43011	42836	43090	42787
Pcb442	40	51207	51883	52072	51886	52240	51907	51311	51750	51940	51750
Fl1577	40	22818	22725	22363	22646	22852	22723	22743	22880	22717	22730

Table D.6: The results of 10 runs of CBA+BNSN(3)+BRO using colony size of 40 bees

Datasets	Colony	1	2	3	4	5	6	7	8	9	10
Berlin52	40	7542	7542	7542	7542	7542	7542	7542	7542	7542	7542
KroA100	40	21305	21282	21282	21282	21292	21292	21292	21282	21292	21630
Pr144	40	58623	58537	58590	58603	58590	58590	58623	58554	58570	58570
Ch150	40	6588	6562	6564	6544	6553	6563	6558	6528	6559	6579
KroB150	40	26130	26130	26252	26233	26148	26374	26192	26374	26141	26141
Pr152	40	73682	73682	73682	73682	73818	73682	73682	73682	73682	73682
Rat195	40	2337	2344	2328	2332	2323	2342	2346	2329	2339	2351
D198	40	15818	15826	15846	15860	15823	15807	15829	15809	15842	15793
KroA200	40	29926	29585	29944	29516	29576	29459	29529	29973	29827	29382
Ts225	40	126796	126783	126713	126726	126713	126713	126866	126713	126713	126713
Pr226	40	80923	80534	80373	80373	80373	80773	80373	80373	80373	80373
Pr299	40	48281	49072	48397	48636	48386	48336	48352	48606	48195	48256
Lin318	40	42266	42588	42434	42412	42354	42416	42481	42149	42334	42306
Pcb442	40	51236	51311	51436	51417	51501	51634	51460	51567	51671	51130
Fl1577	40	22484	22527	22363	22427	22513	22575	22408	22710	22397	22417

Table D.7: The NFE results of 10 runs of CBA+BNSN(1)+BRO using a colony size of 40 bees

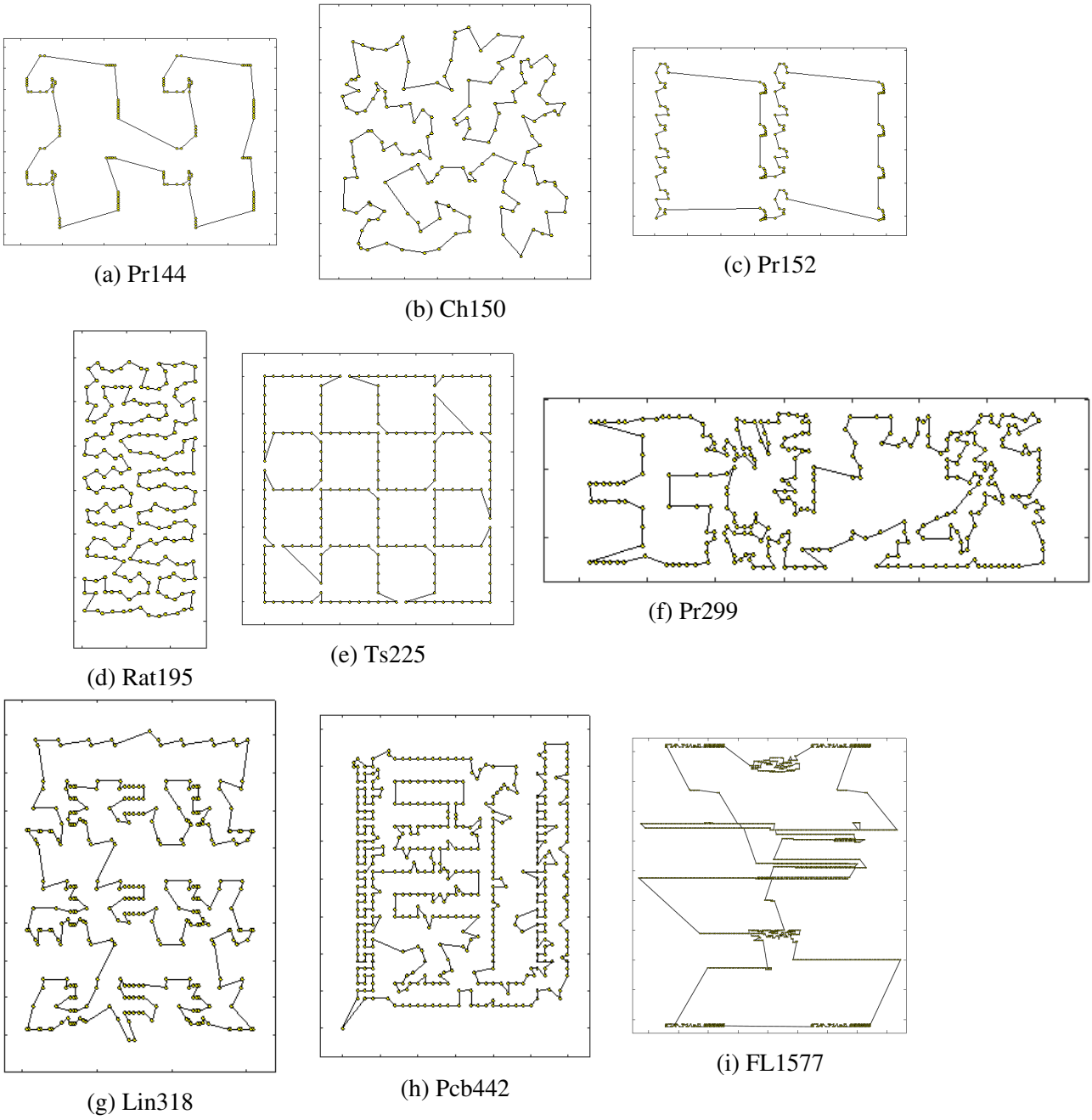
Datasets	Colony	1	2	3	4	5	6	7	8	9	10
Berlin52	40	3658	3000	905	8998	3855	3585	1086	2615	2203	4998
KroA100	40	50330	139399	2108	51833	51674	78554	78309	58921	105536	159399
Pr144	40	37165	3121	17263	48601	8039	25466	41205	34253	44932	58601
Ch150	40	116705	512159	219206	53923	293979	429096	395472	166366	490083	612159
KroB150	40	106170	413683	141855	49747	56883	124978	367935	380873	195831	419683
Pr152	40	50179	75912	174390	46290	48585	67942	99822	59173	75034	211390
Rat195	40	318767	557047	270765	59963	482924	249016	136625	90417	152548	612047
D198	40	794583	359712	243391	48512	681865	724550	679257	223929	628253	849583
KroA200	40	452509	596977	391641	49465	64924	543843	563105	276660	420690	965977
Ts225	40	50170	94445	74725	52230	71363	57519	62064	49642	69645	94825
Pr226	40	529243	678207	796706	48299	231370	101788	425867	543562	615798	769706
Pr299	40	979819	267161	721717	51745	56203	458566	384461	760252	351410	799819
Lin318	40	428223	500444	860403	48891	431851	305897	203886	310959	370568	706403
Pcb442	40	681718	724638	593426	51803	612182	132968	140899	336870	574814	724638
Fl1577	40	620139	731539	751020	769697	833270	556022	499151	738764	728760	797569

Table D.8: The NFE results of 10 runs of CBA+BNSN(2)+BRO using a colony size of 40 bees

Datasets	Colony	1	2	3	4	5	6	7	8	9	10
Berlin52	40	1832	2906	11115	4404	2896	1879	3572	6353	8136	8123
KroA100	40	209675	17630	5700	249060	741512	95312	65956	61277	725700	75500
Pr144	40	2833	5864	50443	463334	31837	3985	6879	85002	25774	92839
Ch150	40	713994	592145	160262	170319	26080	251670	25180	527331	694732	65186
KroB150	40	75820	647557	773975	256722	73580	465260	435923	795528	157533	93583
Pr152	40	92466	739518	59324	466276	77868	25206	26888	472644	348037	65205
Rat195	40	140980	218367	289995	488617	98215	578406	627545	458482	569728	98215
D198	40	497388	615265	590882	742551	167387	154441	247908	547344	358732	414441
KroA200	40	784216	136439	329207	712656	477938	728177	42855	289424	243774	42855
Ts225	40	41125	13225	15700	25615	24940	31099	24490	126965	74275	92145
Pr226	40	160595	570922	417297	677104	640257	642501	392207	414570	658146	160595
Pr299	40	725768	730012	132364	469637	751319	727079	746534	660133	585268	332164
Lin318	40	752849	551527	555163	700004	661282	86638	560189	296508	122284	686638
Pcb442	40	446978	785071	731339	731678	442213	789442	565616	467864	390921	440291
Fl1577	40	578801	774935	749901	556895	725766	790126	331561	592329	741856	727070

Table D.9: The NFE results of 10 runs of CBA+BNSN(3)+BRO using colony size of 40 bees

Datasets	Colony	1	2	3	4	5	6	7	8	9	10
Berlin52	40	2245	3936	811	6252	1828	2812	1542	980	2421	3462
KroA100	40	60744	145877	330226	52843	76618	53720	135263	45205	71616	112554
Pr144	40	4994	15391	6890	50475	10119	32186	45236	36780	48315	50475
Ch150	40	253967	679950	188013	50407	219220	400774	134213	444260	88399	679950
KroB150	40	85788	100915	686006	50187	378785	269989	188954	384433	169340	686006
Pr152	40	33733	40598	67993	30672	45733	227419	40070	66414	27319	227419
Rat195	40	271960	527317	666163	59500	78600	99585	315818	135558	138656	666163
D198	40	414007	566775	265075	48167	722909	269900	460063	58393	749995	749995
KroA200	40	155289	384555	466620	51301	280530	50675	471521	569626	467467	569626
Ts225	40	26290	49045	232537	50007	51277	72641	45829	55941	51109	232537
Pr226	40	261708	363857	427182	497498	506877	576492	485392	494281	425802	576492
Pr299	40	167078	743999	461528	498333	139139	537155	91489	413192	624290	743999
Lin318	40	418717	574214	123238	475100	450297	688524	215195	695388	334154	695388
Pcb442	40	588457	271054	464038	516400	483232	452143	519122	193702	629231	629231
Fl1577	40	541974	712769	653641	419001	764569	874360	481698	743922	641524	697468



(a) Pr144 (b) Ch150 (c) Pr152

(d) Rat195 (e) Ts225 (f) Pr299

(g) Lin318 (h) Pcb442 (i) FL1577

Figure D.1: Result of the best solution of CBA+BRO for selected [144,1577] TSPLIB's datasets

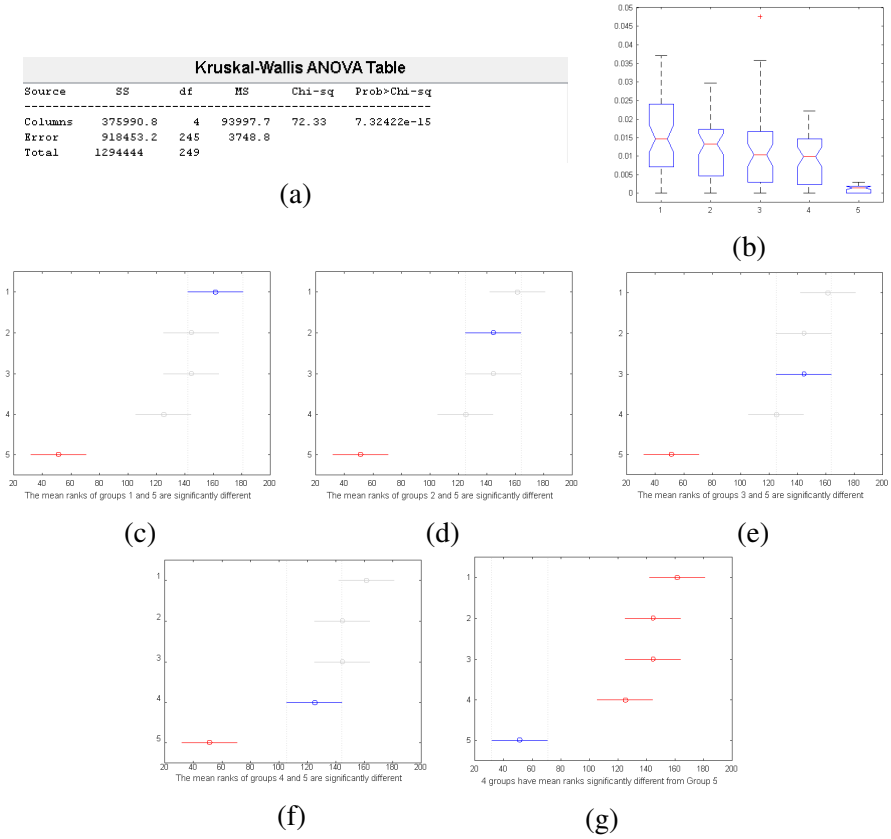


Figure D.2: Statistic Kruskal Wallis test of Exploration Vs Exploitation strategy (Accuracy) [50,76] datasets, (1 = CBA; 2 = CBA+NNH; 3 = CBA+BNSN(2); 4 = CBA+BNSN(3); 5 = CBA+BRO)

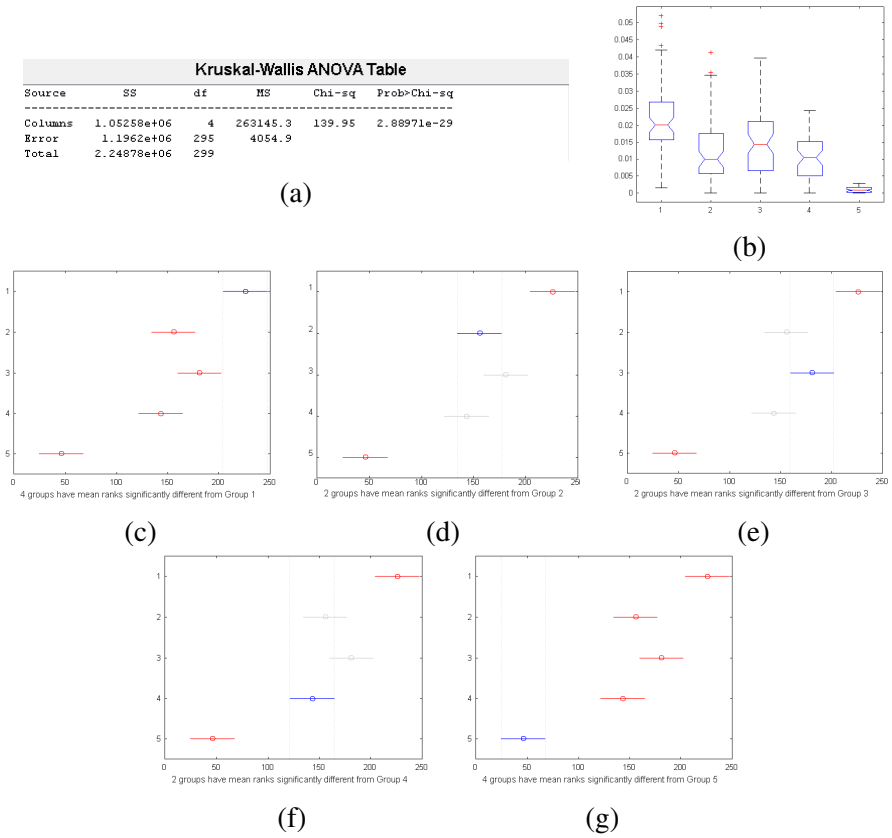


Figure D.3: Statistic Kruskal Wallis test of Exploration Vs Exploitation strategy (Accuracy) [99,100] datasets, (1 = CBA; 2 = CBA+NNH; 3 = CBA+BNSN(2); 4 = CBA+BNSN(3); 5 = CBA+BRO)

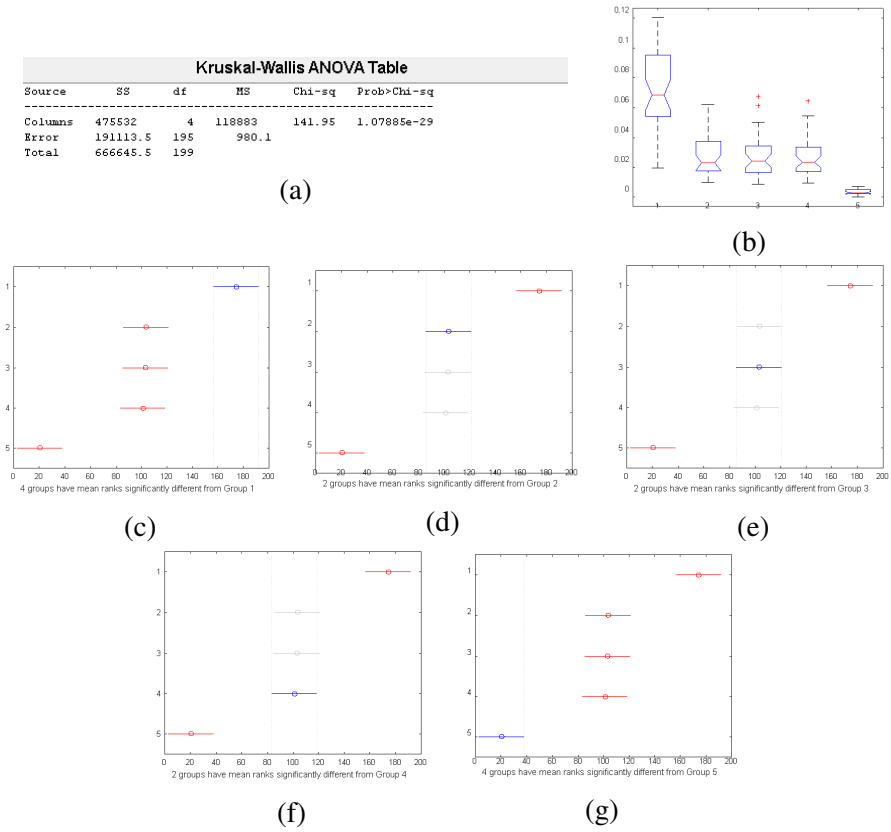


Figure D.4: Statistic Kruskal Wallis test of Exploration Vs Exploitation strategy (Accuracy) [150,200] datasets, (1 = CBA; 2 = CBA+NNH; 3 = CBA+BNSN(2); 4 = CBA+BNSN(3); 5 = CBA+BRO)

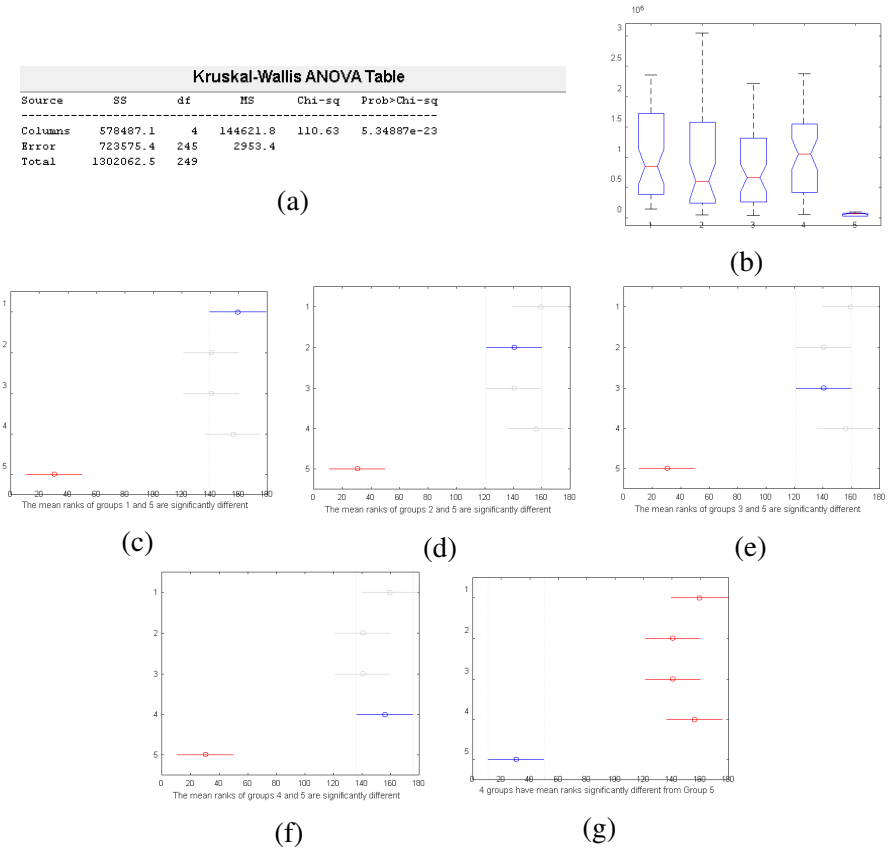


Figure D.5: The Kruskal Wallis test of Exploration Vs Exploitation strategy (NFE) [50,76] datasets, (1 = CBA; 2 = CBA+NNH; 3 = CBA+BNSN(2); 4 = CBA+BNSN(3); 5 = CBA+BRO)

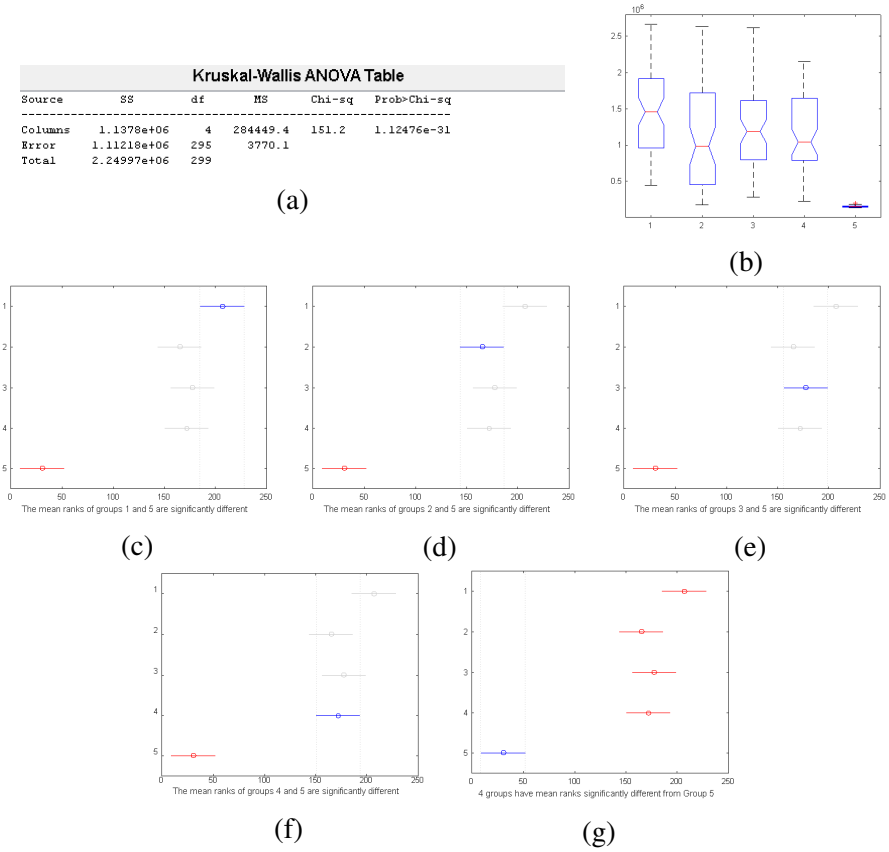


Figure D.6: The Kruskal Wallis test of Exploration Vs Exploitation strategy (NFE) [99,100] datasets, (1 = CBA; 2 = CBA+NNH; 3 = CBA+BNSN(2); 4 = CBA+BNSN(3); 5 = CBA+BRO)

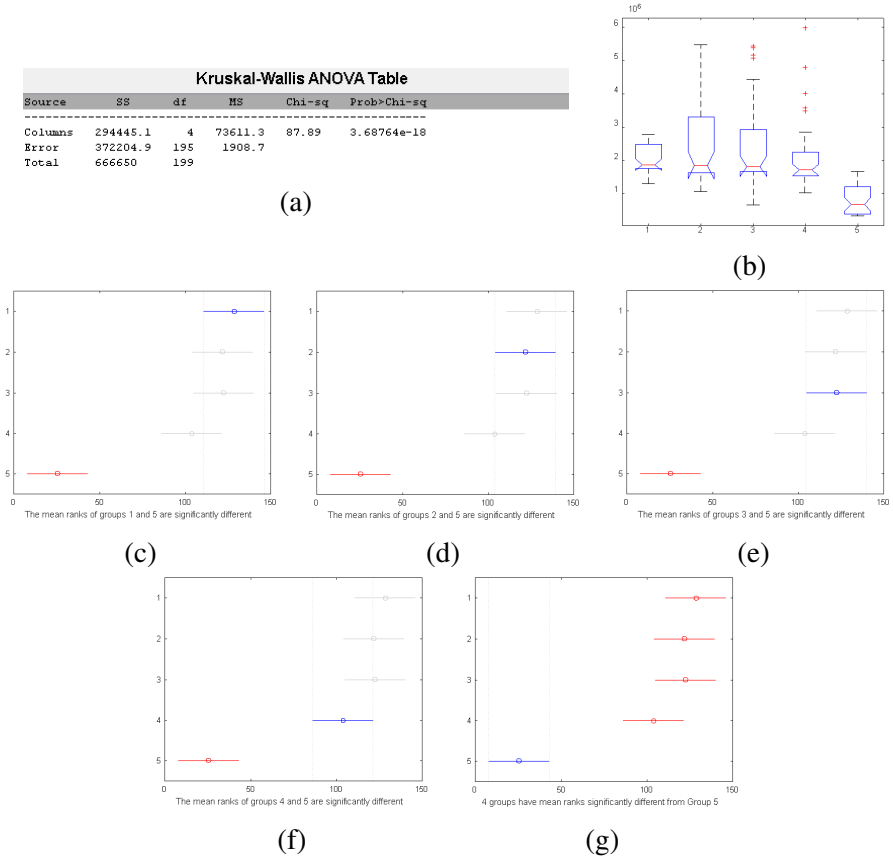


Figure D.7: The Kruskal Wallis test of Exploration Vs Exploitation strategy (NFE) [150,200] datasets, (1 = CBA; 2 = CBA+NNH; 3 = CBA+BNSN(2); 4 = CBA+BNSN(3); 5 = CBA+BRO)

Appendix E

The results of Chapter 6

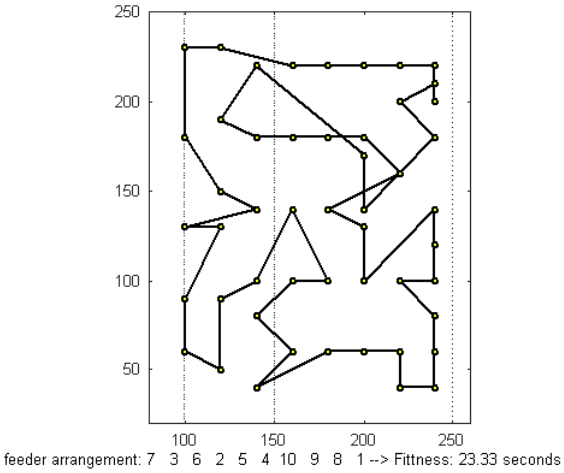


Figure E.1: The result of CBA (seed) with Domino Sequence Heuristic under 23.4 sec

Table E.1: The simulation result of the PCB assembly of 50 components obtained using the Bees algorithm with Domino operators for a CS machine with a twin assembly head turret and 10 component feeders where the total assembly time is 23.33s.

x	y	comp.type	t1	t2	t3	tmax
220	160	9	0.67	0.25	0.25	0.67
180	140	8	0.33	0.00	0.25	0.33
200	130	9	0.50	0.25	0.25	0.50
200	100	9	0.67	0.75	0.25	0.75
240	140	10	0.33	0.25	0.25	0.33
240	120	1	0.33	0.50	0.25	0.50
240	100	8	0.33	0.25	0.25	0.33
220	100	10	0.33	0.50	0.25	0.50
240	80	4	0.33	0.25	0.25	0.33
240	60	9	0.33	0.25	0.25	0.33
240	40	10	0.33	0.00	0.25	0.33
220	40	9	0.33	0.25	0.25	0.33
220	60	9	0.33	0.25	0.25	0.33
200	60	8	0.33	0.00	0.25	0.33
180	60	9	0.67	0.75	0.25	0.75
140	40	9	0.33	0.25	0.25	0.33

Table E.1 continued from previous page

x	y	comp.type	t_1	t_2	t_3	t_{max}
160	60	5	0.33	0.00	0.25	0.33
140	80	2	0.33	0.25	0.25	0.33
160	100	2	0.33	0.00	0.25	0.33
180	100	5	0.67	0.75	0.25	0.75
160	140	5	0.67	0.25	0.25	0.67
140	100	9	0.33	0.25	0.25	0.33
120	90	10	0.67	0.75	0.25	0.75
120	50	4	0.33	0.25	0.25	0.33
100	60	6	0.50	0.25	0.25	0.50
100	90	3	0.67	0.25	0.25	0.67
120	130	6	0.33	0.50	0.25	0.50
100	130	2	0.67	0.50	0.25	0.67
140	140	4	0.33	0.25	0.25	0.33
120	150	9	0.50	0.25	0.25	0.50
100	180	10	0.83	0.50	0.25	0.83
100	230	4	0.33	0.00	0.25	0.33
120	230	9	0.67	0.25	0.25	0.67
160	220	9	0.33	0.75	0.25	0.75
180	220	8	0.33	0.25	0.25	0.33
200	220	4	0.33	0.25	0.25	0.33
220	220	5	0.33	0.25	0.25	0.33
240	220	2	0.33	0.50	0.25	0.50
240	200	6	0.17	0.00	0.25	0.25
240	210	7	0.33	0.00	0.25	0.33
220	200	7	0.33	0.25	0.25	0.33
240	180	7	0.67	0.25	0.25	0.67
200	140	3	0.50	0.00	0.25	0.50
200	170	7	1.00	1.00	0.25	1.00

Table E.1 continued from previous page

x	y	comp.type	t_1	t_2	t_3	t_{max}
140	220	7	0.50	0.50	0.25	0.50
120	190	5	0.33	0.25	0.25	0.33
140	180	10	0.33	0.00	0.25	0.33
160	180	4	0.33	0.25	0.25	0.33
180	180	4	0.33	0.25	0.25	0.33
200	180	10	0.33	0.25	0.25	0.33
						23.33333

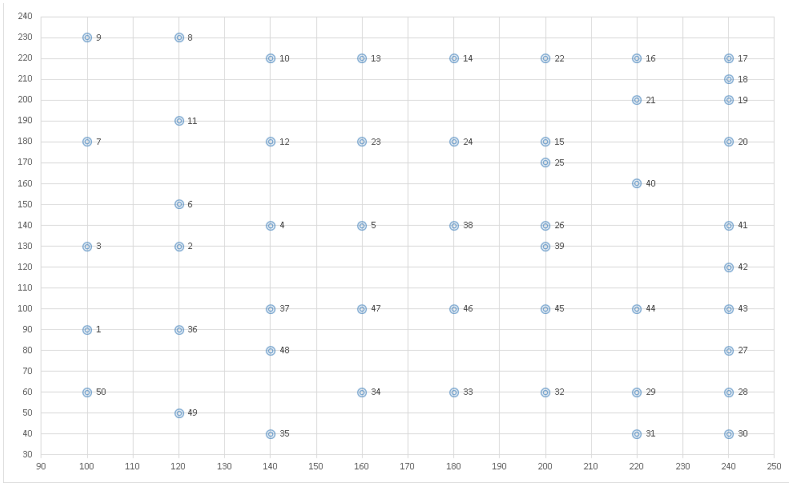


Figure E.2: The 50 placement locations

Table E.2: Bees Algorithm + BNSN + Domino (1-15 running experiments)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T	25.7	24.4	26.0	25.5	24.7	25.8	25.3	25.2	26.8	26.9	26.2	25.5	23.9	25.9	26.4
F1	5	7	7	7	7	7	5	7	8	9	8	3	7	8	8
F2	2	3	8	4	3	4	4	4	9	8	7	7	3	9	2
F3	3	6	6	5	6	5	10	5	10	10	3	4	6	10	5
F4	6	2	4	2	2	2	9	2	5	2	6	6	2	5	6
F5	4	5	5	3	5	6	2	6	2	5	5	5	5	6	3
F6	10	4	2	6	4	3	6	3	6	4	2	2	4	2	9

Table E.2 continued from previous page

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
F7	7	10	3	9	10	9	3	9	3	6	4	10	10	3	10
F8	9	9	9	10	9	10	7	10	4	3	10	9	9	4	4
F9	8	8	10	8	8	8	8	8	7	7	9	8	8	7	7
F10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C1	50	19	50	35	21	26	30	49	40	17	49	17	35	33	2
C2	1	18	2	34	20	39	31	34	41	19	50	15	49	35	47
C3	48	21	4	49	26	38	33	48	26	18	1	14	50	34	32
C4	34	20	5	48	25	12	49	47	42	21	2	13	1	48	43
C5	46	26	26	47	10	7	50	46	27	20	3	8	3	47	38
C6	47	25	39	46	9	8	1	5	29	25	4	9	2	26	5
C7	26	10	45	26	8	13	2	4	28	42	6	10	47	42	46
C8	5	11	29	40	6	14	3	3	30	26	7	24	48	27	26
C9	2	7	31	41	2	15	6	2	31	27	12	25	34	4	39
C10	3	9	30	42	3	19	4	1	32	43	9	26	46	2	44
C11	11	8	28	43	11	17	5	36	33	28	8	27	45	3	45
C12	9	12	44	44	7	16	24	37	44	30	13	28	39	1	29
C13	8	38	42	45	4	22	16	38	43	29	15	30	38	49	30
C14	10	39	41	39	5	21	22	26	45	31	24	31	37	50	31
C15	13	45	40	38	46	20	23	39	46	32	23	29	36	36	28
C16	14	46	15	37	45	18	12	45	47	33	22	32	5	37	27
C17	38	47	13	36	44	25	7	44	48	35	16	33	4	38	42
C18	25	48	8	50	39	24	11	43	34	34	17	35	6	5	41
C19	24	34	9	1	38	23	9	42	36	49	19	34	7	6	40
C20	23	36	10	3	37	10	8	41	37	50	18	48	9	7	19
C21	4	37	14	2	36	9	13	40	35	1	20	50	8	11	17
C22	7	6	22	6	49	11	10	15	50	2	21	49	13	12	16
C23	6	4	16	12	50	4	14	14	49	4	14	1	14	13	14

Table E.2 continued from previous page

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C24	12	5	17	7	1	5	21	13	1	5	10	2	22	14	11
C25	22	2	19	11	48	46	20	12	3	23	11	4	16	8	7
C26	16	3	18	10	47	48	18	6	2	24	5	5	17	9	8
C27	17	1	20	9	34	35	19	7	4	15	38	12	19	10	9
C28	19	50	21	8	35	34	17	8	23	41	25	38	18	23	10
C29	18	49	25	13	33	49	15	9	9	40	26	39	20	24	13
C30	20	35	38	14	32	50	40	10	10	38	43	44	21	25	22

Table E.3: Bees Algorithm + BNSN + Domino (16-30 running experiments)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
T	23.3	25.5	24.5	26.3	25.8	25.3	23.9	24.7	26.4	27.8	24.0	23.6	25.2	23.0	24.7
F1	7	7	7	10	8	7	7	7	7	9	7	7	7	3	7
F2	3	4	3	9	4	4	3	3	4	10	3	3	4	6	3
F3	6	5	6	2	5	5	6	6	5	5	6	6	5	7	6
F4	2	2	2	5	2	2	2	2	2	2	2	2	2	4	2
F5	5	6	5	4	6	6	5	5	6	3	5	4	6	2	5
F6	4	3	4	6	3	3	4	4	3	6	4	5	3	5	4
F7	10	9	10	3	9	9	10	10	9	4	10	9	9	9	10
F8	9	10	9	7	10	10	9	9	10	7	9	10	10	10	9
F9	8	8	8	8	7	8	8	8	8	8	8	8	8	8	8
F10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C1	40	33	3	9	48	46	35	21	39	16	42	9	49	38	21
C2	38	44	1	10	46	4	49	20	38	15	43	8	34	39	20
C3	39	43	50	14	26	5	50	26	37	13	44	11	48	5	26
C4	45	42	49	22	39	27	1	25	36	8	27	7	47	4	25
C5	41	41	48	16	45	28	3	10	1	9	28	6	46	2	10
C6	42	40	47	17	44	29	2	9	2	23	30	4	5	1	9
C7	43	15	5	19	42	30	47	8	3	12	31	3	4	50	8
C8	44	38	26	18	41	31	48	6	4	11	29	2	3	49	6
C9	27	39	25	21	40	32	34	2	5	7	32	47	2	48	2

Table E.3 continued from previous page

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
C10	28	26	21	20	25	33	46	3	24	6	33	46	1	47	3
C11	30	45	20	26	15	45	45	11	25	3	35	45	36	46	11
C12	31	46	18	27	21	44	39	7	18	2	34	27	37	45	7
C13	29	47	19	28	20	43	38	4	20	48	48	28	38	44	4
C14	32	49	17	30	18	42	37	5	21	49	47	29	26	43	5
C15	33	50	16	31	19	41	36	46	22	50	5	31	39	42	46
C16	35	1	22	29	17	40	5	45	16	1	25	30	45	41	45
C17	34	36	14	45	16	26	4	44	17	4	10	32	44	40	44
C18	48	37	13	44	22	39	6	39	19	26	9	33	43	15	39
C19	47	35	8	39	14	38	7	38	40	39	8	34	42	14	38
C20	46	34	9	41	24	15	9	37	15	40	13	35	41	13	37
C21	5	48	10	40	23	13	8	36	13	41	14	49	40	8	36
C22	37	3	11	15	11	14	13	49	14	45	22	50	15	9	49
C23	36	2	7	13	9	12	14	50	12	46	16	1	14	10	50
C24	49	6	6	12	8	8	22	1	6	47	17	48	13	23	1
C25	50	7	12	11	13	9	16	48	7	5	19	37	12	24	48
C26	1	12	23	23	10	10	17	47	8	37	18	36	6	22	47
C27	2	14	24	4	12	23	19	34	9	36	21	5	7	16	34
C28	3	13	15	5	7	24	18	35	11	35	20	38	8	17	35
C29	4	8	40	24	6	25	20	33	10	34	26	39	9	18	33
C30	6	9	41	25	2	21	21	32	23	33	46	44	10	19	32

Table E.4: Bees Algorithm + BNSN (1-15 running experiments)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T	24.7	27.3	25.5	28.4	26.0	26.3	27.7	25.5	26.7	27.3	26.7	24.9	26.9	25.8	26.7
F1	7	7	7	8	9	8	8	8	7	7	8	7	8	8	7
F2	3	4	4	7	8	10	3	10	9	4	9	9	9	9	8
F3	6	5	5	10	10	9	7	6	8	5	10	8	10	10	2
F4	2	2	10	4	5	2	5	2	5	10	5	10	3	5	5
F5	4	6	9	5	4	6	4	3	10	2	2	5	6	2	4
F6	5	10	3	2	2	3	2	9	2	6	6	4	2	6	10

Table E.4 continued from previous page

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
F7	10	3	2	6	6	4	6	4	6	3	3	2	7	3	6
F8	9	9	6	3	3	5	10	7	3	9	4	6	4	4	3
F9	8	8	8	9	7	7	9	5	4	8	7	3	5	7	9
F10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C1	8	25	7	38	25	11	45	44	6	8	5	23	8	8	11
C2	13	21	6	25	21	10	44	27	7	13	38	24	10	9	10
C3	15	22	3	15	20	24	33	30	11	14	39	15	9	10	14
C4	24	20	2	24	18	25	35	31	12	40	40	41	11	23	22
C5	23	18	47	23	19	42	37	28	5	15	41	40	23	24	16
C6	12	16	37	12	17	27	39	29	38	19	15	25	24	25	17
C7	11	17	48	4	16	28	42	32	39	17	14	21	16	20	18
C8	7	19	50	7	22	29	41	33	45	16	13	20	22	21	20
C9	6	15	1	9	10	30	40	35	46	22	11	18	21	22	21
C10	4	40	36	8	9	31	15	49	47	21	7	16	20	18	25
C11	2	26	35	13	8	32	13	50	2	18	6	17	18	19	38
C12	3	39	49	11	13	33	12	1	4	20	37	19	17	17	5
C13	1	38	34	10	14	35	23	3	9	25	36	26	19	16	4
C14	50	14	33	14	12	34	24	2	8	24	34	42	26	15	2
C15	49	13	44	22	11	49	5	5	10	23	35	27	40	40	36
C16	48	8	27	16	7	50	38	38	13	10	33	44	41	41	1
C17	47	12	29	18	6	1	26	37	14	11	32	43	44	39	37
C18	5	23	28	21	37	48	25	36	16	12	29	28	43	45	33
C19	38	10	30	20	36	36	20	4	18	7	31	30	28	46	35
C20	37	9	31	41	34	37	19	6	21	6	30	31	30	5	50
C21	36	11	32	43	35	47	17	7	20	2	28	29	31	38	49
C22	34	7	43	27	33	45	22	11	40	3	43	32	29	37	34
C23	35	6	42	44	32	43	16	23	41	36	44	33	32	36	32
C24	33	2	38	32	30	44	18	24	44	49	45	35	33	47	43
C25	32	47	26	30	31	38	21	15	43	34	46	34	45	26	27
C26	31	48	39	46	29	39	14	41	25	48	47	36	39	42	30
C27	28	50	45	45	28	40	10	40	15	47	48	37	38	27	31

Table E.4 continued from previous page

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C28	29	49	46	28	45	41	11	25	17	41	50	38	12	44	29
C29	30	34	5	29	43	15	9	39	19	42	49	39	7	43	28
C30	27	46	4	31	27	17	8	45	22	43	1	45	6	28	45

Table E.5: Bees Algorithm + BNSN (16-30 running experiments)

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
T	26.2	26.8	24.4	26.8	26.3	26.3	25.8	26.0	26.3	26.4	26.3	26.7	25.5	26.8	27.7
F1	7	8	7	8	9	7	8	7	7	3	8	7	8	8	8
F2	8	7	3	9	3	9	5	10	4	7	10	8	10	7	3
F3	9	3	6	3	6	3	4	5	5	8	9	2	6	3	7
F4	10	5	2	2	4	6	2	2	2	9	2	5	2	5	5
F5	5	4	5	6	5	2	6	6	6	10	6	4	3	4	4
F6	2	6	4	4	2	5	3	3	3	6	3	10	9	6	2
F7	6	2	10	5	7	4	9	4	9	2	4	6	4	2	6
F8	3	10	9	10	10	8	10	9	8	4	5	3	7	10	10
F9	4	9	8	7	8	10	7	8	10	5	7	9	5	9	9
F10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C1	18	23	19	7	24	38	40	46	27	12	11	11	44	23	45
C2	20	24	18	10	23	46	15	44	28	6	10	10	27	24	44
C3	40	5	21	12	22	32	21	30	29	7	24	14	30	5	33
C4	15	46	20	23	16	30	20	28	30	11	25	22	31	46	35
C5	16	47	26	24	17	27	18	45	31	9	42	16	28	47	37
C6	17	48	25	15	18	43	19	42	32	8	27	17	29	48	39
C7	19	37	10	25	21	44	17	43	33	10	28	18	32	37	42
C8	22	36	11	36	15	42	16	27	35	13	29	20	33	36	41
C9	24	50	7	1	25	41	22	29	48	14	30	21	35	50	40
C10	23	49	9	35	38	15	14	31	34	21	31	25	49	49	15
C11	9	1	8	50	12	17	11	32	49	20	32	38	50	1	13
C12	8	4	12	49	14	19	12	33	50	26	33	5	1	4	12
C13	10	2	38	34	10	16	7	35	1	38	35	4	3	2	23
C14	13	3	39	46	7	22	6	49	36	39	34	2	2	3	24

Table E.5 continued from previous page

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
C15	14	6	45	47	11	14	2	50	37	40	49	36	5	6	5
C16	12	7	46	48	9	24	3	1	6	25	50	1	38	7	38
C17	11	8	47	37	8	23	1	4	7	18	1	37	37	8	26
C18	7	9	48	6	13	12	36	5	12	19	48	33	36	9	25
C19	6	11	34	3	6	7	37	12	14	17	36	35	4	11	20
C20	3	10	36	2	3	11	33	11	13	16	37	50	6	10	19
C21	2	14	37	4	36	9	35	7	8	22	47	49	7	14	17
C22	4	22	6	5	48	8	50	10	9	24	45	34	11	22	22
C23	1	17	4	30	34	10	49	15	10	5	43	32	23	17	16
C24	49	19	5	31	47	13	48	25	11	4	44	43	24	19	18
C25	50	16	2	28	46	25	34	41	4	2	38	27	15	16	21
C26	48	18	3	29	45	21	32	20	2	3	39	30	41	18	14
C27	37	20	1	32	39	18	43	21	3	48	40	31	40	20	10
C28	36	21	50	33	26	20	27	18	5	34	41	29	25	21	11
C29	5	25	49	45	40	40	28	16	25	49	15	28	39	25	9
C30	26	38	35	38	19	26	30	17	24	50	17	45	45	38	8

Table E.6: CBA + Bi-BA + BRO for VRP (10 running experiments)

Eil33		Eil51		A76		B76		C76		D76		A101		B101	
Cost	Eval	Cost	Eval	Cost	Eval	Cost	Eval	Cost	Eval	Cost	Eval	Cost	Eval	Cost	Eval
835	82,391	523	4,946	861	7,487	1,073	409,405	763	123,511	708	6,226	843	173,404	1,128	1,188
859	186,820	523	2,522	876	245,255	1,076	138,044	765	8,523	716	1,631	854	14,299	1,127	172,219
866	92,660	521	17,035	866	20,345	1,132	204,564	761	45,828	708	26,155	854	14,445	1,147	39,461
835	1,289	527	55,941	869	38,657	1,083	149,912	770	19,739	703	211,439	843	209,263	1,134	173,370
835	1,768	532	29,316	863	17,577	1,090	119,958	761	96,210	713	202,524	859	101,218	1,152	624
837	6,181	534	18,041	874	151,448	1,064	81,194	762	40,547	721	12,365	859	113,017	1,147	80,786
837	86,517	522	54,069	861	4,497	1,132	15,962	772	114,109	722	17,292	841	218,628	1,144	189,398
835	7,081	524	10,942	877	136,709	1,076	210,415	768	13,389	709	2,946	848	176,969	1,144	2,617
863	24,849	528	54,373	874	106,039	1,064	183,626	755	98,034	705	58,097	846	149,933	1,130	174,085
835	170,749	534	53,656	872	53,862	1,132	720	766	16,886	721	6,184	845	101,218	1,133	142,090
843.7	66,030.5	526.8	30,084.1	869.3	78,187.6	1,092.2	151,380.0	764.3	57,677.6	712.6	54,485.9	849.2	127,239.4	1,138.6	97,583.8

Appendix F

MATLAB code of Bi-Parameters Bees Algorithm (Bi-BA) for numerical function

```
clc;
clear;
close all;
tic
%% Problem Definition
[typeOfFunction] = 'Rastrigin_10';
Instance=TestFunction38(typeOfFunction);
Dims=Instance.dim;
ObjFunction=@(x) Instance.evaluation( x ); % Objective Function
VarSize=[1 Dims]; % Decision Variables Matrix Size
VarMin=Instance.lowerBoundaries; % Decision Variables Lower Bound
VarMax=Instance.upperBoundaries; % Decision Variables Upper Bound
range=VarMax-VarMin;
%% Bees Algorithm Parameters
n = 7; nep = 30; Shrink = 0.99; stlim = 20;
MaxEval = 500000; accuracy = 0.001;
recruitment = round(linspace(nep,1,n));
assigntment = linspace(0,1,n);
ColonySize=sum(recruitment);
MaxIt=round(MaxEval/ColonySize);
%% Initialization
Empty_Bees.Position=[];
Empty_Bees.Cost=[];
Empty_Bees.Size=[];
Empty_Bees.Stagnated =[];
Empty_Bees.counter=[];
Bees=repmat(Empty_Bees,n,1);
counter=0;
% Generate Initial Solutions
for i=1:n
    Bees(i).Position=unifrnd(VarMin,VarMax,VarSize);
    Bees(i).Cost=ObjFunction(Bees(i).Position);
    Bees(i).Size = range;
    Bees(i).Stagnated = 0;
    counter=counter+1;
    Bees(i).counter= counter;
end
size = linspace(0,1,n);
%% Sites Selection
[~, RankOrder]=sort([Bees.Cost]);
Bees=Bees(RankOrder);
BestSol.Cost=inf;
P=1;
%% Bees Algorithm Local and Global Search
for it=1:MaxIt
    if counter >= MaxEval
        break;
    end
    % All Sites (Exploitation and Exploration)
    for i=1:n
        bestnewbee.Cost=inf;
        assigntment=D_Triangular_real(0,size(i),1,1,recruitment(i));
        for j=1:recruitment(i)
            if P==1
                newbee.Position= Integrated_Foraging_stlim_unif(Bees(i).Position,↙
```

```
assigntment(j),VarMax,VarMin,Bees(i).Size);
            else
                newbee.Position= Integrated_Foraging_stlim(Bees(i).Position,↙
assigntment(j),VarMax,VarMin,Bees(i).Size);
            end
            newbee.Cost=ObjFunction(newbee.Position);
            newbee.Size= Bees(i).Size;
            newbee.Stagnated = Bees(i).Stagnated;
            counter=counter+1;
            newbee.counter= counter;
            if newbee.Cost<bestnewbee.Cost
                bestnewbee=newbee;
            end
        end
        if bestnewbee.Cost<Bees(i).Cost
            Bees(i)=bestnewbee;
            Bees(i).Stagnated=0;
        else
            Bees(i).Stagnated=Bees(i).Stagnated+1;
            Bees(i).Size=Bees(i).Size*Shrink;
        end
        %site abandonment procedure
        if(Bees(i).Stagnated>stlim)
            Bees(i)=Bees(end);
            Bees(i).Size=range;
            Bees(i).Stagnated=0;
            P=P*-1;
        end
    end
    % SORTING
    [~, RankOrder]=sort([Bees.Cost]);
    Bees=Bees(RankOrder);
    % Update Best Solution Ever Found
    OptSol=Bees(1);
    if OptSol.Cost < BestSol.Cost
        BestSol=OptSol;
    end
    % taking of result
    OptCost(it)=BestSol.Cost;
    Counter(it)=counter;
    Time(it)=toc;
    % Display Iteration Information
    disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(OptCost(it)) ' --> Time =↙
' num2str(Time(it)) ' seconds' '; Fittness Evaluations = ' num2str(Counter(it))]);
    if(abs(Instance.optima-BestSol.Cost) <= accuracy)
        break;
    end
end
%% Results
figure;
semilogy(OptCost,'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
```

```
function [ M ] = D_Triangular_real(k,t,b,baris,kolom)
    M=zeros(baris,kolom);
    for i=1:baris
        for j=1:kolom
            M(i,j)=D_Tri_real(k,t,b);
        end
    end
end

function [ angka ] = D_Tri_real(k,t,b)
m=randi([1 10]);
    a=(t-k)/10;
    b=(b-t)/10;
    switch m
        case 1
            angka=lapis1(t,a,b);
        case 2
            angka=lapis2(t,a,b);
        case 3
            angka=lapis3(t,a,b);
        case 4
            angka=lapis4(t,a,b);
        case 5
            angka=lapis5(t,a,b);
        case 6
            angka=lapis6(t,a,b);
        case 7
            angka=lapis7(t,a,b);
        case 8
            angka=lapis8(t,a,b);
        case 9
            angka=lapis9(t,a,b);
        case 10
            angka=lapis10(t,a,b);
    end
end

function angka=lapis1(t,a,b)
    angka=unifrnd((t-a),(t+b),1);
end

function angka=lapis2(t,a,b)
    angka=unifrnd((t-2*a),(t+2*b),1);
end

function angka=lapis3(t,a,b)
    angka=unifrnd((t-3*a),(t+3*b),1);
end

function angka=lapis4(t,a,b)
    angka=unifrnd((t-4*a),(t+4*b),1);
end

function angka=lapis5(t,a,b)
    angka=unifrnd((t-5*a),(t+5*b),1);
```

```
end

function angka=lapis6(t,a,b)
    angka=unifrnd((t-6*a),(t+6*b),1);
end

function angka=lapis7(t,a,b)
    angka=unifrnd((t-7*a),(t+7*b),1);
end

function angka=lapis8(t,a,b)
    angka=unifrnd((t-8*a),(t+8*b),1);
end

function angka=lapis9(t,a,b)
    angka=unifrnd((t-9*a),(t+9*b),1);
end

function angka=lapis10(t,a,b)
    angka=unifrnd((t-10*a),(t+10*b),1);
end
```

```
function y=Integrated_Foraging_stlim(x,ass,Vmx,Vmn,size)

    r=ass*size;

    nVar=numel(x);

    k=randi([1 nVar]);

    y=x;
    y(k)=y(k)+ r*((-1)^randi(2));
    y(y>Vmx)=Vmx;
    y(y<Vmn)=Vmn;

end
```

```
function y=Integrated_Foraging_stlim_unif(x,ass,Vmx,Vmn,size)

    r=ass*size;

    nVar=numel(x);

    pert=randi([0 1], 1, nVar);

    y=x;

    y = y + (random('unif',-r,r).*pert);
    y(y>Vmx)=Vmx;
    y(y<Vmn)=Vmn;

end
```

```
classdef TestFunction38
    properties
        type;                   % type of fitness function
        lowerBoundaries;        % minimum x and y coordinates
        upperBoundaries;        % maximum x and y coordinates
        dim;                    % number of parameters of function
        optima;                 % global maximum
        definedFunctions        % functions defined in this class
    end
    methods
        function obj = TestFunction38(typeOfFunction)
            obj.definedFunctions=↙
{'Ackley','Griewangk_10','Griewangk_2','Rastrigin_10','Rosenbrock_10','Rosenbrock_2','↙
Rosenbrock_4',...
                'Goldstein','Martin','Shekel_5','Easom','Schaffer_6','Schwefel_1.↙
2','Sphere','Axis',...
                'Sum_diff_pow','Beale','Colville','Hartmann_1','Hartmann_2',...
                'Levy','Matyas','Perm','Zakharov','Schwefel_2.22','Schwefel_2.21',...
↙
'Quartic','Kowalik','Shekel_7','Shekel_10','Tripod','DeJong_2','Dejong_4',...
                'Alpine','Pathological','Masters','Step','6humpCamelBack',...
                'Michalewicz_5','Michalewicz_10','Branin','Weierstrass',↙
'Trid','Powell','MovedHyper'};
            obj.type=typeOfFunction;
            switch(obj.type)
                case 'Ackley_10',
                    obj.dim=10;
                    obj.optima=evaluation(obj,linspace(0,0,10));
                    obj.lowerBoundaries=-32;
                    obj.upperBoundaries=32;
                case 'DeJong_2',
                    obj.dim=2;
                    obj.optima=evaluation(obj,[1,1]);
                    obj.lowerBoundaries=-1;
                    obj.upperBoundaries=1.5;
                case 'Goldstein',
                    obj.dim=2;
                    obj.optima=evaluation(obj,[0,-1]);
                    obj.lowerBoundaries=-2;
                    obj.upperBoundaries=2;
                case 'Martin',
                    obj.dim=2;
                    obj.optima=evaluation(obj,linspace(5,5,2));
                    obj.lowerBoundaries=0;
                    obj.upperBoundaries=10;
                case 'Griewangk_10',
                    obj.dim=10;
                    obj.optima=evaluation(obj,linspace(0,0,10));
                    obj.lowerBoundaries=-512;
                    obj.upperBoundaries=512;
                case 'Griewangk_2',
                    obj.dim=2;
                    obj.optima=evaluation(obj,linspace(0,0,2));
                    obj.lowerBoundaries=-512;
                    obj.upperBoundaries=512;
```

```
case 'Rastrigin_10',
    obj.dim=10;
    obj.optima=evaluation(obj,linspace(0,0,10));
    obj.lowerBoundaries= -5.12;
    obj.upperBoundaries= 5.12;
case 'Rosenbrock_10',
    obj.dim=10;
    obj.optima=evaluation(obj,linspace(1,1,10));
    obj.lowerBoundaries= -1.2;
    obj.upperBoundaries= 1.2;
case 'Rosenbrock_2',
    obj.dim=2;
    obj.optima=evaluation(obj,linspace(1,1,2));
    obj.lowerBoundaries= -1.2;
    obj.upperBoundaries= 1.2;
case 'Rosenbrock_4',
    obj.dim=4;
    obj.optima=evaluation(obj,linspace(1,1,4));
    obj.lowerBoundaries= -1.2;
    obj.upperBoundaries= 1.2;
case {'Shekel_5','Shekel_7'},
    obj.dim=9;
    obj.optima=evaluation(obj,linspace(4,4,9));
    obj.lowerBoundaries=0;
    obj.upperBoundaries=10;
case 'Shekel_4',
    obj.dim=4;
    obj.optima=evaluation(obj,linspace(4,4,4));
    obj.lowerBoundaries=0;
    obj.upperBoundaries=10;
case 'Shekel_10',
    obj.dim=10;
    obj.optima=evaluation(obj,linspace(4,4,10));
    obj.lowerBoundaries=0;
    obj.upperBoundaries=10;
case 'Easom',
    obj.dim=2;
    obj.optima=evaluation(obj,[pi, pi]);
    obj.lowerBoundaries= -100;
    obj.upperBoundaries= 100;
case 'Schaffer_6',
    obj.dim=2;
    obj.optima=evaluation(obj,[0, 0]);
    obj.lowerBoundaries=-10;
    obj.upperBoundaries=10;
case 'Schwefel_2',
    obj.dim=2;
    obj.optima=evaluation(obj,[420.9687 420.9687]);
    obj.lowerBoundaries=-500;
    obj.upperBoundaries=500;
case 'Schwefel_1.2',
    obj.dim=2;
    obj.optima=evaluation(obj,linspace(0,0,2));
    obj.lowerBoundaries=-65;
    obj.upperBoundaries=65;
```

```
            case 'Schwefel_2.22',
                obj.dim=10;
                obj.optima=evaluation(obj,linspace(0,0,10));
                obj.lowerBoundaries= -10;
                obj.upperBoundaries= 10;
            case 'Schwefel_2.21',
                obj.dim=10;
                obj.optima=evaluation(obj,linspace(0,0,10));
                obj.lowerBoundaries= -100;
                obj.upperBoundaries= 100;
            case {'Axis','Sphere'},
                obj.dim=30;
                obj.optima=evaluation(obj,linspace(0,0,30));
                obj.lowerBoundaries= -5.12;
                obj.upperBoundaries= 5.12;
            case 'Sphere_6',
                obj.dim=6;
                obj.optima=evaluation(obj,linspace(0,0,6));
                obj.lowerBoundaries= -5.12;
                obj.upperBoundaries= 5.12;
            case 'Sphere_10',
                obj.dim=10;
                obj.optima=evaluation(obj,linspace(0,0,10));
                obj.lowerBoundaries= -5.12;
                obj.upperBoundaries= 5.12;
            case 'Sum_diff_pow',
                obj.dim=30;
                obj.optima=evaluation(obj,linspace(0,0,30));
                obj.lowerBoundaries= -1;
                obj.upperBoundaries= 1;
            case 'Beale',
                obj.dim=2;
                obj.optima=evaluation(obj,[3,0.5]);
                obj.lowerBoundaries= -4.5;
                obj.upperBoundaries= 4.5;
            case 'Colville',
                obj.dim=4;
                obj.optima=evaluation(obj,linspace(1,1,4));
                obj.lowerBoundaries= -10;
                obj.upperBoundaries= 10;
            case 'Hartmann_1',
                obj.dim=3;
                obj.optima=evaluation(obj,[0.114614,0.555649,0.852547]);
                obj.lowerBoundaries= 0;
                obj.upperBoundaries= 1;
            case 'Hartmann_2',
                obj.dim=6;
                obj.optima=evaluation(obj,↙
[0.20169,0.150011,0.476874,0.275332,0.311652,0.6573]);%optima=3.32237
                obj.lowerBoundaries= 0;
                obj.upperBoundaries= 1;
            case 'Levy',
                obj.dim=30;
                obj.optima=evaluation(obj,linspace(1,1,30));
                obj.lowerBoundaries= -10;
```

```
        obj.upperBoundaries= 10;
    case 'Matyas',
        obj.dim=2;
        obj.optima=evaluation(obj,linspace(0,0,2));
        obj.lowerBoundaries= -10;
        obj.upperBoundaries= 10;
    case 'Perm',
        obj.dim=4;
        obj.optima=evaluation(obj,[1,2,3,4]);
        obj.lowerBoundaries= -4;
        obj.upperBoundaries= 4;
    case 'Zakharov',
        obj.dim=10;
        obj.optima=evaluation(obj,linspace(0,0,10));
        obj.lowerBoundaries= -5;
        obj.upperBoundaries= 10;
    case 'Quartic',
        obj.dim=30;
        obj.optima=evaluation(obj,linspace(0,0,30));
        obj.lowerBoundaries= -1.28;
        obj.upperBoundaries= 1.28;
    case 'Kowalik',
        obj.dim=4;
        obj.optima=evaluation(obj,[0.19,0.19,0.12,0.14]);
        obj.lowerBoundaries= -5;
        obj.upperBoundaries= 5;
    case 'Tripod',
        obj.dim=2;
        obj.optima=evaluation(obj,[0,-50]);
        obj.lowerBoundaries= -100;
        obj.upperBoundaries= 100;
    case 'Dejong_4',
        obj.dim=2;
        obj.optima=evaluation(obj,[0,0]);
        obj.lowerBoundaries= -2.048;
        obj.upperBoundaries= 2.048;
    case 'Alpine',
        obj.dim=2;
        obj.optima=evaluation(obj,linspace(0,0,2));
        obj.lowerBoundaries= -10;
        obj.upperBoundaries= 10;
    case 'Pathological',
        obj.dim=5;
        obj.optima=evaluation(obj,linspace(0,0,5));
        obj.lowerBoundaries= -100;
        obj.upperBoundaries= 100;
    case 'Masters',
        obj.dim=5;
        obj.optima=evaluation(obj, linspace(0,0,5));
        obj.lowerBoundaries= -5;
        obj.upperBoundaries= 5;
    case '6humpCamelBack',
        obj.dim=2;
        obj.optima=evaluation(obj,[0.0898,-0.7126]);
        obj.lowerBoundaries= -5;
```

```
            obj.upperBoundaries= 5;
        case 'Branin'
            obj.dim=2;
            obj.optima=evaluation(obj,[-pi,12.275]);
            obj.lowerBoundaries= -5;
            obj.upperBoundaries= 10 ;
        case 'Step',
            obj.dim=30;
            obj.optima=evaluation(obj,linspace(0,0,30));
            obj.lowerBoundaries= -100;
            obj.upperBoundaries= 100;
        case 'Weierstrass',
            D=30;
            obj.dim=D;
            obj.optima=evaluation(obj,linspace(0,0,D));
            obj.lowerBoundaries= -100;
            obj.upperBoundaries= 100;
        case 'Michalewicz_10',
            obj.dim=10;
            obj.optima=-9.66015;
            obj.lowerBoundaries=0;
            obj.upperBoundaries= pi;
        case 'Michalewicz_5',
            obj.dim=5;
            obj.optima=-4.687658;
            obj.lowerBoundaries=0;
            obj.upperBoundaries= pi;
        case 'Trid',
            obj.dim=6;
            obj.optima=-50;
            obj.lowerBoundaries= -36;
            obj.upperBoundaries= 36;
        case 'Powell',
            obj.dim=24;
            obj.optima=0;
            obj.lowerBoundaries= -4;
            obj.upperBoundaries= 5;
        case'MovedHyper',
            obj.dim = 20;
            obj.optima = evaluation(obj,linspace(0,0,20));
            obj.lowerBoundaries = -5.12;
            obj.upperBoundaries = 5.12;
        case'MovedHyper_10',
            obj.dim = 10;
            obj.optima = evaluation(obj,linspace(0,0,10));
            obj.lowerBoundaries = -5.12;
            obj.upperBoundaries = 5.12;
      otherwise,
        disp('fitness function not defined');
     end
end
function fitness = evaluation(obj, x)
    global eval;
    eval = eval+1;
    switch (obj.type)
```

```
                case 'Ackley_10',
                    n = 2;
                    a = 20; b = 0.2; c = 2*pi;
                    s1 = 0; s2 = 0;
                    for i=1:n;
                       s1 = s1+x(i)^2;
                       s2 = s2+cos(c*x(i));
                    end
                    y = -a*exp(-b*sqrt(1/n*s1))-exp(1/n*s2)+a+exp(1);
                    fitness=y;
                case 'DeJong_2',
                    x1=x(1);
                    x2=x(2);
                    c1=(3905.93)-100*(x1^2-x2)^2-(1-x1^2);
                    fitness=-c1;
                case 'Goldstein',
                    a = 1+(x(1)+x(2)+1)^2*(19-14*x(1)+3*x(1)^2-14*x(2)+6*x(1)*x(2)+3*x↙
(2)^2);
                    b = 30+(2*x(1)-3*x(2))^2*(18-32*x(1)+12*x(1)^2+48*x(2)-36*x(1)*x↙
(2)+27*x(2)^2);
                    y = a*b;
                    fitness=y;
                case 'Martin',
                    x1=x(1);
                    x2=x(2);
                    z=(x1-x2)^2 + ((x1+x2-10)/3)^2;
                    fitness=z;
                case 'Griewangk_10',
                    n = 10;
                    fr = 4000;
                    s = 0;
                    p = 1;
                    for j = 1:n; s = s+x(j)^2; end
                    for j = 1:n; p = p*cos(x(j)/sqrt(j)); end
                    y = s/fr-p+1;
                    fitness=y;
                case 'Griewangk_2',
                    n = 2;
                    fr = 4000;
                    s = 0;
                    p = 1;
                    for j = 1:n; s = s+x(j)^2; end
                    for j = 1:n; p = p*cos(x(j)/sqrt(j)); end
                    y = s/fr-p+1;
                    fitness=y;
                case 'Rastrigin_10',
                    n = 10;
                    s = 0;
                    for j = 1:n
                        s = s+(x(j)^2-10*cos(2*pi*x(j)));
                    end
                    y = 10*n+s;
                    fitness=y;
                case 'Rosenbrock_10',
                    n = 10;
```

```
        c1 = 0;
        for j = 1:n-1;
            c1 = c1+100*(x(j)^2-x(j+1))^2+(x(j)-1)^2;
        end
        y = c1;
        fitness=y;
    case 'Rosenbrock_2',
        n = 2;
        c1 = 0;
        for j = 1:n-1;
            c1 = c1+100*(x(j)^2-x(j+1))^2+(x(j)-1)^2;
        end
        y = c1;
        fitness=y;
    case 'Rosenbrock_4',
        n = 4;
        c1 = 0;
        for j = 1:n-1;
            c1 = c1+100*(x(j)^2-x(j+1))^2+(x(j)-1)^2;
        end
        y = c1;
        fitness=y;
    case 'Shekel_5',
        A=[ 4 4 4 4;
            1 1 1 1;
            8 8 8 8;
            6 6 6 6;
            3 7 3 7 ];
        C=[ 0.1;0.2;0.2;0.4;0.4 ];
        c1=0;
        for i=1:5
            c1=c1-1/(C(i)+sum((x-A(i,:)).^2));
        end
        fitness=c1;
    case 'Shekel_7',
        A=[ 4 4 4 4;
            1 1 1 1;
            8 8 8 8;
            6 6 6 6;
            3 7 3 7;
            2 9 2 9;
            5 5 3 3];
        C=[ 0.1;0.2;0.2;0.4;0.4;0.6;0.3 ];
        c1=0;
        for i=1:7
            c1=c1-1/(C(i)+sum((x-A(i,:)).^2));
        end
        fitness=c1;
    case 'Shekel_10',
        A=[ 4 4 4 4;
            1 1 1 1;
            8 8 8 8;
            6 6 6 6;
            3 7 3 7;
            2 9 2 9;
```

```
            5 5 3 3;
            8 1 8 1;
            6 2 6 2;
            7 3.6 7 3.6];
        C=[ 0.1;0.2;0.2;0.4;0.4;0.6;0.3;0.7;0.5;0.5 ];
        c1=0;
        for i=1:10
            c1=c1-1/(C(i)+sum((x-A(i,:)).^2));
        end
        fitness=c1;
    case 'Easom',
        y = -cos(x(1))*cos(x(2))*exp(-(x(1)-pi)^2-(x(2)-pi)^2);
        fitness=y;
    case 'Schaffer_6',
        s=sum(x.^2);
        f1=sin(sqrt(s))^2-0.5;
        f2=(1+0.01*s)^2;
        c1=0.5+f1/f2;
        fitness=c1;
    case 'Schwefel_2',
        n = 2;
        s = sum(-x.*sin(sqrt(abs(x))));
        y = 418.9829*n+s;
        fitness=y;
    case 'Schwefel_1.2',
        c1=0;
        for i=1:2
            c1=c1+sum(x(1:i)).^2;
        end
        fitness=c1;
    case 'Schwefel_2.22',
        f1=sum((abs(x)));
        f2=prod((abs(x)));
        fitness=(f1+f2);
    case 'Schwefel_2.21',
        f=abs(x);
        c1=max(f);
        fitness=c1;
    case 'Axis',
        f = zeros(1,10);
            for i=1:10
                f(i)=i*x(i)^2;
            end
        fitness = sum(f);
    case 'Sphere',
        c1=sum(x.^2);
        fitness=c1;
    case 'Sphere_6',
        c1=sum(x.^2);
        fitness=c1;
    case 'Sphere_10',
        c1=sum(x.^2);
        fitness=c1;
    case 'Sum_diff_pow',
        f = zeros (1,30);
```

```
        for i= 1:30
            f(i)=abs(x(i)).^(i+1);
        end
    fitness=sum(f);
case 'Beale',
    f1=(1.5-x(1)*(1-x(2)))^2;
    f2=(2.25-x(1)*(1-x(2)^2))^2;
    f3=(2.625-x(1)*(1-x(2)^3))^2;
    fitness=(f1+f2+f3);
case 'Colville',
    f1=100*(x(2)-x(1)^2)^2+(1-x(1))^2;
    f2=90*(x(4)-x(3)^2)^2+(1-x(3))^2;
    f3=10.1*((x(2)-1)^2+(x(4)-1)^2);
    f4=19.8*(x(2)-1)*(x(4)-1);
    fitness=(f1+f2+f3+f4);
case 'Hartmann_1',
    a=[1;1.2;3;3.2];
    A=[ 3 10 30;
        0.1 10 35;
        3 10 30;
        0.1 10 35 ];
    P=[ 0.3689 0.117 0.2673;
        0.4699 0.4387 0.747;
        0.1091 0.8732 0.5547;
        0.03815 0.5743 0.8828 ];
    c1=0;
    for i=1:4;
        f=0;
        for j=1:3;
            f=f+A(i,j)*(x(j)-P(i,j))^2;
        end
        c1=c1-a(i)*exp(-f);
    end
    fitness=c1;
case 'Hartmann_2',
    a=[1;1.2;3;3.2];
    B=[ 10 3 17 3.5 1.7 8;
        0.05 10 17 0.1 8 14;
        3 3.5 1.7 10 17 8;
        17 8 0.05 10 0.1 14 ];
    Q=[ 0.1312 0.1696 0.5569 0.0124 0.8283 0.5886;
        0.2329 0.4135 0.8307 0.3736 0.1004 0.9991;
        0.2348 0.1451 0.3522 0.2883 0.3047 0.665;
        0.4047 0.8828 0.8732 0.5743 0.1091 0.00381 ];
    c1=0;
    for i=1:4;
        f=0;
        for j=1:6;
            f=f+B(i,j)*(x(j)-Q(i,j))^2;
        end
        c1=c1-a(i)*exp(-f);
    end
    fitness=c1;
case 'Levy',
    f1=(sin(3*pi*x(1)))^2;
```

```
                f2=0;
                for i=2:30;
                    f2=f2+(x(i-1)-1)^2*(1+(sin(3*pi*x(i)))^2);
                end
                f3=(x(30)-1)*(1+(sin(2*pi*x(30)))^2);
                fitness=(f1+f2+f3);
            case 'Matyas',
                f1=0.26*sum(x.^2);
                f2=0.48*prod(x);
                fitness=(f1-f2);
            case 'Perm',
                c1=0;
                for k=1:4
                    f=0;
                    for j=1:4
                        f=f+(j^k+0.5)*(((x(j)/j)^k)-1);
                    end
                       c1=c1+f^2;
                end
                fitness=c1;
            case 'Zakharov',
                f1=sum(x.^2);
                f2=0;
                for i=1:10
                    f2=f2+0.5*i*x(i);
                end
                fitness=(f1+f2^2+f2^4);
            case 'Quartic',
                f1=zeros(1,30);
                   for i=1:30
                       f1(i)=i*x(i).^4;
                   end
                fitness = (sum(f1)+rand());
            case 'Kowalik',
                a=[0.1957;0.1947;0.1735;0.16;0.0844;0.0627;0.0456;0.0342;0.0323;↙
0.0235;0.0246];
                b=[0.25;0.50;1.0;2.0;4.0;6.0;8.0;10.0;12.0;14.0;16.0];
                f=0;
                for i=1:11
                    f=f+a(i)-(x(1)*(b(i)^2+b(i)*x(2))/(b(i)^2+b(i)*x(3)+x(4)));
                end
                fitness=sum(f.^2);
            case 'Tripod',
                px1=((x(1))>=0);
                px2=((x(2))>=0);
                c1=(px2.*(1+px1)+abs(x(1)+50*px2.*(1-2*px1))+abs(x(2)+50*(1-2.↙
*px2)));
                fitness=c1;
            case 'Dejong_4',
                f1=zeros(1,2);
                for i=1:2
                    f1(i)=i*x(i).^4;
                end
                fitness=sum(f1);
            case 'Alpine',
```

```
        c1=sum(abs(x.*sin(x)+0.1*x));
        fitness=c1;
    case 'Pathological',
        c1=0;
        for i=2:5;
            f1=sqrt(100*x(i-1).^2+x(i).^2);
            f2=sin(f1).^2-0.5;
            f3=1+0.001*(x(i-1).^2-2*x(i-1)*x(i)+x(i).^2).^2;
            c1=c1+0.5+(f2/f3);
        end
        fitness=c1;
    case 'Masters',
        c1=0;
        for i=2:5;
            f1=x(i-1).^2+x(i).^2+0.5*x(i-1)*x(i);
            f2=exp(-f1./8);
            f3=cos(4*sqrt(f1));
            c1=c1-f2*f3;
        end
        fitness=c1;
    case '6humpCamelBack',
        c1=4*x(1)^2-2.1*x(1)^4+(1/3)*x(1)^6+x(1)*x(2)-4*x(2)^2+4*x(2)^4;
        fitness=c1;
    case 'Branin',
        a=1; b=5.1/(4*pi^2); c=5/pi; d=6; e=10; f=1/(8*pi);
        c1=a*(x(2)-b*x(1)^2+c*x(1)-d)^2+e*(1-f)*cos(x(1))+e;
        fitness=c1;
    case 'Step',
        c1=0;
        for i=1:30
            f1=floor(x(i)+0.5);
            c1=c1+(f1*f1);
        end
        fitness=c1;
    case 'Weierstrass',
        c1=0;
        D=30;
        for i=1:D
            for k=0:20
                c1=c1+0.5^k*cos(2*pi*3^k*(x(i)+0.5));
            end
        end
        temp=0;
        for k=0:20
            temp=temp+0.5^k*cos(2*pi*3^k*0.5);
        end
        c1=c1-D*temp;
        fitness=-c1;
    case'Michalewicz_10',
        n = 10;
        m = 10;
        s = 0;
        for i = 1:n;
            s = s+sin(x(i))*(sin(i*x(i)^2/pi))^(2*m);
        end
```

```
                    y = -s;
                    fitness=y;
                case'Michalewicz_5',
                    n = 5;
                    m = 10;
                    s = 0;
                    for i = 1:n;
                        s = s+sin(x(i))*(sin(i*x(i)^2/pi))^(2*m);
                    end
                    y = -s;
                    fitness=y;
                case 'Trid',
                    s1 = 0;
                    s2 = 0;
                    for j = 1:6;
                        s1 = s1+(x(j)-1)^2;
                    end
                    for j = 2:6;
                        s2 = s2+x(j)*x(j-1);
                    end
                    y = s1-s2;
                    fitness=y;
                case 'Powell',
                    f1 = 0;
                    for ii = 1:(24/4)
                        term1 = (x(4*ii-3) + 10*x(4*ii-2))^2;
                        term2 = 5 * (x(4*ii-1) - x(4*ii))^2;
                        term3 = (x(4*ii-2) - 2*x(4*ii-1))^4;
                        term4 = 10 * (x(4*ii-3) - x(4*ii))^4;
                        f1 = f1 + term1 + term2 + term3 + term4;
                    end
                    y = f1;
                    fitness=y;
                case'MovedHyper',
                    f=zeros(1,20);
                    for i=1:20
                        f(i)=5*i*x(i).^2;
                    end
                    c1 = sum(f);
                    fitness = c1;
                case'MovedHyper_10',
                    f=zeros(1,10);
                    for i=1:10
                        f(i)=5*i*x(i).^2;
                    end
                    c1 = sum(f);
                    fitness = c1;
                otherwise,
                    disp('fitness function not defined');
            end
        end
    end
end
```

Appendix G

MATLAB code of Combinatorial Bees Algorithm with Bees Nearest Straight Neighbour (BNSN) and/or Bees Routing Optimiser (BRO) for travelling salesman problem

```
clc;
clear;
close all;

%% Problem Definition
[typeOfFunction] = 'KroB200';  % You can change the problem
Instance=Tsplib(typeOfFunction);
Dims=Instance.dim;      % problem dimension
ObjFunction=@(x) Instance.evaluation( x );             % Objective Function
VarSize=[1 Dims];                                  % Decision Variables Matrix Size

%% Bees Algorithm Parameters

n= 10;
nep = 9;
m=5;
e=2;
nsp=4;
ngh=0.1;
stlim=3;
%% BNSN parameter
explore = 1;  % 1 = BNSN "on"; 0 = BNSN "off"
F= randi(3);  % it means there are 3 flowers within the bees vision.
VisionRange = rand(); % the value is between 0 and 1
%% BRO parameter
exploit = 1; % 1 = BRO "on"; 0 = BRO "off"
MinNFN = 1;
MaxNFN = ceil(Dims*ngh);
%% Stop criteria
MaxEval =  800000;
accuracy=0.0001;
ColonySize=(e*nep)+((m-e)*nsp)+(n-m);
MaxIt=round(MaxEval/ColonySize);

%% Initialization
Empty_Patch.Position=[];
Empty_Patch.Cost=[];
Empty_Patch.Stagnated =[];
Empty_Patch.Counter=[];
Patch=repmat(Empty_Patch,n,1);
counter=0;

% Create Initial Solutions
for i=1:n
    if explore == 1
        Patch(i).Position= BNSN_0(randi(Dims),F,VisionRange,Instance);
    else
        Patch(i).Position= randperm(Dims);
    end
    Patch(i).Cost=ObjFunction(Patch(i).Position);
    Patch(i).Stagnated = 0;
    counter = counter + 1;
    Patch(i).Counter = counter;
end
```

```
% Sort
[~, SortOrder]=sort([Patch.Cost]);
Patch=Patch(SortOrder);
% Update Best Solution Ever Found
BestSol=Patch(1);
% Array to Hold Best Cost Values
BestCost=zeros(MaxIt,1);
Counter=zeros(MaxIt,1);
OptSol.Cost=inf;

%% Bees Algorithm Main Loop
for it=1:MaxIt
    if counter >= MaxEval
        break;
    end

    for i=1:e  % ELITE SITES
        BestForager.Cost=inf;
        for j=1:nep
            ForagerBees.Position= Foraging(Patch(i).Position);
            ForagerBees.Cost=ObjFunction(ForagerBees.Position);
            ForagerBees.Stagnated = Patch(i).Stagnated;
            counter = counter + 1;
            ForagerBees.Counter = counter;
            if ForagerBees.Cost<BestForager.Cost
                BestForager=ForagerBees;
            end
        end
        if BestForager.Cost<Patch(i).Cost
            Patch(i)=BestForager;
            Patch(i).Stagnated=0;
        else
            Patch(i).Stagnated=Patch(i).Stagnated+1;
        end
        % ABANDONMENT
        if(Patch(i).Stagnated>stlim)
            Patch(i).Stagnated=0;
            if exploit == 1
                [Patch(i).Position,Patch(i).Cost] = BRO_0(Patch(i).Position, Instance,↙
MinNFN, MaxNFN);
                counter = counter + 1;
            end
            Patch(i).Counter = counter;
        end
    end

    for i=e+1:m  % NON-ELITE SELECTED SITES
        BestForager.Cost=inf;
        for j=1:nsp
            ForagerBees.Position= Foraging(Patch(i).Position);
            ForagerBees.Cost=ObjFunction(ForagerBees.Position);
            ForagerBees.Stagnated = Patch(i).Stagnated;
            counter = counter + 1;
            ForagerBees.Counter = counter;
            if ForagerBees.Cost<BestForager.Cost
```

```
                BestForager=ForagerBees;
            end
        end
        if BestForager.Cost<Patch(i).Cost
            Patch(i)=BestForager;
            Patch(i).Stagnated=0;
        else
            Patch(i).Stagnated=Patch(i).Stagnated+1;
        end
        % ABANDONMENT
        if(Patch(i).Stagnated>stlim)
            Patch(i).Stagnated=0;
            if exploit == 1
                [Patch(i).Position,Patch(i).Cost] = BRO_0(Patch(i).Position, Instance,↙
2*MinNFN, 2*MaxNFN);
                counter = counter + 1;
            end
            Patch(i).Counter = counter;
        end
    end

    for i=m+1:n  % REMAINING (NON-SELECTED) SITES
        if exploit == 1
            Patch(i).Position= BNSN_0(randi(Dims),F,VisionRange,Instance);
            [Patch(i).Position,Patch(i).Cost] = BRO_1(Patch(i).Position, Instance,↙
randi(ceil(0.25*Dims)));
        else
            Patch(i).Position= randperm(Dims);
        end
        Patch(i).Cost=ObjFunction(Patch(i).Position);
        Patch(i).Stagnated = 0;
        counter = counter + 1;
        Patch(i).Counter = counter;
    end

    % SORTING
    [~, SortOrder]=sort([Patch.Cost]);
    Patch=Patch(SortOrder);
    % UPDATE SOLUTION
    BestSol=Patch(1);

    if BestSol.Cost < OptSol.Cost
        OptSol=BestSol;
    end

    % Store Best Cost Ever Found
    BestCost(it)=OptSol.Cost;
    Counter(it)=OptSol.Counter;
    % Display Iteration Information
    disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(BestCost(it)) ': Best Cost↙
= ' num2str(Counter(it))]);

    figure(1);
    PlotSolution(BestSol.Position,Instance);
    pause(0.01);
```

```
    if(abs((OptSol.Cost-Instance.optima)/Instance.optima) <= accuracy)
        break;
    end
end

%% Results
figure(1);
PlotSolution(OptSol.Position,Instance);

figure;
semilogy(BestCost,'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
```

```
function p = BNSN_0 (s,F,Dr,model)
x=model.x;
y=model.y;
D=model.D;
n = size(D,1);
p = zeros(1,n,'uint16');
p(1) = s;
D(s,:) = inf;

for k = 2:2
    D(s,:) = inf;
    [junk,s] = min(D(:,s));
    p(k) = s;

end

for k = 3:n
    D(s,:) = inf;
    Temp=D(:,s);
    R=cell(F,1);
    PQ_PR=cell(F,1);
    PRd=cell(F,1);
    angd=cell(F,1);
    valZ=zeros(F,1);
    val = zeros(F,1);
    idx = zeros(F,1);
    for i=1:F
      [val(i),idx(i)] = min(Temp);
      % remove for the next iteration the last smallest value:
      Temp(idx(i)) = inf;
    end
    if ((val(end)-val(1))/val(1)) <= Dr
        Q=[x(p(k-2)) y(p(k-2))];
        P=[x(p(k-1)) y(p(k-1))];
        PQd= norm(P-Q);
        for a= 1:F
            R{a}=[x(idx(a)) y(idx(a))];
            PQ_PR{a}=sum((Q-P).*(R{a}-P));
            PRd{a}= norm(P-R{a});
            angd{a}=acosd(PQ_PR{a}/(PQd*PRd{a}));
            valZ(a)=(1+(2-(angd{a}/90)))*PRd{a};
        end

        [~,idz] = min(valZ);
        s=idx(idz);

        p(k) = s;
    else
        [junk,s] = min(D(:,s));
        p(k) = s;
    end
end
```

```
function NewPatch = Foraging(sol)

    m=randi([1 3]);

    switch m
        case 1
            NewPatch = Swap2(sol);
        case 2
            NewPatch = Reverse2(sol);
        case 3
            NewPatch = Insert2(sol);
    end
end

function NewPatch = Swap2(sol)
    n=numel(sol);
    i=randsample(n,2);
    i1=i(1);
    i2=i(2);
    NewPatch=sol;
    NewPatch([i1 i2])=sol([i2 i1]);
end

function NewPatch = Reverse2(sol)
    n=numel(sol);
    i=randsample(n,2);
    i1=min(i(1),i(2));
    i2=max(i(1),i(2));
    NewPatch = sol;
    NewPatch(i1:i2)=sol(i2:-1:i1);
end

function NewPatch = Insert2(sol)
    n=numel(sol);
    i=randsample(n,2);
    i1=i(1);
    i2=i(2);
    if i1<i2
        NewPatch = [sol(1:i1-1) sol(i1+1:i2) sol(i1) sol(i2+1:end)];
    else
        NewPatch = [sol(1:i2) sol(i1) sol(i2+1:i1-1) sol(i1+1:end)];
    end
end
```

```
function [ p,L ] = BRO_0( p, model, BatasB, BatasA)
x=model.x;
y=model.y;
D=model.D;
n=model.dim;

Lmin = inf;
idrem = randperm(n,[randi([BatasB BatasA])]);

rem = p(idrem);
p(idrem)=[];

p = TwoOpt_0(p,D);

[p,L] = Insert_Forgotten( rem,p,D,x,y);

% Keep best tour
if L <= Lmin
    Lmin = L;
    pmin = p;
end

% Output
p = double(pmin);
L = Lmin;
end
```

```
function [ p,L ] = BRO_1( p, model , BatasA) %, BatasB, BatasA)
x=model.x;
y=model.y;
X(:,1)=transpose(x);
X(:,2)=transpose(y);
D=model.D;
n=model.dim;

Lmin = inf;
fgn= randi(n);
xx(:,1)= x(fgn);
xx(:,2)= y(fgn);

Dis=pdist2 (xx,X);
[~,idx]=sort(Dis);
rem=idx(1:BatasA);

for i=1:BatasA
   p(p==rem(i))=[];
end

p = TwoOpt_0(p,D);

[p,L] = Insert_Forgotten( rem,p,D,x,y);

% Keep best tour
if L <= Lmin
    Lmin = L;
    pmin = p;
end

% Output
p = double(pmin);
L = Lmin;
end
```

```
function Z = TwoOpt_0(Z,MatrixDistance)
n = numel(Z);
Zmin = -1;

while Zmin < 0
    Zmin = 0;
    i = 0;
    b = Z(n);

    while i < n-2
        a = b;
        i = i+1;
        b = Z(i);
        Distance_a_b = MatrixDistance(a,b);
        j = i+1;
        d = Z(j);
        while j < n
            c = d;
            j = j+1;
            d = Z(j);

            z = (MatrixDistance(a,c) - MatrixDistance(c,d)) + MatrixDistance(b,d) -↙
Distance_a_b;

            if z < Zmin
                Zmin = z;
                imin = i;
                jmin = j;
            end
        end
    end

    if Zmin < 0
        Z(imin:jmin-1) = Z(jmin-1:-1:imin);
    end
end
end
```

```
function [p,L] = Insert_Forgotten( rem,init,D,x,y )
p=init;
nrem = numel (rem);
for i = 1 : nrem
    np = numel (p);
    Dis = zeros(np,1);
    center.x(1) = (x(p(1))+x(p(end)))*0.5;
    center.y(1) = (y(p(1))+y(p(end)))*0.5;
    Dis(1) = (sqrt((x(rem(i))-center.x(1))^2+(y(rem(i))-center.y(1))^2));
    for j=2:np
        center.x(j) = (x(p(j))+x(p(j-1)))*0.5;
        center.y(j) = (y(p(j))+y(p(j-1)))*0.5;
        Dis(j) = (sqrt((x(rem(i))-center.x(j))^2+(y(rem(i))-center.y(j))^2));
    end
   [a,b] = min (Dis);
    s = p(b);
    idx=find(p==s);
    if idx==1
        p = [rem(i) p];
    else
        p = [p(1:idx-1) rem(i) p(idx:end)];
    end
end
L=TourLength_0(p,D);
end
```

```
function PlotSolution(tour,model)
    tour=[tour tour(1)];
    plot(model.x(tour),model.y(tour),'k-o',...
        'MarkerSize',2.5,...
        'MarkerFaceColor','y',...
        'LineWidth',.5);

    xlabel('x');
    ylabel('y');

    axis equal;
    grid off;

    alpha = 0.1;

    xmin = min(model.x);
    xmax = max(model.x);
    dx = xmax - xmin;
    xmin = floor((xmin - alpha*dx)/10)*10;
    xmax = ceil((xmax + alpha*dx)/10)*10;
    xlim([xmin xmax]);
    ymin = min(model.y);
    ymax = max(model.y);
    dy = ymax - ymin;
    ymin = floor((ymin - alpha*dy)/10)*10;
    ymax = ceil((ymax + alpha*dy)/10)*10;
    ylim([ymin ymax]);
end
```

```
classdef Tsplib

    properties
        type;                   % type of fitness function
        x;                      % minimum x and y coordinates
        y;                      % maximum x and y coordinates
        dim;                    % number of parameters of function
        D;                      % matrix Distance
        optima;                 % global maximum
        definedFunctions        % functions defined in this class
    end

    methods
        function obj = Tsplib(typeOfFunction)
            obj.definedFunctions=↙
{'A280','Att532','Berlin52','Ch150','Eil51','Eil76','D198','D1291','Fl1577','KroA100',↙
'KroA150','KroA200','KroB100','KroB150','KroB200','KroC100','KroD100','KroE100','Lin31↙
8','Nrw1379','Pcb442','Pcb1173','Pr76','Pr144','Pr152','Pr226','Pr299','Rat99','Rat195↙
','Rat783','Rl1304','Rl1323','St70','Ts225'};
            obj.type=typeOfFunction;
            switch(obj.type)
                case 'A280',
                    obj.dim=280;
                    obj.optima=2579;
                    x=↙
[288,288,270,256,256,246,236,228,228,220,212,204,196,188,196,188,172,164,156,148,140,1↙
48,164,172,156,140,132,124,116,104,104,104,90,80,64,64,56,56,56,56,56,56,56,40,40,40,4↙
0,40,40,40,32,32,32,32,32,32,32,32,32,40,56,56,48,40,32,32,24,16,16,8,8,8,8,8,8,8,16,8,8,↙
24,32,32,32,32,32,32,40,40,40,40,44,44,44,32,24,16,16,24,32,44,56,56,56,56,56,64,72,72↙
,56,48,56,56,48,48,56,56,48,56,56,104,104,104,104,104,104,104,116,124,132,132,140,148,↙
156,164,172,172,172,172,172,172,180,180,180,180,180,172,172,172,172,164,148,124,124,12↙
4,124,124,124,104,104,104,104,104,104,104,104,104,92,80,72,64,72,80,80,80,88,104,124,1↙
24,132,140,132,124,124,124,124,124,132,124,120,128,136,148,162,156,172,180,180,172,172↙
,172,180,180,188,196,204,212,220,228,228,236,236,236,228,228,236,236,228,228,236,236,2↙
28,228,236,252,260,260,260,260,260,260,260,276,276,276,276,284,284,284,284,284,284,284↙
,288,280,276,276,276,268,260,252,260,260,236,228,228,236,236,228,228,228,228,220,212,2↙
04,196,188,180,180,180,180,180,196,204,212,220,228,236,246,252,260,280];
                    y=↙
[149,129,133,141,157,157,169,169,161,169,169,169,169,169,161,145,145,145,145,145,145,1↙
69,169,169,169,169,169,169,169,161,153,161,169,165,157,157,165,169,161,153,145,137,129,121↙
,121,129,137,145,153,161,169,169,161,153,145,137,129,121,113,113,113,105,99,99,97,89,8↙
9,97,109,109,97,89,81,73,65,57,57,49,41,45,41,49,57,65,73,81,83,73,63,51,43,35,27,25,2↙
5,25,17,17,17,11,9,17,25,33,41,41,41,49,49,51,57,65,63,73,73,81,83,89,97,97,105,113,12↙
1,129,137,145,145,145,145,137,137,137,137,137,125,117,109,101,93,85,85,77,69,61,53,53,↙
61,69,77,81,85,85,93,109,125,117,101,89,81,73,65,49,41,33,25,17,9,9,9,21,25,25,25,41,4↙
9,57,69,77,81,65,61,61,53,45,37,29,21,21,9,9,9,9,9,25,21,21,29,29,37,45,45,37,41,49,57↙
,65,73,69,77,77,69,61,61,53,53,45,45,37,37,29,29,21,21,21,29,37,45,53,61,69,77,77,69,6↙
1,53,53,61,69,77,85,93,101,109,109,101,93,85,97,109,101,93,85,85,85,93,93,101,101,109,↙
117,125,125,117,109,101,93,93,101,109,117,125,145,145,145,145,145,145,141,125,129,133]↙
;
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Att532',
```

```
                    obj.dim=532;
                    obj.optima=27686;
                    x=↙
[7810,7798,7264,7324,7547,7744,7821,7883,7874,7927,7848,7802,7962,7913,7724,7503,7759,↙
7890,7254,7790,7142,7606,7772,7744,7846,7622,6937,7576,7783,7716,7295,7777,7700,7726,7↙
702,7583,7654,7417,7267,6806,5259,7698,7570,7617,7752,7673,7692,7547,7259,5387,7679,76↙
74,7631,7520,7848,5685,7832,6735,7647,7338,4602,4606,7399,7037,7458,7364,6058,6868,383↙
2,6670,7443,7160,6139,7333,6237,5385,6911,6304,7111,6740,7698,7613,7360,6779,7207,6241↙
,7432,4354,6589,7817,6051,5356,7554,7534,4217,7349,7128,3950,6947,7549,5168,6524,5871,↙
7542,6660,7216,6607,7601,6123,6450,6713,7355,7604,7541,7506,4871,2906,6488,6312,6008,4↙
427,4679,5955,6891,7705,7562,4634,4607,6557,7344,5543,7124,7466,6259,6366,5597,4655,78↙
05,3396,6603,6537,4342,7037,7345,7271,5336,5964,7660,7872,6567,6602,4806,7909,5926,744↙
9,6333,3108,7844,5427,6862,6621,6150,7388,7351,4694,6340,6425,6577,6864,5706,4496,4574↙
,3824,5803,5720,6454,6120,7988,6376,7841,5778,5457,5671,4293,7423,7342,5541,5621,7750,↙
6327,7879,199,6652,5678,5207,7429,7262,6427,1851,6207,6069,4780,7603,5751,6365,6958,63↙
17,5417,6426,7922,7331,5965,4965,6833,6798,7667,1047,7803,7370,952,7906,250,5111,6453,↙
7492,6140,5315,5316,4232,7408,8013,5160,7141,5887,4694,7633,7919,1784,1482,236,6713,76↙
96,536,317,5649,6235,7199,5540,5400,5796,2342,7494,7321,6265,8001,226,6148,5987,7582,7↙
422,6623,7475,7654,7838,6570,4364,7316,4857,7533,5719,7452,7747,5841,3229,7076,7657,63↙
60,525,5619,7989,5697,6050,7082,5539,741,6731,7453,7695,7299,863,7861,5960,4252,6402,5↙
342,6656,7532,7434,5679,6518,4537,806,6113,7440,6204,7715,7503,5821,7131,7909,920,6468↙
,5677,218,6881,5650,197,5531,6387,4458,6190,7055,7238,5930,7543,5291,4196,6617,4831,28↙
35,174,5350,7346,6044,4898,3307,1918,7125,6422,5881,141,7851,4929,5963,5470,7458,1263,↙
6766,4763,3461,7309,6848,178,1882,4584,3174,7049,7753,6597,4476,1575,7304,10,6800,5296↙
,7104,6547,7267,3189,5117,4973,4488,7351,6007,4612,7015,3233,240,6686,6307,7448,7087,2↙
067,5260,4174,36,7856,7315,3319,2126,7418,6885,4959,4996,5681,5277,7643,3390,8080,6139↙
,2694,7152,7822,7416,7352,354,6493,7905,8229,6803,4012,4759,8101,7989,8063,8080,7004,6↙
252,6826,7218,464,809,7240,7046,8098,7314,7035,5506,8184,6932,5914,2908,6496,8525,6765↙
,7985,6854,7926,7973,5060,4056,5637,2011,8038,6651,552,6621,8594,4719,5472,8605,345,82↙
28,5005,5114,5964,602,5098,5068,8292,6258,5010,6494,437,413,659,5840,6378,6379,6359,32↙
45,450,478,5571,489,513,6136,4170,1721,893,5930,4619,4125,5139,572,4500,2372,993,527,5↙
788,3719,4805,5140,5344,5532,5069,1595,5666,2260,4244,5596,4569,1072,3499,5136,783,834↙
,1406,3390,2384,982,1422,1361,1926,1213,1415,1082,1254,5070,1212,1249,3477,1322,1253,1↙
276,2647,1443,1961,1790,1503,5393,5469];
                    y=↙
[6053,5709,5575,5560,5503,5476,5457,5408,5405,5365,5358,5317,5287,5280,5210,5191,5143,↙
5130,5129,5038,5032,5009,4989,4933,4923,4917,4917,4915,4912,4909,4887,4869,4854,4833,4↙
815,4813,4795,4788,4779,4755,4751,4745,4741,4724,4721,4718,4666,4664,4630,4623,4581,45↙
79,4573,4572,4546,4546,4542,4509,4504,4481,4478,4468,4467,4446,4428,4427,4426,4418,441↙
0,4401,4375,4370,4369,4335,4332,4318,4296,4294,4288,4282,4279,4275,4275,4273,4270,4268↙
,4265,4262,4256,4252,4246,4241,4236,4227,4224,4219,4215,4215,4209,4208,4208,4207,4202,↙
4198,4193,4180,4173,4171,4167,4160,4154,4151,4146,4141,4138,4132,4131,4128,4126,4117,4↙
109,4084,4081,4075,4065,4058,4054,4049,4047,4046,4042,4039,4037,4030,4002,3993,3992,39↙
91,3990,3982,3982,3966,3965,3951,3948,3943,3935,3924,3922,3922,3920,3914,3912,3912,391↙
1,3909,3908,3902,3894,3892,3891,3888,3879,3877,3877,3870,3867,3858,3854,3844,3844,3843↙
,3838,3824,3823,3821,3821,3820,3819,3818,3813,3808,3807,3788,3776,3775,3769,3768,3760,↙
3745,3743,3743,3742,3742,3742,3737,3725,3717,3710,3700,3695,3694,3690,3681,3679,3678,3↙
673,3673,3656,3655,3634,3624,3622,3618,3610,3608,3602,3598,3588,3583,3580,3578,3569,35↙
67,3560,3558,3557,3554,3551,3534,3523,3517,3514,3508,3502,3499,3496,3494,3494,3494,348↙
8,3486,3481,3476,3472,3471,3469,3468,3461,3459,3439,3430,3429,3426,3418,3415,3413,3402↙
,3396,3390,3389,3388,3377,3375,3371,3362,3360,3359,3358,3352,3339,3329,3328,3312,3302,↙
3301,3301,3297,3291,3271,3269,3242,3235,3235,3235,3234,3229,3220,3219,3219,3216,3207,3↙
206,3190,3188,3181,3175,3173,3171,3165,3143,3123,3101,3100,3099,3086,3086,3086,3081,30↙
80,3065,3050,3049,3031,3029,3023,3021,3011,3008,3007,2985,2981,2957,2948,2929,2929,292↙
9,2928,2917,2912,2901,2867,2858,2848,2840,2833,2832,2823,2820,2817,2814,2809,2803,2789↙
```

```
,↙
2774,2741,2734,2732,2720,2718,2717,2712,2702,2684,2643,2627,2570,2564,2563,2555,2555,2↙
550,2537,2532,2520,2510,2506,2466,2411,2409,2406,2378,2376,2359,2341,2333,2329,2327,23↙
12,2295,2291,2274,2254,2230,2190,2185,2181,2181,2151,2150,2139,2138,2123,2115,2109,207↙
8,2048,2043,2039,2032,2026,2,1992,1953,1952,1950,1931,1921,1905,1886,1886,1883,1876,18↙
76,1860,1835,1805,1795,1774,1773,1773,1766,1762,1757,1746,1739,1733,1719,1685,1683,168↙
2,1681,1678,1664,1663,1657,1640,1627,1606,1577,1564,1558,1558,1535,1534,1526,1513,1510↙
,1504,1482,1479,1476,1471,1458,1430,1421,1395,1394,1390,1383,1354,1351,1347,1344,1338,↙
1331,1325,1314,1302,1298,1281,1274,1256,1255,1254,1247,1243,1232,1165,1161,1151,1132,1↙
125,1124,1108,1093,1084,1084,1077,1053,1043,1033,1018,1003,998,998,942,914,913,896,892↙
,886,883,863,825,825,757,750,698,695,659,658,637,636,633,628,625,617,605,603,600,599,5↙
80,580,559,485,459,445,429,362,355,10];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Berlin52',
                    obj.dim=52;
                    obj.optima=7542;
                    x=↙
[565,25,345,945,845,880,25,525,580,650,1605,1220,1465,1530,845,725,145,415,510,560,300↙
,520,480,835,975,1215,1320,1250,660,410,420,575,1150,700,685,685,770,795,720,760,475,9↙
5,875,700,555,830,1170,830,605,595,1340,1740];
                    y=↙
[575,185,750,685,655,660,230,1,1175,1130,620,580,200,5,680,370,665,635,875,365,465,585↙
,415,625,580,245,315,400,180,250,555,665,1160,580,595,610,610,645,635,650,960,260,920,↙
500,815,485,65,610,625,360,725,245];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Ch150',
                    obj.dim=150;
                    obj.optima=6528;
                    x=↙
[37.43935167,612.1759509,38.13123382,53.44180811,143.0606355,689.9451267,112.7478816,1↙
41.4875865,661.0513902,98.78990366,697.3881697,536.489419,192.4067321,282.7865259,240.↙
8251726,246.9281323,649.7313216,352.9658563,633.3923677,488.3117994,141.4039287,17.363↙
26126,397.5586451,565.7853781,475.8975387,322.4224567,397.5586634,672.8618339,571.2189↙
68,104.6531166,356.7098389,400.4070256,282.3036243,58.77669884,189.7506224,659.912412,↙
639.0307636,415.0258357,547.2662016,616.6547903,494.8592427,629.9980812,471.1014312,13↙
8.2440514,91.58475567,390.6972812,565.1617825,54.52489804,334.350833,531.0291025,475.7↙
345906,228.8325219,578.3805348,358.9170574,486.4648555,343.1693708,530.3626972,498.806↙
5475,224.3182716,595.8360733,661.5588724,43.68920455,79.46534525,210.4163247,432.26422↙
92,623.2487161,436.519474,59.41632655,630.9230074,579.326554,117.8624507,297.7912566,2↙
2.76427037,259.709581,342.3579874,10.02609501,315.2926064,220.7044919,192.118606,271.5↙
042719,530.7320005,42.53314417,396.1274793,118.6631474,395.6913877,559.0157106,22.6471↙
0359,135.6377085,141.4507014,396.7741299,87.74945628,350.424564,216.7010817,130.923773↙
7,72.63298567,144.600295,212.3725077,49.91867865,656.6943526,176.5941624,500.382520,63↙
4.3178678,59.75373727,15.21457651,283.0054379,146.5389001,101.8685605,588.1968537,457.↙
2628633,537.466368,269.3669099,239.9045384,88.46775004,658.9133693,97.73591463,506.619↙
1384,500.2566898,594.4048565,66.23081466,598.4162994,172.3088331,299.4812852,303.83798↙
95,197.896927,56.01995677,255.5566183,608.4256112,70.27227033,398.229900,635.4970237,3↙
78.3484559,484.8029663,278.8710883,381.6537301,557.6070708,249.0589749,562.9048788,398↙
.5504366,590.8939721,558.2008004,461.4114714,354.7242882,193.6611296,352.3140807,308.4↙
```

```
34571,299.588137,334.2748764,690.9658586,48.07981241,91.64676477];
                    y=↙
[541.2090699,494.3166877,353.1484582,131.4849014,631.7200954,468.5354999,529.4177578,5↙
04.8184856,445.9375182,384.5926031,180.3962284,287.2279085,20.43940593,229.8001556,281↙
.5141437,322.4613321,62.33315753,666.7873102,534.9398454,437.4869440,228.4325551,240.2↙
407069,231.3591209,282.3858749,468.5392706,550.3165478,74.75883878,432.8826410,530.261↙
6992,482.8224769,67.64771317,253.6794480,426.8380501,507.1712387,460.3815234,226.62841↙
56,467.2302301,233.3045376,161.6589278,339.3409309,148.1217856,433.4548164,314.2219308↙
,137.1679920,110.0203008,423.9774318,429.1598153,438.5515408,153.7969238,612.3874828,3↙
85.7844619,410.4461940,321.3303495,404.4670353,593.0429937,509.3123571,137.6881276,576↙
.2102675,312.4677490,81.81300514,217.0456944,305.4722789,445.9641738,130.7151137,629.4↙
092661,69.18928508,282.9356456,40.12802344,230.3429888,601.0359411,112.9796834,166.313↙
1887,455.5340094,10.61999259,599.3880183,488.9310558,273.2275476,270.0819746,314.18399↙
23,225.2921990,504.0670155,656.3645163,539.4648066,508.7129104,699.5376048,560.8866941↙
,526.2470393,325.8409902,485.2477928,460.7557115,19.61701291,420.6531187,466.4816411,3↙
51.1491733,645.7852219,457.4224284,594.9216893,541.4350825,558.1109594,648.5239953,198↙
.7428378,612.8291643,551.6321887,143.0441929,376.4439530,39.42317943,635.0986850,580.5↙
946977,350.0164047,472.5842277,367.4763637,102.6297653,384.0507209,583.9575181,157.455↙
8658,233.0022156,64.91363935,275.8741869,24.13173876,414.5557574,344.3963466,251.82951↙
21,21.05260638,512.3888961,243.0663818,448.8651882,222.5421309,77.92270264,119.5576574↙
,133.3225903,272.2907677,677.0730379,299.9308771,360.3337603,595.3185092,76.65951126,6↙
70.0382113,392.6493259,370.7414914,0.4198814510,530.5254969,685.4045362,669.7432521,14↙
0.3273324,115.2054270,530.5889619,152.1494569,134.5793307,270.9680674,166.3541158];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'D198',
                    obj.dim=198;
                    obj.optima=15781;
                    x=↙
[0,551.200,627.400,703.600,703.600,627.400,551.200,881.400,932.200,957.600,983,1008.40↙
,1033.80,1313.20,1287.80,1287.80,1313.20,1465.60,1516.40,1592.60,1592.60,1516.40,1465.↙
60,1567.20,1592.60,1567.20,1541.80,1491,1440.20,1465.60,1414.80,1440.20,1491,1516.40,1↙
592.60,1465.60,1440.20,1389.40,1541.80,1160.80,1160.80,1262.40,1287.80,1338.60,1414.80↙
,1491,1541.80,1643.40,1668.80,1618,1567.20,1516.40,1465.60,1414.80,1338.60,1313.20,123↙
7,1237,1313.20,1338.60,1414.80,1465.60,1694.20,1618,1516.40,1414.80,1338.60,1313.20,12↙
37,1357.70,1237,1313.20,1338.60,1414.80,1465.60,1567.20,1592.60,1618,1694.20,1668.80,1↙
541.80,1440.20,1414.80,1338.60,1262.40,1237,1338.60,1414.80,1465.60,1491,1668.80,1694.↙
20,1618,1592.60,1567.20,1516.40,1440.20,1414.80,1338.60,1262.40,1592.60,1618,1668.80,1↙
694.20,1745,1821.20,1846.60,1948.20,1922.80,1897.40,1872,1821.20,1795.80,1770.40,1770.↙
40,1795.80,1821.20,1872,1897.40,1922.80,1808.50,1757.70,1884.70,1999,2075.20,2113.30,2↙
176.80,2236.50,2176.80,2126,2100.60,2100.60,2126,2100.60,2126,2151.40,2236.50,1999,188↙
4.70,2100.60,2126,2100.60,2126,2176.80,2151.40,2126,2100.60,2100.60,2126,2176.80,2227.↙
60,2126,2100.60,1795.80,1821.20,1846.60,1872,1897.40,1948.20,2056.20,2100.60,2126,2253↙
,2303.80,2380,2405.40,2024.40,2151.40,2075.20,2176.80,2350.80,2350.80,3652.10,3725.70,↙
3725.70,3652.10,3726.20,3802.40,3853.20,3802.40,3700.80,3605.60,3700.80,3802.40,3853.2↙
0,4028.30,3952.10,3728.30,3652.10,3652.10,3728.30,3952.10,4028.30,3853.20,3952.10,4028↙
.30,4028.30,3952.10000000000];
                    y=↙
[0,996.400,996.400,996.400,1047.20,1047.20,1047.20,1352,1352,1352,1352,1352,1352,1123.↙
40,1098,996.400,996.400,996.400,996.400,996.400,1098,1098,1098,1123.40,1148.80,1174.20↙
,1174.20,1174.20,1174.20,1199.60,1225,1225,1225,1225,1250.40,1250.40,1250.40,1250.40,1↙
275.80,1123.40,1225,1301.20,1301.20,1301.20,1301.20,1301.20,1301.20,1301.20,1326.60,13↙
26.60,1326.60,1326.60,1326.60,1326.60,1326.60,1326.60,1326.60,1352,1352,1352,1352,1352↙
```

```
,↙
1352,1377.40,1377.40,1377.40,1377.40,1377.40,1377.40,1390.10,1402.80,1402.80,1402.80,1↙
402.80,1402.80,1402.80,1402.80,1402.80,1428.20,1428.20,1428.20,1428.20,1428.20,1428.20↙
,1428.20,1453.60,1453.60,1453.60,1453.60,1453.60,1453.60,1453.60,1479,1479,1479,1479,1↙
479,1479,1479,1479,1504.40,1504.40,1529.80,1529.80,1529.80,1529.80,1529.80,1529.80,155↙
5.20,1555.20,1555.20,1555.20,1555.20,1555.20,1656.80,1656.80,1656.80,1656.80,1656.80,1↙
656.80,1694.90,1694.90,1733,1733,1733,1733,1733,1733,1783.80,1783.80,1783.80,1809.20,1↙
809.20,1834.60,1834.60,1834.60,1847.30,1847.30,1847.30,1860,1860,1885.40,1885.40,1885.↙
40,1910.80,1910.80,1910.80,1936.20,1936.20,1936.20,1936.20,1961.60,1961.60,1987,1987,1↙
987,1987,1987,1987,1987,1987,1987,1987,1987,1987,1987,1402.80,1402.80,1707.60,1707.60,↙
1733,1847.30,1010.30,1010.30,1086.50,1086.50,1148.80,1148.80,1148.80,1174.20,1174.20,1↙
199.60,1199.60,1199.60,1199.60,1310.30,1310.30,1310.30,1310.30,1386.50,1386.50,1386.50↙
,1386.50,1123.40,1086.50,1086.50,1010.30,1010.30000000000];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'D1291',
                    obj.dim=1291;
                    obj.optima=50801;
                    x=↙
[0,837,862.4,887.8,913.2,938.6,964,989.4,1014.8,1040.2,1065.6,1091,1116.4,1141.8,1167.↙
2,1167.2,1141.8,1116.4,1091,1065.6,1040.2,1014.8,989.4,964,938.6,913.2,887.8,862.4,837↙
,837,862.4,887.8,913.2,938.6,964,989.4,1014.8,1040.2,1065.6,1091,1116.4,1141.8,1167.2,↙
1167.2,1141.8,1116.4,1091,1065.6,1040.2,1014.8,989.4,964,938.6,913.2,887.8,862.4,837,8↙
37,862.4,887.8,913.2,938.6,964,989.4,1014.8,1040.2,1065.6,1091,1116.4,1141.8,1167.2,11↙
67.2,1141.8,1116.4,1091,1065.6,1040.2,1014.8,989.4,964,938.6,913.2,887.8,862.4,837,837↙
,862.4,887.8,913.2,938.6,964,989.4,1014.8,1040.2,1065.6,1091,1116.4,1141.8,1167.2,1167↙
.2,1141.8,1116.4,1091,1065.6,1040.2,1014.8,989.4,964,938.6,913.2,887.8,862.4,837,837,8↙
62.4,887.8,913.2,938.6,964,989.4,1014.8,1040.2,1065.6,1091,1116.4,1141.8,1167.2,1167.2↙
,1141.8,1116.4,1091,1065.6,1040.2,1014.8,989.4,964,938.6,913.2,887.8,862.4,837,837,862↙
.4,887.8,913.2,938.6,964,989.4,1014.8,1040.2,1065.6,1091,1116.4,1141.8,1167.2,1167.2,1↙
141.8,1116.4,1091,1065.6,1040.2,1014.8,989.4,964,938.6,913.2,887.8,862.4,837,837,862.4↙
,887.8,913.2,938.6,964,989.4,1014.8,1040.2,1065.6,1091,1116.4,1141.8,1167.2,1167.2,114↙
1.8,1116.4,1091,1065.6,1040.2,1014.8,989.4,964,938.6,913.2,887.8,862.4,837,837,862.4,8↙
87.8,913.2,938.6,964,989.4,1014.80,1040.20,1065.60,1091,1116.40,1141.80,1167.20,1243.4↙
0,1167.20,1141.80,1116.40,1091,1065.60,1040.20,1014.80,989.400,964,938.600,913.200,887↙
.800,862.400,837,964,1040.20,1141.80,1218,1218,1141.80,1040.20,964,913.200,964,1040.20↙
,1141.80,1218,1218,1141.80,1040.20,964,913.200,964,1040.20,1141.80,1218,1218,1141.80,1↙
040.20,964,964,1040.20,1141.80,1218,1218,1141.80,1040.20,964,1298.70,1323.70,1348.70,1↙
373.70,1398.70,1423.70,1448.70,1473.70,1498.70,1523.70,1548.70,1573.70,1598.70,1623.70↙
,1648.70,1673.70,1698.70,1723.70,1748.70,1773.70,1798.70,1823.70,1848.70,1873.70,1898.↙
70,1923.70,1948.70,1973.70,1998.70,2023.70,2048.70,2073.70,2098.70,2123.70,2148.70,217↙
3.70,2198.70,2248.70,1298.70,1348.70,1398.70,1448.70,1498.70,1548.70,1598.70,1648.70,1↙
698.70,1748.70,1798.70,1848.70,1898.70,1948.70,1998.70,2048.70,2098.70,2148.70,2198.70↙
,2248.70,2298.70,2348.70,2398.70,2448.70,2498.70,2548.70,2598.70,2648.70,2698.70,2748.↙
70,2798.70,2848.70,2898.70,2948.70,2998.70,3048.70,3098.70,3148.70,3198.70,3248.70,329↙
8.70,3348.70,3398.70,3448.70,3423.70,3373.70,3323.70,3273.70,3223.70,3173.70,3123.70,3↙
073.70,3023.70,2973.70,2923.70,2873.70,2823.70,2773.70,2723.70,2673.70,2623.70,2573.70↙
,2448.70,2398.70,2348.70,2298.70,2173.70,2123.70,2073.70,2023.70,1973.70,1923.70,1873.↙
70,1823.70,1773.70,1723.70,1673.70,1623.70,1573.70,1523.70,1473.70,1423.70,1373.70,132↙
3.70,2298.70,2348.70,2398.70,2448.70,2678.50,2202.20,1522.80,1395.80,1472,1573.60,1649↙
.80,2107,2183.20,2284.80,2361,2361,2284.80,2183.20,2107,2005.40,1929.20,1827.60,1751.4↙
0,1649.80,1573.60,1522.80,1472,1395.80,1395.80,1472,1573.60,1649.80,1751.40,1827.60,19↙
29.20,2005.40,2107,2183.20,2284.80,2361,2869,2818.20,2767.40,2716.60,2361,2284.80,2183↙
```

.↙
20,2107,2005.40,1929.20,1827.60,1751.40,1649.80,1573.60,1472,1395.80,1395.80,1472,1573↙
.60,1649.80,1751.40,1827.60,1929.20,2005.40,2107,2183.20,2284.80,2361,2361,2284.80,218↙
3.20,2107,2005.40,1929.20,1827.60,1751.40,1649.80,1573.60,1472,1395.80,1395.80,1472,15↙
73.60,1649.80,1751.40,1827.60,1929.20,2005.40,2107,2183.20,2284.80,2361,2437.20,2513.4↙
0,2361,2284.80,2183.20,2107,2005.40,1929.20,1827.60,1751.40,1649.80,1573.60,1472,1395.↙
80,2437.20,2513.40,2742,2767.40,2843.60,2869,2869,2818.20,2767.40,2716.60,1853,1751.40↙
,1853,1751.40,1649.80,1573.60,1472,1395.80,1395.80,1472,1573.60,1649.80,1751.40,1853,1↙
929.20,2005.40,2107,2183.20,2284.80,2361,2462.60,2538.80,2538.80,2462.60,2361,2284.80,↙
2183.20,2107,2005.40,1929.20,1853,1751.40,1649.80,1573.60,1472,1395.80,1395.80,1472,15↙
73.60,1649.80,1751.40,1853,1929.20,2005.40,2107,2183.20,2284.80,2361,2462.60,2538.80,2↙
869,2843.60,2767.40,2742,2538.80,2462.60,2361,2284.80,2183.20,2107,2005.40,1929.20,185↙
3,1751.40,1649.80,1573.60,1472,1395.80,1395.80,1472,1573.60,1649.80,1929.20,2005.40,21↙
07,2183.20,2284.80,2361,2462.60,2538.80,2538.80,2462.60,2361,2284.80,2183.20,2107,2005↙
.40,1929.20,1649.80,1573.60,1472,1395.80,1395.80,1472,1573.60,1649.80,1929.20,2005.40,↙
2107,2183.20,2284.80,2361,2462.60,2538.80,2538.80,2462.60,2183.20,2107,1649.80,1573.60↙
,1472,1395.80,1395.80,1472,1573.60,1649.80,1929.20,2005.40,2107,2183.20,2284.80,2361,2↙
462.60,2538.80,2640.40,2716.60,3961.20,3885,3834.20,3758,3707.20,3631,3580.20,3504,345↙
3.20,3377,3326.20,3250,3199.20,3123,3072.20,2996,2894.40,2716.60,2640.40,2538.80,2462.↙
60,2361,2284.80,2183.20,2107,2005.40,1929.20,1649.80,1573.60,1472,1395.80,1395.80,1472↙
,1573.60,1649.80,1929.20,2005.40,2107,2183.20,2284.80,2361,2462.60,2538.80,2640.40,271↙
6.60,3961.20,3885,3834.20,3758,3707.20,3631,3580.20,3504,3453.20,3377,3326.20,3250,319↙
9.20,3123,3072.20,2996,2894.40,2716.60,2640.40,2538.80,2462.60,2361,2284.80,2183.20,21↙
07,2005.40,1929.20,1649.80,1573.60,1472,1395.80,1395.80,1472,1573.60,1649.80,1929.20,2↙
005.40,2107,2183.20,2284.80,2361,2462.60,2538.80,2640.40,2716.60,2957.90,3961.20,3885,↙
3834.20,3758,3707.20,3631,3580.20,3504,3453.20,3377,3326.20,3250,3199.20,3123,3072.20,↙
2996,2818.20,2716.60,2640.40,2538.80,2462.60,2361,2284.80,2183.20,2107,2005.40,1929.20↙
,1649.80,1573.60,1472,1395.80,1395.80,1472,1573.60,1649.80,1929.20,2005.40,2107,2183.2↙
0,2284.80,2361,2462.60,2538.80,2640.40,2716.60,2894.40,2869,2957.90,2996,3072.20,3123,↙
3199.20,3250,3326.20,3377,3453.20,3504,3580.20,3631,3707.20,3758,3834.20,3885,3961.20,↙
2894.40,2818.20,3072.20,2538.80,2640.40,2716.60,3758,3148.40,2996,2919.80,2843.60,2716↙
.60,2640.40,2284.80,2132.40,2030.80,1878.40,1764.10,1687.90,1548.20,1395.80,1395.80,15↙
48.20,1687.90,1764.10,1878.40,2030.80,2132.40,2284.80,2538.80,2640.40,2716.60,2843.60,↙
2919.80,3148.40,2996,2919.80,2843.60,2716.60,2640.40,2284.80,2132.40,2030.80,1878.40,1↙
764.10,1687.90,1548.20,1395.80,1395.80,1548.20,1687.90,1764.10,1878.40,2030.80,2132.40↙
,2284.80,3758,3072.20,2538.80,2462.60,2284.80,2132.40,2030.80,1878.40,1764.10,1687.90,↙
1548.20,1395.80,1395.80,1548.20,1687.90,1764.10,1878.40,2030.80,2132.40,2284.80,2462.6↙
0,2538.80,2640.40,2716.60,3072.20,3123,3758,3681.80,3631,2716.60,2640.40,2538.80,2462.↙
60,2284.80,2132.40,2030.80,1878.40,1764.10,1687.90,1548.20,1395.80,1395.80,1548.20,187↙
8.40,2030.80,2132.40,2284.80,2462.60,2538.80,2640.40,2716.60,2716.60,2640.40,2538.80,2↙
462.60,2284.80,2132.40,2030.80,1878.40,1548.20,1395.80,1395.80,1548.20,1878.40,2030.80↙
,2132.40,2284.80,2462.60,2538.80,2640.40,2716.60,3758,3123,3072.20,2716.60,2640.40,253↙
8.80,2462.60,2284.80,2132.40,2030.80,1878.40,1548.20,1395.80,1395.80,1548.20,1878.40,2↙
030.80,2132.40,2284.80,2462.60,2538.80,2640.40,2716.60,3656.40,3681.80,2716.60,2640.40↙
,2538.80,2462.60,2284.80,2132.40,2030.80,1878.40,1751.40,1675.20,1548.20,1395.80,1395.↙
80,1548.20,1675.20,1751.40,1878.40,2030.80,2132.40,2284.80,2462.60,2538.80,2640.40,271↙
6.60,3758,2284.80,2132.40,2030.80,1878.40,1751.40,1675.20,1548.20,1395.80,1395.80,1548↙
.20,1675.20,1751.40,1878.40,2030.80,2132.40,2284.80,2284.80,2132.40,2030.80,1878.40,17↙
51.40,1675.20,1548.20,1395.80,1319.60,1395.80,1548.20,1675.20,1751.40,1878.40,2030.80,↙
2132.40,2284.80,2640.40,2716.60,3758,2716.60,2640.40,2284.80,2132.40,2030.80,1878.40,1↙
751.40,1675.20,1548.20,1395.80,1319.60,1395.80,1548.20,1675.20,1751.40,1878.40,2030.80↙
,2132.40,2284.80,2640.40,2716.60,2640.40,2716.60,3758,2716.60,2640.40,2640.40,2716.60,↙
2640.40,2716.60,3631,3681.80,2716.60,2640.40,3758,2869,2716.60,2640.40,2284.80,2132.40↙
,2030.80,1878.40,1802.20,1649.80,1548.20,1395.80,1395.80,1548.20,1649.80,1802.20,1878.↙
40,2030.80,2132.40,2284.80,2640.40,2716.60,2284.80,2132.40,2030.80,1878.40,1802.20,164↙

```
9.80,1548.20,1395.80,1243.40,1395.80,1548.20,1649.80,1802.20,1878.40,2030.80,2132.40,2↙
284.80,3656.40,3681.80,2869,2538.80,2462.60,2284.80,2132.40,2030.80,1878.40,1802.20,16↙
49.80,1548.20,1395.80,1395.80,1548.20,1649.80,1802.20,1878.40,2030.80,2132.40,2284.80,↙
2462.60,2538.80,2716.60,2640.40,2538.80,2462.60,2284.80,2132.40,2030.80,1878.40,1802.2↙
0,1649.80,1548.20,1395.80,1395.80,1548.20,1649.80,1802.20,1878.40,2030.80,2132.40,2284↙
.80,2462.60,2538.80,2640.40,2716.60,3453.20,3351.60,2716.60,2640.40,2538.80,2462.60,22↙
84.80,2132.40,2030.80,1878.40,1802.20,1649.80,1548.20,1395.80,1395.80,1548.20,1649.80,↙
1802.20,1878.40,2030.80,2132.40,2284.80,2462.60,2538.80,2640.40,2716.60,2869,2818.20,2↙
716.60,2640.40,2538.80,2462.60,2284.80,2132.40,2030.80,1878.40,1802.20,1649.80,1548.20↙
,1395.80,1319.60,1294.20,1268.80,1395.80,1548.20,1649.80,1802.20,1878.40,2030.80,2132.↙
40,2284.80,2462.60,2538.80,2640.40,2716.60,2716.60,2640.40,2538.80,2462.60,2284.80,213↙
2.40,2030.80,1878.40,1802.20,1649.80,1548.20,1395.80,1395.80,1548.20,1649.80,1802.20,1↙
878.40,2030.80,2132.40,2284.80,2462.60,2538.80,2640.40,2716.60,3351.60,3453.20,2869,28↙
18.20,3554.80,3580.20,3605.60,3631,3656.40,3681.80,3707.20,3732.60,3758,3783.40,3808.8↙
0,3834.20,3859.60,3453.20,3402.40,3326.20,3275.40,1522.80,1472,3554.80,3580.20,3605.60↙
,3631,3656.40,3681.80,3707.20,3732.60,3758,3783.40,3808.80,3834.20,3859.60,2498.70,254↙
8.70,2573.70,2598.70,2623.70,2648.70,2673.70,2698.70,2723.70,2748.70,2773.70,2798.70,2↙
823.70,2848.70,2873.70,2898.70,2923.70,2948.70,2973.70,2998.70,3023.70,3048.70,3073.70↙
,3098.70,3123.70,3148.70,3173.70,3198.70,3223.70,3248.70,3273.70,3298.70,3323.70,3348.↙
70,3373.70,3398.70,3423.70,3448.70,3554.80,3580.20,3605.60,3631,3656.40,3681.80,3707.2↙
0,3732.60,3758,3783.40,3808.80,3834.20,3859.60,3859.60,3834.20,3808.80,3783.40,3758,37↙
32.60,3707.20,3681.80,3656.40,3631,3605.60,3580.20,3554.800000000000];
                    y=↙
[0,958.3,958.3,958.3,958.3,958.3,958.3,958.3,958.3,958.3,958.3,958.3,958.3,958.3,958.3↙
,1110.7,1110.7,1110.7,1110.7,1110.7,1110.7,1110.7,1110.7,1110.7,1110.7,1110.7,1110.7,1↙
110.7,1110.7,1186.9,1186.9,1186.9,1186.9,1186.9,1186.9,1186.9,1186.9,1186.9,1186.9,118↙
6.9,1186.9,1186.9,1186.9,1339.3,1339.3,1339.3,1339.3,1339.3,1339.3,1339.3,1339.3,1339.↙
3,1339.3,1339.3,1339.3,1339.3,1339.3,1415.5,1415.5,1415.5,1415.5,1415.5,1415.5,1415.5,↙
1415.5,1415.5,1415.5,1415.5,1415.5,1415.5,1415.5,1567.9,1567.9,1567.9,1567.9,1567.9,15↙
67.9,1567.9,1567.9,1567.9,1567.9,1567.9,1567.9,1567.9,1567.9,1644.1,1644.1,1644.1,1644↙
.1,1644.1,1644.1,1644.1,1644.1,1644.1,1644.1,1644.1,1644.1,1644.1,1644.1,1796.5,1796.5↙
,1796.5,1796.5,1796.5,1796.5,1796.5,1796.5,1796.5,1796.5,1796.5,1796.5,1796.5,1796.5,1↙
898.1,1898.1,1898.1,1898.1,1898.1,1898.1,1898.1,1898.1,1898.1,1898.1,1898.1,1898.1,189↙
8.1,1898.1,2050.5,2050.5,2050.5,2050.5,2050.5,2050.5,2050.5,2050.5,2050.5,2050.5,2050.↙
5,2050.5,2050.5,2050.5,2126.7,2126.7,2126.7,2126.7,2126.7,2126.7,2126.7,2126.7,2126.7,↙
2126.7,2126.7,2126.7,2126.7,2126.7,2279.1,2279.1,2279.1,2279.1,2279.1,2279.1,2279.1,22↙
79.1,2279.1,2279.1,2279.1,2279.1,2279.1,2279.1,2355.3,2355.3,2355.3,2355.3,2355.3,2355↙
.3,2355.3,2355.3,2355.3,2355.3,2355.3,2355.3,2355.30,2355.30,2507.70,2507.70,2507.70,2↙
507.70,2507.70,2507.70,2507.70,2507.70,2507.70,2507.70,2507.70,2507.70,2507.70,2507.70↙
,2583.90,2583.90,2583.90,2583.90,2583.90,2583.90,2583.90,2583.90,2583.90,2583.90,2583.↙
90,2583.90,2583.90,2583.90,2583.90,2736.30,2736.30,2736.30,2736.30,2736.30,2736.30,273↙
6.30,2736.30,2736.30,2736.30,2736.30,2736.30,2736.30,2736.30,2888.70,2888.70,2888.70,2↙
888.70,2914.10,2914.10,2914.10,2914.10,2914.10,2939.50,2939.50,2939.50,2939.50,2964.90↙
,2964.90,2964.90,2964.90,2964.90,2990.30,2990.30,2990.30,2990.30,3015.70,3015.70,3015.↙
70,3015.70,3041.10,3041.10,3041.10,3041.10,3066.50,3066.50,3066.50,3066.50,874.500,874↙
.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,8↙
74.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500↙
,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.5↙
00,874.500,874.500,874.500,874.500,899.500,899.500,899.500,899.500,899.500,899.500,899↙
.500,899.500,899.500,899.500,899.500,899.500,899.500,899.500,899.500,899.500,899.500,8↙
99.500,899.500,899.500,899.500,899.500,899.500,899.500,899.500,899.500,899.500,899.500↙
,899.500,899.500,899.500,899.500,899.500,899.500,899.500,899.500,899.500,899.500,899.5↙
00,899.500,899.500,899.500,899.500,899.500,924.500,924.500,924.500,924.500,924.500,924↙
.500,924.500,924.500,924.500,924.500,924.500,924.500,924.500,924.500,924.500,924.500,9↙
24.500,924.500,924.500,924.500,924.500,924.500,924.500,924.500,924.500,924.500,924.500↙
```

```
,↙
924.500,924.500,924.500,924.500,924.500,924.500,924.500,924.500,924.500,924.500,924.50↙
0,924.500,924.500,949.500,949.500,949.500,949.500,996.400,996.400,1085.30,1110.70,1110↙
.70,1110.70,1110.70,1110.70,1110.70,1110.70,1136.10,1136.10,1136.10,1136.10,11↙
36.10,1136.10,1136.10,1136.10,1136.10,1136.10,1136.10,1136.10,1161.50,1161.50,↙
1161.50,1161.50,1161.50,1161.50,1161.50,1161.50,1161.50,1161.50,1161.50,1186.9↙
0,1186.90,1186.90,1186.90,1186.90,1186.90,1186.90,1186.90,1186.90,1186.90,1186↙
.90,1186.90,1186.90,1186.90,1212.30,1212.30,1212.30,1212.30,1212.30,1212.30,12↙
12.30,1212.30,1212.30,1212.30,1212.30,1212.30,1237.70,1237.70,1237.70,1237.70,↙
1237.70,1237.70,1237.70,1237.70,1237.70,1237.70,1237.70,1263.10,1263.10,1263.1↙
0,1263.10,1263.10,1263.10,1263.10,1263.10,1263.10,1263.10,1263.10,1263.10,1288↙
.50,1288.50,1288.50,1288.50,1288.50,1288.50,1288.50,1288.50,1288.50,1288.50,12↙
88.50,1313.90,1313.90,1313.90,1313.90,1313.90,1313.90,1364.70,1364.70,1364.70,↙
1364.70,1364.70,1390.10,1390.10,1390.10,1390.10,1390.10,1390.10,1415.50,1415.5↙
0,1415.50,1415.50,1415.50,1415.50,1415.50,1415.50,1415.50,1415.50,1415.50,1415↙
.50,1440.90,1440.90,1440.90,1440.90,1440.90,1440.90,1440.90,1440.90,1440.90,14↙
40.90,1440.90,1440.90,1440.90,1466.30,1466.30,1466.30,1466.30,1466.30,1466.30,↙
1466.30,1466.30,1466.30,1466.30,1466.30,1466.30,1466.30,1491.70,1491.70,1491.7↙
0,1491.70,1491.70,1491.70,1491.70,1491.70,1491.70,1491.70,1491.70,1491.70,1491↙
.70,1491.70,1491.70,1491.70,1517.10,1517.10,1517.10,1517.10,1517.10,1517.10,15↙
17.10,1517.10,1517.10,1517.10,1517.10,1542.50,1542.50,1542.50,1542.50,1542.50,↙
1542.50,1542.50,1542.50,1542.50,1542.50,1542.50,1567.90,1567.90,1567.90,1567.9↙
0,1567.90,1567.90,1567.90,1567.90,1567.90,1567.90,1567.90,1694.90,1694.90,1694↙
.90,1694.90,1694.90,1694.90,1694.90,1720.30,1720.30,1720.30,1720.30,1720.30,17↙
20.30,1720.30,1720.30,1720.30,1720.30,1720.30,1720.30,1720.30,1745.70,1745.70,↙
1745.70,1745.70,1745.70,1745.70,1745.70,1745.70,1745.70,1745.70,1745.70,1745.7↙
0,1745.70,1745.70,1745.70,1745.70,1745.70,1745.70,1745.70,1745.70,1745.70,1745↙
.70,1745.70,1745.70,1745.70,1745.70,1745.70,1745.70,1771.10,1771.10,1771.10,17↙
71.10,1771.10,1771.10,1771.10,1771.10,1771.10,1771.10,1771.10,1771.10,1796.50,↙
1796.50,1796.50,1796.50,1796.50,1796.50,1796.50,1796.50,1796.50,1796.50,1796.5↙
0,1796.50,1796.50,1796.50,1796.50,1796.50,1796.50,1796.50,1796.50,1796.50,1796↙
.50,1796.50,1796.50,1796.50,1796.50,1796.50,1796.50,1796.50,1796.50,1821.90,18↙
21.90,1821.90,1821.90,1821.90,1821.90,1821.90,1821.90,1821.90,1821.90,1821.90,↙
1821.90,1821.90,1847.30,1847.30,1847.30,1847.30,1847.30,1847.30,1847.30,1847.3↙
0,1847.30,1847.30,1847.30,1847.30,1847.30,1847.30,1847.30,1847.30,1847.30,1847↙
.30,1847.30,1847.30,1847.30,1847.30,1847.30,1847.30,1847.30,1847.30,1847.30,18↙
47.30,1872.70,1872.70,1872.70,1872.70,1872.70,1872.70,1872.70,1872.70,1872.70,↙
1872.70,1872.70,1872.70,1872.70,1910.80,1923.50,1923.50,1923.50,1923.50,1923.5↙
0,1923.50,1923.50,1923.50,1923.50,1923.50,1923.50,1923.50,1923.50,1923.50,1923↙
.50,1923.50,1936.20,1948.90,1961.60,1974.30,1974.30,1974.30,1974.30,1999.70,19↙
99.70,1999.70,1999.70,1999.70,1999.70,1999.70,1999.70,1999.70,1999.70,1999.70,↙
1999.70,2025.10,2025.10,2025.10,2025.10,2025.10,2025.10,2025.10,2025.10,2025.1↙
0,2025.10,2025.10,2025.10,2050.50,2050.50,2050.50,2050.50,2050.50,2050.50,2050↙
.50,2050.50,2050.50,2050.50,2050.50,2050.50,2050.50,2075.90,2075.90,2075.90,20↙
75.90,2075.90,2075.90,2075.90,2075.90,2088.60,2101.30,2101.30,2101.30,2101.30,↙
2101.30,2101.30,2101.30,2101.30,2101.30,2126.70,2126.70,2126.70,2126.70,2126.7↙
0,2126.70,2126.70,2126.70,2126.70,2126.70,2126.70,2126.70,2126.70,2126.70,2152↙
.10,2152.10,2152.10,2152.10,2152.10,2152.10,2152.10,2152.10,2152.10,2152.10,21↙
52.10,2152.10,2177.50,2177.50,2177.50,2177.50,2177.50,2177.50,2177.50,2177.50,↙
2177.50,2202.90,2202.90,2202.90,2202.90,2202.90,2202.90,2202.90,2202.90,2202.9↙
0,2228.30,2228.30,2228.30,2228.30,2228.30,2228.30,2228.30,2228.30,2228.30,2253↙
.70,2253.70,2253.70,2253.70,2253.70,2253.70,2253.70,2253.70,2253.70,2253.70,22↙
53.70,2253.70,2279.10,2279.10,2279.10,2279.10,2279.10,2279.10,2279.10,2279.10,↙
2279.10,2279.10,2279.10,2304.50,2304.50,2304.50,2304.50,2304.50,2304.50,2304.5↙
0,2304.50,2304.50,2304.50,2304.50,2329.90,2329.90,2329.90,2329.90,2329.90,2329↙
```

```
.↙
90,2329.90,2329.90,2329.90,2329.90,2329.90,2355.30,2355.30,2355.30,2355.30,2355.30,235↙
5.30,2355.30,2355.30,2355.30,2380.70,2380.70,2380.70,2380.70,2380.70,2380.70,2380.70,2↙
380.70,2406.10,2406.10,2406.10,2406.10,2406.10,2406.10,2406.10,2406.10,2418.80,2431.50↙
,2431.50,2431.50,2431.50,2431.50,2431.50,2431.50,2431.50,2431.50,2431.50,2456.90,2456.↙
90,2456.90,2456.90,2456.90,2456.90,2456.90,2456.90,2456.90,2456.90,2456.90,2469.60,248↙
2.30,2482.30,2482.30,2482.30,2482.30,2482.30,2482.30,2482.30,2482.30,2482.30,2507.70,2↙
507.70,2507.70,2533.10,2533.10,2558.50,2558.50,2583.90,2583.90,2583.90,2583.90,2609.30↙
,2609.30,2634.70,2634.70,2634.70,2634.70,2634.70,2634.70,2634.70,2634.70,2634.70,2634.↙
70,2634.70,2634.70,2660.10,2660.10,2660.10,2660.10,2660.10,2660.10,2660.10,2660.10,266↙
0.10,2660.10,2685.50,2685.50,2685.50,2685.50,2685.50,2685.50,2685.50,2685.50,2685.50,2↙
710.90,2710.90,2710.90,2710.90,2710.90,2710.90,2710.90,2710.90,2710.90,2710.90,2736.30↙
,2736.30,2736.30,2736.30,2736.30,2736.30,2736.30,2736.30,2736.30,2736.30,2761.70,2761.↙
70,2761.70,2761.70,2761.70,2761.70,2761.70,2761.70,2761.70,2761.70,2787.10,2787.10,278↙
7.10,2787.10,2787.10,2787.10,2787.10,2787.10,2787.10,2787.10,2787.10,2787.10,2812.50,2↙
812.50,2812.50,2812.50,2812.50,2812.50,2812.50,2812.50,2812.50,2812.50,2812.50,2812.50↙
,2812.50,2837.90,2837.90,2837.90,2837.90,2837.90,2837.90,2837.90,2837.90,2837.90,2837.↙
90,2837.90,2837.90,2837.90,2837.90,2863.30,2863.30,2863.30,2863.30,2863.30,2863.30,286↙
3.30,2863.30,2863.30,2863.30,2863.30,2863.30,2888.70,2888.70,2888.70,2888.70,2888.70,2↙
888.70,2888.70,2888.70,2888.70,2888.70,2888.70,2888.70,2888.70,2888.70,2901.40,2901.40↙
,2901.40,2914.10,2914.10,2914.10,2914.10,2914.10,2914.10,2914.10,2914.10,2914.10,2914.↙
10,2914.10,2914.10,2939.50,2939.50,2939.50,2939.50,2939.50,2939.50,2939.50,2939.50,293↙
9.50,2939.50,2939.50,2939.50,2964.90,2964.90,2964.90,2964.90,2964.90,2964.90,2964.90,2↙
964.90,2964.90,2964.90,2964.90,2964.90,2964.90,2964.90,2964.90,2990.30,2990.30,3015.70,3015.70↙
,3015.70,3015.70,3015.70,3015.70,3015.70,3015.70,3015.70,3015.70,3015.70,3015.70,3015.↙
70,3041.10,3041.10,3041.10,3041.10,3041.10,3041.10,3066.50,3066.50,3066.50,3066.50,306↙
6.50,3066.50,3066.50,3066.50,3066.50,3066.50,3066.50,3066.50,3066.50,874.500,874.500,8↙
74.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500↙
,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.5↙
00,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874.500,874↙
.500,874.500,874.500,874.500,907.500,907.500,907.500,907.500,907.500,907.500,907.500,9↙
07.500,907.500,907.500,907.500,907.500,907.500,958.300,958.300,958.300,958.300,958.300↙
,958.300,958.300,958.300,958.300,958.300,958.300,958.300,958.300000000000];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Eil51',
                    obj.dim=51;
                    obj.optima=426;
                    x=↙
[37,49,52,20,40,21,17,31,52,51,42,31,5,12,36,52,27,17,13,57,62,42,16,8,7,27,30,43,58,5↙
8,37,38,46,61,62,63,32,45,59,5,10,21,5,30,39,32,25,25,48,56,30];
                    y=↙
[52,49,64,26,30,47,63,62,33,21,41,32,25,42,16,41,23,33,13,58,42,57,57,52,38,68,48,67,4↙
8,27,69,46,10,33,63,69,22,35,15,6,17,10,64,15,10,39,32,55,28,37,40];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Eil76',
                    obj.dim=76;
                    obj.optima=538;
                    x=↙
[22,36,21,45,55,33,50,55,26,40,55,35,62,62,62,21,33,9,62,66,44,26,11,7,17,41,55,35,52,↙
```

```
43,31,22,26,50,55,54,60,47,30,30,12,15,16,21,50,51,50,48,12,15,29,54,55,67,10,6,65,40,↙
70,64,36,30,20,15,50,57,45,38,50,66,59,35,27,40,40,40];
                    y=↙
[22,26,45,35,20,34,50,45,59,66,65,51,35,57,24,36,44,56,48,14,13,13,28,43,64,46,34,16,2↙
6,26,76,53,29,40,50,10,15,66,60,50,17,14,19,48,30,42,15,21,38,56,39,38,57,41,70,25,27,↙
60,64,4,6,20,30,5,70,72,42,33,4,8,5,60,24,20,37,40];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Fl1577',
                    obj.dim=1577;
                    obj.optima=22249;
                    x =↙
[1214.88,1226.70,1238.53,1250.35,1262.18,1285.82,1297.65,1350.85,1380.41,1392.24,1303.↙
56,1173.50,1185.32,1197.15,1208.97,1339.03,1350.85,1362.68,1374.50,1386.32,1398.15,126↙
2.18,1250.35,1238.53,1226.70,1214.88,1155.76,1143.94,1132.11,1120.29,1108.47,1138.03,1↙
149.85,1161.67,1173.50,1185.32,1303.56,1315.38,1327.20,1339.03,1350.85,1356.77,1344.94↙
,1333.12,1321.29,1309.47,1214.88,1203.06,1191.23,1179.41,1197.15,1208.97,1279.91,1291.↙
73,1303.56,1386.32,1398.15,1409.97,1226.70,1214.88,1203.06,1191.23,1078.91,1090.73,110↙
2.56,1114.38,1126.20,1138.03,1149.85,1256.26,1268.09,1279.91,1291.73,1303.56,1339.03,1↙
350.85,1362.68,1344.94,1333.12,1321.29,1309.47,1279.91,1268.09,1256.26,1244.44,1149.85↙
,1138.03,1126.20,1114.38,1102.56,1090.73,1285.82,1297.65,1309.47,1321.29,1409.97,1398.↙
15,1386.32,1303.56,1291.73,1279.91,1126.20,1114.38,1102.56,1090.73,1333.12,1344.94,135↙
6.77,1368.59,1380.41,1374.50,1362.68,1350.85,1339.03,1327.20,1244.44,1232.62,1220.79,1↙
208.97,1197.15,1185.32,1173.50,1161.67,1149.85,1138.03,1214.88,1226.70,1238.53,1250.35↙
,1262.18,1274,1285.82,1297.65,1309.47,1664.18,670.992,676.904,688.728,700.551,712.375,↙
724.199,736.022,1670.09,1681.92,1693.74,1705.56,1717.39,1729.21,1865.18,1853.36,871.99↙
4,860.171,777.405,789.229,801.053,812.876,824.700,1770.59,1782.42,1794.24,1806.06,1817↙
.89,1244.44,1232.62,1220.79,1208.97,1197.15,1185.32,1173.50,1161.67,1149.85,1138.03,11↙
26.20,1114.38,1102.56,1090.73,1078.91,1067.08,1055.26,1043.44,1031.61,1019.79,1007.97,↙
996.143,984.319,972.496,960.672,948.848,937.025,925.201,913.377,901.554,889.730,877.90↙
6,866.083,854.259,842.435,830.612,818.788,806.964,795.141,783.317,771.494,759.670,747.↙
846,736.022,724.199,712.375,700.551,688.728,676.904,665.081,653.257,641.433,629.609,61↙
7.786,605.962,594.138,582.315,570.491,558.668,546.844,535.020,523.196,511.373,499.549,↙
487.725,475.902,464.078,452.255,440.431,428.607,416.784,404.960,393.136,381.313,369.48↙
9,357.665,345.842,375.401,387.224,399.048,410.872,422.696,434.519,446.343,458.166,469.↙
990,481.814,493.637,505.461,517.285,529.108,540.932,552.756,564.579,576.403,588.227,60↙
0.050,611.874,623.698,635.521,647.345,659.169,670.992,682.816,694.640,706.463,718.287,↙
730.111,741.934,753.758,765.581,777.405,789.229,801.053,812.876,824.700,836.524,848.34↙
7,860.171,871.994,883.818,895.642,907.466,919.289,931.113,942.937,954.760,966.584,978.↙
407,990.231,1002.05,1013.88,1025.70,1037.53,1049.35,1061.17,1073,1084.82,1096.64,1108.↙
47,1120.29,1132.11,1143.94,1155.76,1167.59,1179.41,1191.23,1203.06,1214.88,2066.18,205↙
4.36,2042.54,2030.71,2018.89,2007.07,1995.24,1983.42,1971.59,1959.77,1947.95,1936.12,1↙
924.30,1912.48,1900.65,1888.83,1877.01,1865.18,1853.36,1841.54,1829.71,1817.89,1806.06↙
,1794.24,1782.42,1770.59,1758.77,1746.95,1735.12,1723.30,1711.47,1699.65,1687.83,1676,↙
1664.18,1652.36,1640.53,1628.71,1616.89,1605.06,1593.24,1581.41,1569.59,1557.77,1545.9↙
4,1534.12,1522.30,1510.47,1498.65,1486.82,1475,1463.18,1451.35,1439.53,1427.71,1415.88↙
,1404.06,1392.24,1380.41,1368.59,1356.77,1344.94,1333.12,1321.29,1309.47,1297.65,1285.↙
82,1274,1262.18,1250.35,1238.53,1226.70,1214.88,1256.26,1268.09,1279.91,1291.73,1303.5↙
6,1315.38,1327.20,1339.03,1350.85,1362.68,1374.50,1386.32,1398.15,1409.97,1421.80,1433↙
.62,1445.44,1457.27,1469.09,1480.91,1492.74,1504.56,1516.38,1528.21,1540.03,1551.85,15↙
63.68,1575.50,1587.33,1599.15,1610.97,1622.80,1634.62,1646.44,1658.27,1670.09,1681.92,↙
1693.74,1705.56,1717.39,1729.21,1741.03,1752.86,1764.68,1776.51,1782.42,1770.59,1758.7↙
7,1746.95,1735.12,1723.30,1711.47,1699.65,1687.83,1676,1664.18,1652.36,1368.59,1356.77↙
```

```
,↙
1344.94,1333.12,1321.29,1309.47,1297.65,1374.50,1386.32,1398.15,1409.97,1421.80,1433.6↙
2,1445.44,1457.27,1469.09,1480.91,1492.74,1504.56,1516.38,1528.21,1540.03,1551.85,1563↙
.68,1575.50,1587.33,1599.15,1610.97,1622.80,1634.62,1646.44,1658.27,1670.09,1681.92,16↙
93.74,1705.56,1717.39,1729.21,1741.03,1752.86,1764.68,1776.51,1788.33,1800.15,1811.98,↙
1823.80,1835.62,1847.45,1788.33,1776.51,1764.68,1752.86,1741.03,1729.21,1717.39,1705.5↙
6,1693.74,1681.92,1670.09,1658.27,1646.44,1782.42,1794.24,1806.06,1817.89,1817.89,1806↙
.06,1794.24,1782.42,1770.59,1758.77,1746.95,1735.12,1723.30,1711.47,1699.65,1687.83,16↙
76,1664.18,1652.36,1640.53,1628.71,1616.89,1605.06,1593.24,1581.41,1569.59,1557.77,154↙
5.94,1534.12,1522.30,1510.47,1498.65,1486.82,1475,1463.18,1451.35,1439.53,1427.71,1415↙
.88,1404.06,1392.24,1380.41,1368.59,1356.77,1344.94,1333.12,1321.29,1309.47,1297.65,12↙
85.82,1274,1262.18,1250.35,1238.53,1120.29,1132.11,1143.94,1155.76,1167.59,1179.41,119↙
1.23,1203.06,1214.88,1226.70,1238.53,1250.35,1262.18,1274,1285.82,1297.65,1309.47,1321↙
.29,1333.12,1344.94,1356.77,1368.59,1380.41,1392.24,1404.06,1415.88,1427.71,1439.53,14↙
51.35,1463.18,1475,1486.82,1498.65,1510.47,1522.30,1534.12,1545.94,1557.77,1569.59,158↙
1.41,1593.24,1605.06,1616.89,1628.71,1640.53,1652.36,1664.18,1676,1687.83,1699.65,1711↙
.47,1723.30,1735.12,1800.15,1788.33,1776.51,1764.68,1752.86,1741.03,1729.21,1717.39,17↙
05.56,1693.74,1681.92,1670.09,1658.27,1646.44,1634.62,1622.80,1610.97,1599.15,1587.33,↙
1575.50,1563.68,1551.85,1540.03,1528.21,1516.38,1504.56,1492.74,1480.91,1469.09,1457.2↙
7,1445.44,1433.62,1421.80,1409.97,1398.15,1386.32,1374.50,1362.68,1350.85,1339.03,1327↙
.20,1315.38,1303.56,1291.73,1279.91,1268.09,1256.26,1244.44,1238.53,1250.35,1262.18,12↙
74,1285.82,1297.65,1309.47,1321.29,1333.12,1344.94,1356.77,1368.59,1380.41,1392.24,140↙
4.06,1415.88,1427.71,1439.53,1451.35,1463.18,1475,1486.82,1498.65,1510.47,1522.30,1534↙
.12,1545.94,1557.77,1569.59,1581.41,1593.24,1605.06,1616.89,1628.71,1640.53,1652.36,16↙
64.18,1676,1687.83,1699.65,1711.47,1723.30,1735.12,1746.95,1758.77,1770.59,1782.42,185↙
9.27,1847.45,1835.62,1823.80,1811.98,1800.15,1788.33,1776.51,1764.68,1752.86,1741.03,1↙
729.21,1717.39,1705.56,1693.74,1681.92,1670.09,1658.27,1646.44,1634.62,1622.80,1610.97↙
,1599.15,1587.33,1575.50,1563.68,1551.85,1540.03,1528.21,1516.38,1504.56,1492.74,1480.↙
91,1469.09,1457.27,1445.44,1433.62,1421.80,1409.97,1398.15,1386.32,1132.11,1143.94,115↙
5.76,1167.59,1179.41,1191.23,1203.06,1214.88,1226.70,1238.53,1250.35,1262.18,1274,1285↙
.82,1297.65,1309.47,1321.29,1333.12,1344.94,1356.77,1368.59,1380.41,1392.24,1404.06,14↙
15.88,1427.71,1439.53,1451.35,1463.18,1475,1486.82,1498.65,1510.47,1522.30,1534.12,154↙
5.94,1557.77,1569.59,1581.41,1593.24,1605.06,1616.89,1628.71,1640.53,1652.36,1664.18,1↙
676,1687.83,1699.65,1711.47,1723.30,1735.12,1746.95,1232.62,1220.79,1208.97,1197.15,11↙
85.32,1173.50,1161.67,1149.85,1138.03,1126.20,1114.38,1102.56,1090.73,1078.91,1067.08,↙
1055.26,1043.44,1031.61,1019.79,1007.97,996.143,984.319,972.496,960.672,948.848,937.02↙
5,925.201,913.377,901.554,889.730,877.906,866.083,854.259,842.435,830.612,818.788,806.↙
964,795.141,783.317,771.494,759.670,747.846,736.022,724.199,712.375,700.551,688.728,67↙
6.904,665.081,653.257,641.433,629.609,617.786,605.962,594.138,582.315,570.491,558.668,↙
546.844,535.020,523.196,511.373,499.549,487.725,475.902,464.078,452.255,440.431,428.60↙
7,416.784,404.960,393.136,381.313,369.489,357.665,345.842,334.018,1120.29,1132.11,1143↙
.94,1155.76,1167.59,1179.41,1191.23,1203.06,1214.88,1226.70,1238.53,1250.35,1262.18,12↙
74,1285.82,1297.65,1309.47,1321.29,1333.12,1344.94,1356.77,1368.59,1380.41,1392.24,140↙
4.06,1415.88,1427.71,1439.53,1451.35,1463.18,1475,1486.82,1498.65,1510.47,1522.30,1534↙
.12,1545.94,1557.77,1569.59,1581.41,1593.24,1605.06,1616.89,1628.71,1640.53,1652.36,16↙
64.18,1676,1687.83,1699.65,1711.47,1723.30,1735.12,1226.70,1214.88,1203.06,1191.23,117↙
9.41,1220.79,1232.62,1244.44,1415.88,1404.06,1392.24,1380.41,1368.59,1356.77,1344.94,1↙
309.47,1297.65,1285.82,1274,1262.18,1250.35,1214.88,1203.06,1191.23,1179.41,1167.59,11↙
32.11,1120.29,1108.47,1096.64,1084.82,1073,1090.73,1102.56,1114.38,1126.20,1138.03,114↙
9.85,1161.67,1173.50,1185.32,1197.15,1220.79,1232.62,1244.44,1256.26,1268.09,1279.91,1↙
339.03,1350.85,1362.68,1374.50,1380.41,1368.59,1356.77,1344.94,1333.12,1262.18,1250.35↙
,1238.53,1226.70,1167.59,1155.76,1143.94,1132.11,1208.97,1220.79,1232.62,1244.44,1291.↙
73,1303.56,1315.38,1327.20,1339.03,1350.85,1380.41,1368.59,1356.77,1262.18,1250.35,123↙
8.53,1226.70,1179.41,1167.59,1155.76,1143.94,1132.11,1120.29,1108.47,1096.64,1084.82,1↙
073,1061.17,1049.35,1037.53,1025.70,1013.88,1002.05,990.231,978.407,966.584,954.760,94↙
```

```
2.937,931.113,919.289,907.466,895.642,883.818,871.994,860.171,848.347,836.524,824.700,↙
812.876,801.053,789.229,777.405,765.581,753.758,741.934,1173.50,1279.91,1291.73,1303.5↙
6,1315.38,1339.03,1350.85,1362.68,1374.50,1386.32,1398.15,1409.97,1321.29,1226.70,1214↙
.88,1203.06,1090.73,1102.56,1114.38,1126.20,1138.03,1208.97,1220.79,1232.62,1256.26,12↙
68.09,1279.91,1291.73,1350.85,1333.12,1321.29,1309.47,1214.88,1203.06,1191.23,1143.94,↙
1138.03,1149.85,1232.62,1244.44,1368.59,1356.77,1274,1167.59,1155.76,1143.94,1132.11,1↙
120.29,1108.47,1386.32,1398.15,1392.24,1380.41,1368.59,795.141,806.964,818.788,830.612↙
,842.435,854.259,866.083,877.906,889.730,901.554,913.377,925.201,937.025,948.848,960.6↙
72,972.496,984.319,996.143,1007.97,1019.79,1031.61,1043.44,1055.26,1067.08,1078.91,109↙
0.73,1102.56,1114.38,1126.20,1138.03,1149.85,1161.67,1173.50,1185.32,1197.15,1208.97,1↙
220.79,1368.59,1380.41,1392.24,1404.06,1415.88,1427.71,1439.53,1451.35,1463.18,1475,14↙
86.82,1498.65,1510.47,1522.30,1534.12,1545.94,1557.77,1569.59,1581.41,1593.24,1605.06,↙
1616.89,1628.71,1640.53,1652.36,1664.18,1676,1687.83,1699.65,1711.47,1723.30,1735.12,1↙
746.95,1758.77,1770.59,1782.42,1794.24,1806.06,1817.89,1829.71,1841.54,1853.36,1865.18↙
,1877.01,1888.83,1900.65,1912.48,1924.30,1936.12,1947.95,1959.77,1971.59,1983.42,1995.↙
24,2007.07,2018.89,2030.71,2042.54,2054.36,2066.18,2078.01,2089.83,1220.79,1208.97,119↙
7.15,1185.32,1888.83,1877.01,1865.18,1853.36,1841.53,1829.71,1817.89,1806.06,1794.24,1↙
782.42,1770.59,1758.77,1723.30,1711.47,1664.18,1640.53,1616.89,1605.06,1569.59,1557.77↙
,1534.12,1522.30,1510.47,1498.65,907.466,895.642,883.818,871.994,860.171,848.347,836.5↙
24,824.700,812.876,801.053,789.229,777.405,741.934,730.111,682.816,659.169,635.521,623↙
.698,588.227,576.403,552.756,540.932,529.108,517.285,511.373,523.196,570.491,582.315,5↙
94.138,629.609,641.433,700.551,712.375,771.494,783.317,795.141,806.964,818.788,830.612↙
,842.435,854.259,866.083,877.906,889.730,901.554,1492.74,1504.56,1551.85,1563.68,1575.↙
50,1610.97,1622.80,1681.92,1693.74,1752.86,1764.68,1776.50,1788.33,1800.15,1811.97,182↙
3.80,1835.62,1847.45,1859.27,1871.09,1882.92,1888.83,1877.01,1865.18,1853.36,1841.53,1↙
829.71,1817.89,1806.06,1794.24,1782.42,1770.59,1758.77,1711.47,1699.65,1687.83,1652.36↙
,1640.53,1557.77,1545.94,1534.12,1498.65,907.466,895.642,883.818,871.994,860.171,848.3↙
47,836.524,824.700,812.876,801.053,789.229,777.405,730.111,718.287,706.463,670.992,659↙
.169,576.403,564.579,552.756,517.285,511.373,523.196,546.844,558.668,582.315,605.962,6↙
29.609,653.257,665.081,688.728,700.551,712.375,724.199,736.022,747.846,771.494,783.317↙
,795.141,806.964,818.788,830.612,842.435,854.259,866.083,877.906,889.730,901.554,1492.↙
74,1504.56,1528.21,1540.03,1563.68,1587.33,1610.97,1634.62,1646.44,1670.09,1681.92,169↙
3.74,1705.56,1717.39,1729.21,1752.86,1764.68,1776.50,1788.33,1800.15,1811.97,1823.80,1↙
835.62,1847.45,1859.27,1871.09,1882.92,1888.83,1877.01,1865.18,1853.36,1841.53,1829.71↙
,1817.89,1806.06,1794.24,1782.42,1770.59,1758.77,907.466,895.642,883.818,871.994,860.1↙
71,848.347,836.524,824.700,812.876,801.053,789.229,777.405,517.285,529.108,540.932,552↙
.756,576.403,588.227,623.698,635.521,659.169,682.816,730.111,741.934,1498.65,1510.47,1↙
522.30,1534.12,1557.77,1569.59,1605.06,1616.89,1640.53,1664.18,1711.47,1723.30,1882.92↙
,1871.09,1859.27,1847.45,1835.62,1823.80,1811.97,1800.15,1788.33,1776.50,1764.68,1752.↙
86,901.554,889.730,877.906,866.083,854.259,842.435,830.612,818.788,806.964,795.141,783↙
.317,771.494,511.373,523.196,570.491,582.315,594.138,629.609,641.433,700.551,712.375,1↙
492.74,1504.56,1551.85,1563.68,1575.50,1610.97,1622.80,1681.92,1693.74,1888.83,1877.01↙
,1865.18,1853.36,1841.53,1829.71,1817.89,1806.06,1794.24,1782.42,1770.59,1758.77,907.4↙
66,895.642,883.818,871.994,860.171,848.347,836.524,824.700,812.876,801.053,789.229,777↙
.405,517.285,552.756,564.579,576.403,659.169,670.992,706.463,718.287,730.111,1498.65,1↙
534.12,1545.94,1557.77,1640.53,1652.36,1687.83,1699.65,1711.47,1882.92,1871.09,1859.27↙
,1847.45,1835.62,1823.80,1811.97,1800.15,1788.33,1776.50,1764.68,1752.86,901.554,889.7↙
30,877.906,866.083,854.259,842.435,830.612,818.788,806.964,795.141,783.317,771.494,511↙
.373,523.196,546.844,558.668,582.315,605.962,629.609,653.257,665.081,688.728,700.551,7↙
12.375,724.199,736.022,747.846,1492.74,1504.56,1528.21,1540.03,1563.68,1587.33,1610.97↙
,1634.62,1646.44,1670.09,1681.92,1693.74,1705.56,1717.39,1729.21000000000];
                    y =↙
[1959.49,1959.49,1959.49,1959.49,1959.49,1959.49,1959.49,1965.39,1971.30,1971.30,1953.↙
58,1941.77,1941.77,1941.77,1941.77,1941.77,1941.77,1941.77,1941.77,1941.77,1941.77,193↙
5.87,1935.87,1935.87,1935.87,1935.87,1935.87,1935.87,1935.87,1935.87,1935.87,1929.96,1↙
```

```
929.96,1929.96,1929.96,1929.96,1929.96,1929.96,1929.96,1929.96,1929.96,1924.06,1924.06↙
,1924.06,1924.06,1924.06,1924.06,1924.06,1924.06,1924.06,1918.15,1918.15,1918.15,1918.↙
15,1918.15,1918.15,1918.15,1918.15,1912.25,1912.25,1912.25,1912.25,1906.34,1906.34,190↙
6.34,1906.34,1906.34,1906.34,1906.34,1906.34,1906.34,1906.34,1906.34,1906.34,1906.34,1↙
906.34,1906.34,1900.44,1900.44,1900.44,1900.44,1894.53,1894.53,1894.53,1894.53,1894.53↙
,1894.53,1894.53,1894.53,1894.53,1894.53,1888.63,1888.63,1888.63,1888.63,1882.72,1882.↙
72,1882.72,1882.72,1882.72,1882.72,1882.72,1882.72,1882.72,1882.72,1876.82,1876.82,187↙
6.82,1876.82,1876.82,1870.91,1870.91,1870.91,1870.91,1870.91,1870.91,1870.91,1870.91,1↙
870.91,1870.91,1870.91,1870.91,1870.91,1870.91,1870.91,1865.01,1865.01,1865.01,1865.01↙
,1865.01,1865.01,1865.01,1865.01,1865.01,1746.91,1746.91,1741,1741,1741,1741,1741,1741↙
,1741,1741,1741,1741,1741,1741,1723.29,1723.29,1723.29,1723.29,1522.52,1522.52,1522.52↙
,1522.52,1522.52,1522.52,1522.52,1522.52,1522.52,1522.52,1516.61,1516.61,1516.61,1516.↙
61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,151↙
6.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1↙
516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61↙
,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.↙
61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,151↙
6.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1↙
516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1516.61,1487.08,1487.08↙
,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.↙
08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,148↙
7.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1↙
487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08↙
,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.↙
08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,1487.08,148↙
7.08,1487.08,1487.08,1487.08,1487.08,1487.08,1475.28,1475.28,1475.28,1475.28,1475.28,1↙
475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28↙
,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.↙
28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,147↙
5.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1↙
475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28↙
,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.28,1475.↙
28,1475.28,1475.28,1475.28,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75,144↙
5.75,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75,1↙
445.75,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75↙
,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75,1445.75,1445.↙
75,1445.75,1445.75,1445.75,1445.75,1445.75,1416.22,1416.22,1416.22,1416.22,1416.22,141↙
6.22,1416.22,1416.22,1416.22,1416.22,1416.22,1416.22,1416.22,1416.22,1416.22,1416.22,1↙
416.22,1416.22,1416.22,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32↙
,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.↙
32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,141↙
0.32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,1410.32,1↙
410.32,1398.51,1398.51,1398.51,1398.51,1398.51,1398.51,1398.51,1398.51,1398.51,1398.51↙
,1398.51,1398.51,1398.51,1392.60,1392.60,1392.60,1392.60,1274.50,1274.50,1274.50,1274.↙
50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,127↙
4.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1↙
274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50↙
,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.50,1274.↙
50,1274.50,1274.50,1274.50,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,125↙
0.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1↙
250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88↙
,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.↙
88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,1250.88,125↙
0.88,1250.88,1250.88,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1↙
244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98↙
```

```
,↙
1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.9↙
8,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244↙
.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1244.98,1227.26,1227.26,1227.26,12↙
27.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,↙
1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.2↙
6,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227↙
.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,1227.26,12↙
27.26,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,↙
1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.3↙
6,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221↙
.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1221.36,1156.40,11↙
56.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,↙
1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.4↙
0,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156↙
.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,11↙
56.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1156.40,1150.50,1150.50,↙
1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.5↙
0,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150↙
.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,11↙
50.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,↙
1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.5↙
0,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150↙
.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,1150.50,11↙
44.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,↙
1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.5↙
9,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144↙
.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,11↙
44.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1144.59,1002.87,↙
1002.87,1002.87,1002.87,1002.87,996.967,996.967,996.967,802.101,802.101,802.101,802.10↙
1,802.101,802.101,802.101,802.101,802.101,802.101,802.101,802.101,802.101,802.101,802.↙
101,802.101,802.101,802.101,802.101,802.101,802.101,802.101,802.101,802.101,796.196,79↙
6.196,796.196,796.196,796.196,796.196,796.196,796.196,796.196,796.196,796.196,796.196,↙
796.196,796.196,796.196,796.196,796.196,796.196,796.196,796.196,790.290,790.290,790.29↙
0,790.290,790.290,790.290,790.290,790.290,790.290,790.290,790.290,790.290,790.290,784.↙
386,784.386,784.386,784.386,784.386,784.386,784.386,784.386,784.386,784.386,778.481,77↙
8.481,778.481,778.481,778.481,778.481,778.481,778.481,778.481,778.481,778.481,778.481,↙
778.481,778.481,778.481,778.481,778.481,778.481,778.481,778.481,778.481,778.481,778.48↙
1,778.481,778.481,778.481,778.481,778.481,778.481,778.481,778.481,778.481,778.481,778.↙
481,778.481,778.481,778.481,778.481,778.481,778.481,778.481,778.481,778.481,778.481,77↙
8.481,772.576,772.576,772.576,772.576,772.576,772.576,772.576,772.576,772.576,772.576,↙
772.576,772.576,766.670,766.670,766.670,766.670,760.765,760.765,760.765,760.765,760.76↙
5,760.765,760.765,760.765,760.765,760.765,760.765,760.765,760.765,754.861,754.861,754.↙
861,754.861,754.861,754.861,754.861,748.955,748.955,748.955,748.955,743.050,743.050,74↙
3.050,731.240,731.240,731.240,731.240,731.240,731.240,725.335,725.335,719.430,719.430,↙
719.430,548.184,548.184,548.184,548.184,548.184,548.184,548.184,548.184,548.184,548.18↙
4,548.184,548.184,548.184,548.184,548.184,548.184,548.184,548.184,548.184,548.184,548.↙
184,548.184,548.184,548.184,548.184,548.184,548.184,548.184,548.184,548.184,548.184,54↙
8.184,548.184,548.184,548.184,548.184,548.184,601.329,601.329,601.329,601.329,601.329,↙
601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.32↙
9,601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.↙
329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,60↙
1.329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,↙
601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.329,601.32↙
9,601.329,601.329,601.329,536.374,536.374,536.374,536.374,2048.06,2048.06,2048.06,2048↙
```

```
.↙
06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,204↙
8.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2↙
048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06↙
,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.06,2048.↙
06,2048.06,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,204↙
2.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2↙
042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16↙
,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.16,2042.↙
16,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,203↙
6.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2↙
036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25↙
,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2036.25,2030.↙
35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,203↙
0.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2↙
030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35↙
,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.↙
35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,2030.35,170↙
.262,170.262,170.262,170.262,170.262,170.262,170.262,170.262,170.262,170.262,170.262,1↙
70.262,170.262,170.262,170.262,170.262,170.262,170.262,170.262,170.262,170.262,170.262↙
,170.262,170.262,170.261,170.261,170.261,170.261,170.261,170.261,170.261,170.261,170.2↙
61,170.261,170.261,170.261,170.261,170.261,170.261,170.261,170.261,170.261,170.261,170↙
.261,170.261,170.261,170.261,170.261,164.357,164.357,164.357,164.357,164.357,164.357,1↙
64.357,164.357,164.357,164.357,164.357,164.357,164.357,164.357,164.357,164.357,164.357↙
,164.357,164.357,164.357,164.357,164.357,164.357,164.357,164.357,164.357,164.357,164.3↙
57,164.357,164.357,164.357,164.357,164.357,164.357,164.357,164.357,164.357,164.357,164↙
.357,164.357,164.357,164.357,158.452,158.452,158.452,158.452,158.452,158.452,158.452,1↙
58.452,158.452,158.452,158.452,158.452,158.452,158.452,158.452,158.452,158.452,158.452↙
,158.452,158.452,158.452,158.452,158.452,158.452,158.451,158.451,158.451,158.451,158.4↙
51,158.451,158.451,158.451,158.451,158.451,158.451,158.451,158.451,158.451,158.451,158↙
.451,158.451,158.451,152.547,152.547,152.547,152.547,152.547,152.547,152.547,152.547,1↙
52.547,152.547,152.547,152.547,152.547,152.547,152.547,152.547,152.547,152.547,152.547↙
,152.547,152.547,152.547,152.547,152.547,152.546,152.546,152.546,152.546,152.546,152.5↙
46,152.546,152.546,152.546,152.546,152.546,152.546,152.546,152.546,152.546,152.546,152↙
.546,152.546,152.546,152.546,152.546,152.546,152.546,152.546,152.546,152.546,152.546,1↙
52.546,152.546,152.546000000000];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'KroA100',
                    obj.dim=100;
                    obj.optima=21282;
                    x=↙
[1380,2848,3510,457,3888,984,2721,1286,2716,738,1251,2728,3815,3683,1247,123,1234,252,↙
611,2576,928,53,1807,274,2574,178,2678,1795,3384,3520,1256,1424,3913,3085,2573,463,387↙
5,298,3479,2542,3955,1323,3447,2936,1621,3373,1393,3874,938,3022,2482,3854,376,2519,29↙
45,953,2628,2097,890,2139,2421,2290,1115,2588,327,241,1917,2991,2573,19,3911,872,2863,↙
929,839,3893,2178,3822,378,1178,2599,3416,2961,611,3113,2597,2586,161,1429,742,1625,11↙
87,1787,22,3640,3756,776,1724,198,3950];
                    y=↙
[939,96,1671,334,666,965,1482,525,1432,1325,1832,1698,169,1533,1945,862,1946,1240,673,↙
1676,1700,857,1711,1420,946,24,1825,962,1498,1079,61,1728,192,1528,1969,1670,598,1513,↙
821,236,1743,280,1830,337,1830,1646,1368,1318,955,474,1183,923,825,135,1622,268,1479,9↙
81,1846,1806,1007,1810,1052,302,265,341,687,792,599,674,1673,1559,558,1766,620,102,161↙
```

```
9,899,1048,100,901,143,1605,1384,885,1830,1286,906,134,1025,1651,706,1009,987,43,882,3↙
92,1642,1810,1558];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'KroA150',
                    obj.dim=150;
                    obj.optima=26524;
                    x=↙
[1380,2848,3510,457,3888,984,2721,1286,2716,738,1251,2728,3815,3683,1247,123,1234,252,↙
611,2576,928,53,1807,274,2574,178,2678,1795,3384,3520,1256,1424,3913,3085,2573,463,387↙
5,298,3479,2542,3955,1323,3447,2936,1621,3373,1393,3874,938,3022,2482,3854,376,2519,29↙
45,953,2628,2097,890,2139,2421,2290,1115,2588,327,241,1917,2991,2573,19,3911,872,2863,↙
929,839,3893,2178,3822,378,1178,2599,3416,2961,611,3113,2597,2586,161,1429,742,1625,11↙
87,1787,22,3640,3756,776,1724,198,3950,3477,91,3972,198,1806,538,3430,2186,1513,2143,5↙
3,3404,1034,2823,3104,3232,2790,374,741,3083,3502,1280,3326,217,2503,3527,739,3548,48,↙
1419,1689,3468,1628,382,3029,3646,285,1782,1067,2849,920,1741,876,2753,2609,3941,3613,↙
1754,2916,2445];
                    y=↙
[939,96,1671,334,666,965,1482,525,1432,1325,1832,1698,169,1533,1945,862,1946,1240,673,↙
1676,1700,857,1711,1420,946,24,1825,962,1498,1079,61,1728,192,1528,1969,1670,598,1513,↙
821,236,1743,280,1830,337,1830,1646,1368,1318,955,474,1183,923,825,135,1622,268,1479,9↙
81,1846,1806,1007,1810,1052,302,265,341,687,792,599,674,1673,1559,558,1766,620,102,161↙
9,899,1048,100,901,143,1605,1384,885,1830,1286,906,134,1025,1651,706,1009,987,43,882,3↙
92,1642,1810,1558,949,1732,329,1632,733,1023,1088,766,1646,1611,1657,1307,1344,376,193↙
1,324,1457,9,146,1938,1067,237,1846,38,1172,41,1850,1999,154,872,1223,1404,253,872,124↙
2,1758,1029,93,371,1214,1835,712,220,283,1286,258,523,559,1724,1820];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'KroA200',
                    obj.dim=200;
                    obj.optima=29368;
                    x=↙
[1357,2650,1774,1307,3806,2687,43,3092,185,834,40,1183,2048,1097,1838,234,3314,737,779↙
,2312,2576,3078,2781,705,3409,323,1660,3729,693,2361,2433,554,913,3586,2636,1000,482,3↙
704,3635,1362,2049,2552,3939,219,812,901,2513,242,826,3278,86,14,1327,2773,2469,3835,1↙
031,3853,1868,1544,457,3174,192,2318,2232,396,2365,2499,1410,2990,3646,3394,1779,1058,↙
2933,3099,2178,138,2082,2302,805,22,3213,99,1533,3564,29,3808,2221,3499,3124,781,1027,↙
3249,3297,213,721,3736,868,960,1380,2848,3510,457,3888,984,2721,1286,2716,738,1251,272↙
8,3815,3683,1247,123,1234,252,611,2576,928,53,1807,274,2574,178,2678,1795,3384,3520,12↙
56,1424,3913,3085,2573,463,3875,298,3479,2542,3955,1323,3447,2936,1621,3373,1393,3874,↙
938,3022,2482,3854,376,2519,2945,953,2628,2097,890,2139,2421,2290,1115,2588,327,241,19↙
17,2991,2573,19,3911,872,2863,929,839,3893,2178,3822,378,1178,2599,3416,2961,611,3113,↙
2597,2586,161,1429,742,1625,1187,1787,22,3640,3756,776,1724,198,3950];
                    y=↙
[1905,802,107,964,746,1353,1957,1668,1542,629,462,1391,1628,643,1732,1118,1881,1285,77↙
7,1949,189,1541,478,1812,1917,1714,1556,1188,1383,640,1538,1825,317,1909,727,457,1337,↙
1082,1174,1526,417,1909,640,898,351,1552,1572,584,1226,799,1065,454,1893,1286,1838,963↙
,428,1712,197,863,1607,1064,1004,1925,1374,828,1649,658,307,214,1018,1028,90,372,1459,↙
173,978,1610,1753,1127,272,1617,1085,536,1780,676,6,1375,291,1885,408,671,1041,378,491↙
,220,186,1542,731,303,939,96,1671,334,666,965,1482,525,1432,1325,1832,1698,169,1533,19↙
45,862,1946,1240,673,1676,1700,857,1711,1420,946,24,1825,962,1498,1079,61,1728,192,152↙
```

```
8,1969,1670,598,1513,821,236,1743,280,1830,337,1830,1646,1368,1318,955,474,1183,923,82↙
5,135,1622,268,1479,981,1846,1806,1007,1810,1052,302,265,341,687,792,599,674,1673,1559↙
,558,1766,620,102,1619,899,1048,100,901,143,1605,1384,885,1830,1286,906,134,1025,1651,↙
706,1009,987,43,882,392,1642,1810,1558];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'KroB100',
                    obj.dim=100;
                    obj.optima=22141;
                    x=↙
[3140,556,3675,1182,3595,962,2030,3507,2642,3438,3858,2937,376,839,706,749,298,694,387↙
,2801,3133,1517,1538,844,2639,3123,2489,3834,3417,2938,71,3245,731,2312,2426,380,2310,↙
2830,3829,3684,171,627,1490,61,422,2698,2372,177,3084,1213,3,1782,3896,1829,1286,3017,↙
2132,2000,3317,1729,2408,3292,193,782,2503,1697,3821,3370,3162,3938,2741,2330,3918,179↙
4,2929,3453,896,399,2614,2800,2630,563,1090,2009,3876,3084,1526,1612,1423,3058,3782,34↙
7,3904,2191,3220,468,3611,3114,3515,3060];
                    y=↙
[1401,1056,1522,1853,111,1895,1186,1851,1269,901,1472,1568,1018,1355,1925,920,615,552,↙
190,695,1143,266,224,520,1239,217,1520,1827,1808,543,1323,1828,1741,1270,1851,478,635,↙
775,513,445,514,1261,1123,81,542,1221,127,1390,748,910,1817,995,742,812,550,108,1432,1↙
110,1966,1498,1747,152,1210,1462,352,1924,147,791,367,516,1583,741,1088,1589,485,1998,↙
705,850,195,653,20,1513,1652,1163,1165,774,1612,328,1322,1276,1865,252,1444,1579,1454,↙
319,1968,1629,1892,155];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'KroB150',
                    obj.dim=150;
                    obj.optima=26130;
                    x=↙
[1357,2650,1774,1307,3806,2687,43,3092,185,834,40,1183,2048,1097,1838,234,3314,737,779↙
,2312,2576,3078,2781,705,3409,323,1660,3729,693,2361,2433,554,913,3586,2636,1000,482,3↙
704,3635,1362,2049,2552,3939,219,812,901,2513,242,826,3278,86,14,1327,2773,2469,3835,1↙
031,3853,1868,1544,457,3174,192,2318,2232,396,2365,2499,1410,2990,3646,3394,1779,1058,↙
2933,3099,2178,138,2082,2302,805,22,3213,99,1533,3564,29,3808,2221,3499,3124,781,1027,↙
3249,3297,213,721,3736,868,960,3825,2779,201,2502,765,3105,1937,3364,3702,2164,3019,30↙
98,3239,3359,2081,1398,618,1878,3803,397,3035,2502,3230,3479,958,3423,78,96,3431,2053,↙
3048,571,3393,2835,144,923,989,3061,2977,1668,878,678,1086,640,3551,106,2243,3796,2643↙
,48];
                    y=↙
[1905,802,107,964,746,1353,1957,1668,1542,629,462,1391,1628,643,1732,1118,1881,1285,77↙
7,1949,189,1541,478,1812,1917,1714,1556,1188,1383,640,1538,1825,317,1909,727,457,1337,↙
1082,1174,1526,417,1909,640,898,351,1552,1572,584,1226,799,1065,454,1893,1286,1838,963↙
,428,1712,197,863,1607,1064,1004,1925,1374,828,1649,658,307,214,1018,1028,90,372,1459,↙
173,978,1610,1753,1127,272,1617,1085,536,1780,676,6,1375,291,1885,408,671,1041,378,491↙
,220,186,1542,731,303,1101,435,693,1274,833,1823,1400,1498,1624,1874,189,1594,1376,169↙
3,1011,1100,1953,59,886,1217,152,146,380,1023,1670,1241,1066,691,78,1461,1,1711,782,14↙
72,1185,108,1997,1211,39,658,715,1599,868,110,1673,1267,1332,1401,1320,267];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
```

```
                case 'KroB200',
                    obj.dim=200;
                    obj.optima=29437;
                    x=↙
[3140,556,3675,1182,3595,962,2030,3507,2642,3438,3858,2937,376,839,706,749,298,694,387↙
,2801,3133,1517,1538,844,2639,3123,2489,3834,3417,2938,71,3245,731,2312,2426,380,2310,↙
2830,3829,3684,171,627,1490,61,422,2698,2372,177,3084,1213,3,1782,3896,1829,1286,3017,↙
2132,2000,3317,1729,2408,3292,193,782,2503,1697,3821,3370,3162,3938,2741,2330,3918,179↙
4,2929,3453,896,399,2614,2800,2630,563,1090,2009,3876,3084,1526,1612,1423,3058,3782,34↙
7,3904,2191,3220,468,3611,3114,3515,3060,2995,202,981,1346,781,1009,2927,2982,555,464,↙
3452,571,2656,1623,2067,1725,3600,1109,366,778,386,3918,3332,2597,811,241,2658,394,378↙
6,264,2050,3538,1646,2993,547,3373,460,3060,1828,1021,2347,3535,1529,1203,1787,2740,55↙
5,47,3935,3062,387,2901,931,1766,401,149,2214,3805,1179,1017,2834,634,1819,1393,1768,3↙
023,3248,1632,2223,3868,1541,2374,1962,3007,3220,2356,1604,2028,2581,2221,2944,1082,99↙
7,2334,1264,1699,235,2592,3642,3599,1766,240,1272,3503,80,1677,3766,3946,1994,278];
                    y=↙
[1401,1056,1522,1853,111,1895,1186,1851,1269,901,1472,1568,1018,1355,1925,920,615,552,↙
190,695,1143,266,224,520,1239,217,1520,1827,1808,543,1323,1828,1741,1270,1851,478,635,↙
775,513,445,514,1261,1123,81,542,1221,127,1390,748,910,1817,995,742,812,550,108,1432,1↙
110,1966,1498,1747,152,1210,1462,352,1924,147,791,367,516,1583,741,1088,1589,485,1998,↙
705,850,195,653,20,1513,1652,1163,1165,774,1612,328,1322,1276,1865,252,1444,1579,1454,↙
319,1968,1629,1892,155,264,233,848,408,670,1001,1777,949,1121,1302,637,1982,128,1723,6↙
94,927,459,1196,339,1282,1616,1217,1049,349,1295,1069,360,1944,1862,36,833,125,1817,62↙
4,25,1902,267,781,456,962,388,1112,581,385,1902,1101,1753,363,540,329,199,920,512,692,↙
980,1629,1977,1619,969,333,1512,294,814,859,1578,871,1906,1742,990,697,354,1944,389,15↙
24,1945,1568,706,1736,121,1578,632,1561,942,523,1090,1294,1059,248,699,514,678,619,246↙
,301,1533,1238,154,459,1852,165];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'KroC100',
                    obj.dim=100;
                    obj.optima=20749;
                    x =↙
[1357,2650,1774,1307,3806,2687,43,3092,185,834,40,1183,2048,1097,1838,234,3314,737,779↙
,2312,2576,3078,2781,705,3409,323,1660,3729,693,2361,2433,554,913,3586,2636,1,482,3704↙
,3635,1362,2049,2552,3939,219,812,901,2513,242,826,3278,86,14,1327,2773,2469,3835,1031↙
,3853,1868,1544,457,3174,192,2318,2232,396,2365,2499,1410,2990,3646,3394,1779,1058,293↙
3,3099,2178,138,2082,2302,805,22,3213,99,1533,3564,29,3808,2221,3499,3124,781,1027,324↙
9,3297,213,721,3736,868,960];
                    y =↙
[1905,802,107,964,746,1353,1957,1668,1542,629,462,1391,1628,643,1732,1118,1881,1285,77↙
7,1949,189,1541,478,1812,1917,1714,1556,1188,1383,640,1538,1825,317,1909,727,457,1337,↙
1082,1174,1526,417,1909,640,898,351,1552,1572,584,1226,799,1065,454,1893,1286,1838,963↙
,428,1712,197,863,1607,1064,1004,1925,1374,828,1649,658,307,214,1018,1028,90,372,1459,↙
173,978,1610,1753,1127,272,1617,1085,536,1780,676,6,1375,291,1885,408,671,1041,378,491↙
,220,186,1542,731,303];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'KroD100',
                    obj.dim=100;
                    obj.optima=21294;
```

```
                    x =↙
[2995,202,981,1346,781,1009,2927,2982,555,464,3452,571,2656,1623,2067,1725,3600,1109,3↙
66,778,386,3918,3332,2597,811,241,2658,394,3786,264,2050,3538,1646,2993,547,3373,460,3↙
060,1828,1021,2347,3535,1529,1203,1787,2740,555,47,3935,3062,387,2901,931,1766,401,149↙
,2214,3805,1179,1017,2834,634,1819,1393,1768,3023,3248,1632,2223,3868,1541,2374,1962,3↙
007,3220,2356,1604,2028,2581,2221,2944,1082,997,2334,1264,1699,235,2592,3642,3599,1766↙
,240,1272,3503,80,1677,3766,3946,1994,278];
                    y =↙
[264,233,848,408,670,1001,1777,949,1121,1302,637,1982,128,1723,694,927,459,1196,339,12↙
82,1616,1217,1049,349,1295,1069,360,1944,1862,36,1833,125,1817,624,25,1902,267,781,456↙
,962,388,1112,581,385,1902,1101,1753,363,540,329,199,920,512,692,980,1629,1977,1619,96↙
9,333,1512,294,814,859,1578,871,1906,1742,990,697,354,1944,389,1524,1945,1568,706,1736↙
,121,1578,632,1561,942,523,1090,1294,1059,248,699,514,678,619,246,301,1533,1238,154,45↙
9,1852,165];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'KroE100',
                    obj.dim=100;
                    obj.optima=22068;
                    x =↙
[3477,91,3972,198,1806,538,3430,2186,1513,2143,53,3404,1034,2823,3104,3232,2790,374,74↙
1,3083,3502,1280,3326,217,2503,3527,739,3548,48,1419,1689,3468,1628,382,3029,3646,285,↙
1782,1067,2849,920,1741,876,2753,2609,3941,3613,1754,2916,2445,3825,2779,201,2502,765,↙
3105,1937,3364,3702,2164,3019,3098,3239,3359,2081,1398,618,1878,3803,397,3035,2502,323↙
0,3479,958,3423,78,96,3431,2053,3048,571,3393,2835,144,923,989,3061,2977,1668,878,678,↙
1086,640,3551,106,2243,3796,2643,48];
                    y =↙
[949,1732,329,1632,733,1023,1088,766,1646,1611,1657,1307,1344,376,1931,324,1457,9,146,↙
1938,1067,237,1846,38,1172,41,1850,1999,154,872,1223,1404,253,872,1242,1758,1029,93,37↙
1,1214,1835,712,220,283,1286,258,523,559,1724,1820,1101,435,693,1274,833,1823,1400,149↙
8,1624,1874,189,1594,1376,1693,1011,1100,1953,59,886,1217,152,146,380,1023,1670,1241,1↙
066,691,78,1461,1,1711,782,1472,1185,108,1997,1211,39,658,715,1599,868,110,1673,1267,1↙
332,1401,1320,267];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Lin318',
                    obj.dim=318;
                    obj.optima=42029;
                    x=↙
[63,94,142,173,205,213,244,276,283,362,394,449,480,512,528,583,591,638,638,638,638,669↙
,677,677,677,709,709,709,701,764,811,843,858,890,921,992,1000,1197,1228,1276,1299,1307↙
,1362,1362,1362,1425,1425,1425,1417,1488,1488,1488,1551,1551,1551,1614,1614,1614,1732,↙
1811,1843,1913,1921,2087,2118,2150,2189,2220,2220,2228,2244,2276,2276,2276,2315,2315,2↙
315,2331,2346,2346,2346,2362,2402,2402,2480,2496,2528,2559,2630,2638,2756,2787,2803,28↙
35,2866,2906,2937,2937,2945,3016,3055,3087,606,1165,1780,63,94,142,173,205,213,244,276↙
,283,362,394,449,480,512,528,583,591,638,638,638,638,669,677,677,677,709,709,709,701,7↙
64,811,843,858,890,921,992,1000,1197,1228,1276,1299,1307,1362,1362,1362,1425,1425,1425↙
,1417,1488,1488,1488,1551,1551,1551,1614,1614,1614,1732,1811,1843,1913,1921,2087,2118,↙
2150,2189,2220,2220,2228,2244,2276,2276,2276,2315,2315,2315,2331,2346,2346,2346,2362,2↙
402,2402,2480,2496,2528,2559,2630,2638,2756,2787,2803,2835,2866,2906,2937,2937,2945,30↙
16,3055,3087,606,1165,1780,63,94,142,173,205,213,244,276,283,362,394,449,480,512,528,5↙
```

```
83,591,638,638,638,638,669,677,677,677,709,709,709,701,764,811,843,858,890,921,992,100↙
0,1197,1228,1276,1299,1307,1362,1362,1362,1425,1425,1425,1417,1488,1488,1488,1551,1551↙
,1551,1614,1614,1614,1732,1811,1843,1913,1921,2087,2118,2150,2189,2220,2220,2228,2244,↙
2276,2276,2276,2315,2315,2315,2331,2346,2346,2346,2362,2402,2402,2480,2496,2528,2559,2↙
630,2638,2756,2787,2803,2835,2866,2906,2937,2937,2945,3016,3055,3087,606,1165,1780,141↙
7,1496,1693];
                    y=↙
[71,71,370,1276,1213,69,69,630,732,69,69,370,1276,1213,157,630,732,654,496,314,142,142↙
,315,496,654,654,496,315,142,220,189,173,370,1276,1213,630,732,1276,1213,205,630,732,6↙
54,496,291,654,496,291,173,291,496,654,654,496,291,291,496,654,189,1276,1213,630,732,3↙
70,1276,1213,205,189,630,732,142,315,496,654,654,496,315,142,315,496,654,142,157,220,1↙
42,370,1276,1213,630,732,69,69,370,1276,1213,69,69,630,732,1276,69,69,220,370,370,1402↙
,1402,1701,2607,2544,1400,1400,1961,2063,1400,1400,1701,2607,2544,1488,1961,2063,1985,↙
1827,1645,1473,1473,1646,1827,1985,1985,1827,1646,1473,1551,1520,1504,1701,2607,2544,1↙
961,2063,2607,2544,1536,1961,2063,1985,1827,1622,1985,1827,1622,1504,1622,1827,1985,19↙
85,1827,1622,1622,1827,1985,1520,2607,2544,1961,2063,1701,2607,2544,1536,1520,1961,206↙
3,1473,1646,1827,1985,1985,1827,1646,1473,1646,1827,1985,1473,1488,1551,1473,1701,2607↙
,2544,1961,2063,1400,1400,1701,2607,2544,1400,1400,1961,2063,2607,1400,1400,1551,1701,↙
1701,2733,2733,3032,3938,3875,2731,2731,3292,3394,2731,2731,3032,3938,3875,2819,3292,3↙
394,3316,3158,2976,2804,2804,2977,3158,3316,3316,3158,2977,2804,2882,2851,2835,3032,39↙
38,3875,3292,3394,3938,3875,2867,3292,3394,3316,3158,2953,3316,3158,2953,2835,2953,315↙
8,3316,3316,3158,2953,2953,3158,3316,2851,3938,3875,3292,3394,3032,3938,3875,2867,2851↙
,3292,3394,2804,2977,3158,3316,3316,3158,2977,2804,2977,3158,3316,2804,2819,2882,2804,↙
3032,3938,3875,3292,3394,2731,2731,3032,3938,3875,2731,2731,3292,3394,3938,2731,2731,2↙
882,3032,3032,-79,-79,4055];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Nrw1379',
                    obj.dim=1379;
                    obj.optima=56638;
                    x=↙
[2918,2925,2926,2927,2930,2934,2938,2941,2945,2947,2948,2948,2950,2950,2951,2959,2963,↙
2965,2965,2968,2969,2972,2973,2977,2981,2982,2983,2987,2995,3001,3002,3004,3005,3007,3↙
009,3009,3010,3010,3014,3018,3019,3020,3021,3024,3025,3026,3026,3026,3029,3029,3033,30↙
33,3035,3036,3036,3038,3038,3042,3045,3045,3046,3046,3048,3048,3048,3049,3051,3052,305↙
2,3053,3055,3056,3057,3057,3058,3062,3063,3064,3065,3068,3068,3068,3072,3076,3078,3079↙
,3080,3082,3083,3086,3087,3087,3088,3088,3090,3093,3093,3093,3093,3095,3096,3097,3097,↙
3099,3099,3101,3102,3103,3104,3107,3110,3113,3115,3116,3116,3117,3118,3122,3124,3124,3↙
125,3128,3130,3135,3139,3140,3140,3141,3143,3143,3144,3145,3145,3145,3146,3147,3147,31↙
48,3148,3148,3152,3155,3157,3159,3159,3161,3163,3165,3167,3170,3170,3171,3175,3175,317↙
6,3177,3179,3179,3182,3182,3182,3183,3183,3186,3186,3188,3188,3188,3188,3189,3194,3194↙
,3195,3196,3201,3203,3203,3203,3205,3205,3206,3208,3208,3211,3212,3214,3214,3215,3216,↙
3217,3218,3218,3219,3220,3220,3222,3224,3225,3225,3227,3228,3228,3230,3230,3231,3234,3↙
240,3241,3241,3244,3245,3245,3247,3247,3247,3251,3252,3253,3253,3253,3254,3255,3257,32↙
58,3259,3261,3262,3264,3266,3267,3268,3269,3270,3270,3272,3275,3276,3278,3281,3282,328↙
4,3285,3288,3288,3289,3289,3292,3294,3295,3296,3296,3296,3296,3296,3301,3302,3302,3302↙
,3303,3304,3305,3306,3307,3310,3310,3310,3313,3314,3316,3319,3320,3322,3322,3323,3324,↙
3325,3328,3329,3329,3329,3333,3334,3335,3336,3336,3336,3336,3337,3337,3339,3339,3340,3↙
340,3340,3341,3341,3341,3341,3345,3351,3351,3353,3353,3355,3358,3359,3362,3364,3364,33↙
66,3366,3368,3369,3369,3369,3370,3371,3375,3376,3377,3378,3378,3379,3379,3381,3381,338↙
1,3382,3382,3382,3382,3386,3388,3390,3391,3392,3392,3394,3394,3395,3398,3398,3398,3400↙
,3404,3404,3404,3405,3406,3406,3406,3406,3409,3410,3411,3411,3412,3413,3413,3415,3417,↙
3418,3418,3419,3421,3422,3423,3425,3427,3427,3427,3428,3429,3432,3433,3433,3435,3437,3↙
```

```
439,3441,3443,3443,3444,3446,3446,3449,3450,3452,3452,3452,3453,3454,3454,3455,3456,34↙
56,3457,3458,3458,3458,3459,3460,3461,3464,3464,3465,3466,3466,3470,3471,3471,3472,347↙
3,3475,3477,3478,3482,3482,3485,3485,3486,3487,3489,3489,3490,3490,3492,3495,3496,3500↙
,3501,3502,3503,3503,3503,3503,3508,3508,3509,3510,3510,3510,3512,3512,3515,3516,3516,↙
3517,3519,3520,3522,3524,3524,3526,3526,3526,3526,3527,3527,3528,3528,3530,3530,3531,3↙
532,3534,3536,3537,3541,3542,3542,3543,3544,3545,3550,3552,3552,3553,3557,3557,3557,35↙
58,3559,3563,3563,3564,3565,3565,3565,3568,3568,3569,3570,3570,3570,3571,3572,3574,357↙
5,3575,3576,3577,3577,3578,3578,3578,3581,3582,3586,3587,3587,3589,3592,3593,3594,3596↙
,3598,3601,3603,3604,3606,3606,3607,3609,3610,3610,3611,3611,3612,3615,3617,3619,3620,↙
3622,3623,3624,3625,3630,3630,3630,3630,3639,3639,3639,3639,3642,3646,3652,3652,3653,3↙
653,3653,3655,3655,3657,3657,3660,3660,3660,3661,3666,3669,3674,3675,3678,3680,3681,36↙
81,3681,3682,3682,3683,3683,3684,3685,3688,3688,3692,3699,3699,3702,3702,3703,3703,370↙
5,3706,3706,3709,3710,3711,3711,3713,3716,3717,3717,3721,3722,3722,3723,3726,3731,3732↙
,3732,3732,3734,3734,3736,3739,3742,3742,3743,3744,3745,3745,3748,3749,3750,3753,3757,↙
3758,3760,3762,3763,3768,3771,3773,3773,3774,3776,3777,3777,3777,3778,3779,3779,3781,3↙
782,3783,3785,3786,3789,3789,3792,3793,3794,3797,3800,3800,3801,3803,3812,3814,3814,38↙
15,3817,3819,3819,3819,3821,3826,3827,3827,3828,3830,3830,3833,3836,3838,3838,3840,384↙
2,3844,3845,3845,3847,3848,3852,3853,3857,3860,3861,3862,3867,3867,3868,3868,3869,3872↙
,3874,3875,3879,3882,3882,3883,3883,3883,3884,3887,3890,3891,3892,3892,3896,3896,3897,↙
3899,3900,3901,3904,3905,3905,3906,3907,3908,3909,3909,3909,3910,3912,3914,3917,3918,3↙
922,3922,3922,3931,3932,3934,3935,3935,3937,3938,3941,3943,3944,3945,3945,3946,3949,39↙
51,3951,3952,3952,3955,3959,3962,3962,3968,3969,3970,3970,3971,3973,3973,3975,3977,397↙
7,3981,3982,3983,3983,3983,3984,3985,3987,3988,3991,3991,3993,3994,3995,3995,3995,3999↙
,4,4004,4004,4004,4005,4006,4007,4012,4014,4014,4023,4027,4030,4032,4032,4032,4034,403↙
8,4038,4041,4043,4045,4045,4046,4046,4047,4050,4051,4051,4052,4053,4054,4054,4055,4055↙
,4057,4059,4062,4064,4064,4065,4067,4068,4068,4068,4073,4073,4079,4084,4085,4089,4092,↙
4094,4095,4095,4097,4099,4100,4103,4110,4118,4119,4122,4122,4122,4123,4123,4126,4127,4↙
127,4128,4129,4130,4130,4130,4131,4133,4136,4137,4139,4141,4142,4142,4143,4143,4144,41↙
46,4149,4163,4164,4167,4167,4168,4170,4171,4173,4176,4177,4177,4180,4183,4189,4189,419↙
0,4190,4192,4193,4199,4203,4204,4204,4204,4210,4210,4210,4211,4213,4214,4214,4214,4217↙
,4218,4220,4223,4224,4233,4234,4235,4237,4241,4245,4246,4248,4249,4250,4253,4254,4257,↙
4260,4263,4265,4266,4266,4271,4275,4276,4276,4278,4279,4281,4282,4283,4283,4284,4285,4↙
287,4292,4292,4293,4294,4297,4298,4300,4300,4303,4306,4307,4308,4309,4312,4314,4314,43↙
22,4324,4328,4329,4331,4336,4337,4337,4337,4338,4339,4340,4341,4343,4348,4348,4348,434↙
9,4351,4351,4357,4360,4360,4361,4363,4365,4366,4366,4366,4368,4370,4370,4371,4372,4375↙
,4375,4376,4382,4384,4388,4391,4393,4393,4395,4396,4396,4396,4398,4398,4398,4400,4404,↙
4405,4408,4410,4413,4414,4416,4417,4418,4418,4419,4421,4421,4423,4424,4430,4430,4433,4↙
443,4444,4448,4452,4454,4454,4461,4466,4469,4470,4471,4472,4477,4478,4478,4478,4480,44↙
80,4480,4481,4481,4484,4488,4493,4497,4498,4500,4501,4501,4502,4506,4507,4508,4511,451↙
4,4517,4519,4521,4525,4526,4528,4533,4535,4535,4538,4539,4542,4542,4543,4545,4546,4549↙
,4551,4551,4555,4558,4561,4561,4563,4566,4567,4568,4569,4572,4574,4575,4576,4577,4577,↙
4579,4581,4582,4584,4584,4585,4587,4588,4590,4591,4594,4597,4598,4598,4601,4602,4602,4↙
606,4606,4608,4608,4610,4611,4613,4615,4617,4623,4625,4626,4626,4628,4629,4629,4630,46↙
30,4631,4632,4632,4633,4634,4638,4639,4641,4644,4645,4645,4647,4649,4649,4651,4654,465↙
5,4657,4657,4658,4659,4660,4661,4662,4663,4664,4664,4668,4669,4671,4672,4674,4675,4678↙
,4680,4683,4686,4693,4695,4696,4697,4699,4699,4700,4700,4700,4701,4705,4710,4713,4713,↙
4714,4715,4715,4715,4718,4718,4725,4727,4729,4730,4730,4735,4736,4736,4736,4738,4738,4↙
739,4740,4741,4743,4743,4744,4748,4748,4749,4751,4751,4753,4756,4760,4761,4763,4766,47↙
66,4767,4768,4768,4770,4771,4772,4773,4774,4776,4776,4778,4778,4780,4784,4791,4792,479↙
4,4795,4795,4796,4800,4801,4803,4805,4809,4812,4812,4813,4817,4820,4822,4823,4827,4829↙
,4833,4836,4838,4839,4839,4840,4846,4849,4852,4859,4861,4863,4863,4865,4867,4868,4869,↙
4875,4876,4881,4883,4883,4885,4885,4886,4887,4887,4891,4892,4895,4897,4898,4899,4899,4↙
904,4906,4909,4910,4912,4916,4917,4918,4919,4920,4920,4930,4933,4939,4943,4944,4945,49↙
47,4954,4954,4955,4956,4963,4964,4968,4971,4977,4979,4980,4982,4985,4988,4988,4990,499↙
2,4996,4996,4997,5,5,5005,5009,5013,5013,5013,5019,5021,5024,5025,5025,5025,5029,5036,↙
```

```
5038,5040,5041,5045,5048,5057,5061,5064,5065,5065,5069,5070,5071,5077,5077,5080,5080,5↙
085,5086,5088,5090,5093,5103,5103,5116,5117,5126,5131,5136,5141,5149,5155,5156,5158,51↙
62,5167,5169,5175,5190,5193,5193,5195,5202,5206,5206,5211,5212,5219,5239,5250,5259,526↙
3,5294];
                    y=↙
[6528,6597,6609,6312,6328,6545,7412,6456,6284,6663,6513,7456,6350,6480,7293,6385,6476,↙
6450,6602,7322,6212,6406,6671,6493,6576,6333,7487,6551,7425,6577,6598,6394,6345,7303,6↙
209,6657,6230,6643,6478,6382,7435,6429,6137,6768,6547,6604,7407,7421,5998,6048,6180,66↙
39,6801,6232,6669,7183,7291,7212,6035,6846,6527,7502,6270,6488,6886,6691,6572,6596,675↙
3,6639,7323,7378,6012,6351,6070,7227,6054,6217,6534,6450,6788,7146,6121,6326,6724,6259↙
,6857,6348,6505,7082,6768,7046,6091,7339,6962,6059,6374,6414,7177,6567,6686,6766,6984,↙
6275,6833,7028,6459,7463,6870,6900,6091,6336,7272,6445,6489,7239,7194,6617,6563,7075,7↙
161,6113,7349,6650,6528,6372,6807,6324,6440,7104,6595,5877,6750,7447,6063,6176,6981,60↙
97,6902,7403,6204,6692,5987,5921,6482,6627,6220,6020,6844,6146,7025,6372,6440,7163,703↙
3,6887,6246,7078,6331,6578,6810,5959,7202,6354,6469,6184,6672,7242,7304,6617,5840,6531↙
,6031,7354,6507,6221,6930,7370,6412,7014,7425,6743,6860,5999,6360,6564,7119,6709,6115,↙
7074,6482,6673,6636,6396,6868,6992,6262,6180,7283,6951,6241,7468,6050,6303,6833,6449,7↙
257,5956,6806,6696,6479,7040,6222,6594,7384,6670,6879,7013,7334,7451,6637,5840,6251,65↙
03,6332,6101,6982,6735,5874,6133,6938,7278,6012,6558,5994,7480,6459,7163,6521,6792,594↙
0,7434,6170,6294,6203,6817,6452,6654,6737,6114,6342,6512,6602,7079,6897,6138,6364,6572↙
,7012,6231,6263,6706,7184,6391,6490,6622,6600,5898,6102,6878,6859,5956,6189,7155,7129,↙
7223,6735,5833,5896,6986,7057,6638,6253,6088,6124,6360,6662,7128,7335,6065,6756,5974,6↙
424,6908,6177,6305,6820,7490,6469,6889,7198,7048,7376,7255,7024,7454,5997,6412,6913,61↙
74,6501,6993,5819,6698,6835,6111,6489,6450,7219,6964,6093,6531,5933,6809,6372,6556,674↙
0,6232,6654,6707,6860,6015,6951,7505,7068,5909,6420,6258,7102,6795,6296,7178,7327,6586↙
,6166,6633,6950,7046,5820,6080,6194,7166,7117,6776,6130,6336,6509,6691,7017,7453,6022,↙
6364,6405,6479,6973,6535,6053,6875,6294,6853,7021,5978,7061,6245,6448,7140,6929,6139,7↙
260,7183,6472,6643,6338,6193,6601,7080,7104,6766,7003,7060,6103,6704,7220,6882,6905,69↙
23,6936,6193,6514,7386,6310,6417,6163,6272,6375,7099,6125,6952,6075,6541,7523,6772,622↙
5,6479,6245,6451,6101,6393,6621,7114,6710,5976,6356,6888,6153,6750,7720,5927,6650,6756↙
,6372,6488,6124,6588,6845,7129,7090,7667,6699,6223,7023,7370,7123,7275,6172,6910,6977,↙
7048,7459,7107,6645,6325,6618,6171,6992,7347,7525,6200,6532,6149,6288,6557,7286,6470,6↙
513,7566,6267,6460,6802,6424,6882,6358,6762,6584,6097,6845,7154,6250,6279,6325,7180,72↙
06,6703,6473,7097,6757,6448,6573,6631,6542,6921,6363,6401,7349,7502,7619,7699,7773,615↙
9,6439,6344,7041,7434,6076,6121,6317,6249,6608,6513,6566,6877,6793,6663,6544,7253,6219↙
,7029,6970,6093,6691,7698,7716,7151,6729,6945,7221,6631,6764,6582,7508,6305,7357,7021,↙
6258,6597,6192,6280,6823,7103,7311,7572,6152,6261,6889,7712,6330,7509,7207,7853,6765,7↙
404,7436,6108,6955,6213,6761,6105,7566,7825,6701,6945,6855,7592,7133,7181,6638,6287,65↙
46,6603,6093,7121,6222,7305,6578,7245,6499,7442,6903,6157,7769,6303,7657,6467,7754,666↙
7,6193,6625,7044,7167,6762,7345,6222,6117,6422,7631,6252,6128,7118,6391,6152,6715,6353↙
,6497,7375,6286,6609,6948,7546,6329,6520,6208,6811,6555,6762,7333,6115,6664,7194,6174,↙
7507,7450,7041,7856,7110,6742,7368,7642,6440,7638,6153,6905,7783,6476,6340,7308,6920,6↙
630,6723,6568,6185,6258,7229,6322,6432,7726,7188,6781,6879,6132,6974,7509,6484,7430,68↙
28,7647,6529,7209,7327,6194,7594,6338,6754,7117,6606,7541,7286,7747,6480,6673,6864,770↙
8,6439,6241,6996,6206,7062,7851,6732,7232,7784,6838,6526,7124,7781,7089,7541,6558,6801↙
,7135,7678,7469,6399,7294,6327,6636,6936,6491,7409,7889,6257,7196,7354,7037,7260,7098,↙
6554,7753,7435,6206,6514,7076,6376,7857,6247,6717,6948,7544,7590,6441,6975,6611,7502,6↙
497,7166,7358,6906,6792,6571,6329,7075,7926,7745,7243,6415,6624,6548,7612,7279,7666,67↙
10,7782,6975,7522,6234,7453,6280,7708,7868,6486,6773,6822,6594,6395,6997,7136,6841,697↙
0,7202,7190,7111,7233,7322,7985,6443,6246,7533,6265,7882,7416,7087,6531,7564,7808,7292↙
,7342,6894,6964,6997,6480,7146,6288,7753,6652,7823,7175,6364,7126,6560,7316,7917,6912,↙
7486,7945,6270,7096,6422,6743,6620,6797,7717,8038,7057,7381,7661,6527,6409,7240,6864,7↙
044,7573,7190,7903,6994,7161,7124,6319,7508,7806,8111,6832,7220,7325,6489,6352,6450,70↙
98,7442,6921,7524,7278,7251,6450,6966,7601,6553,7150,6868,7390,7924,6700,7222,6430,663↙
5,7522,7349,8025,6489,7212,7469,6836,6563,6764,7763,6790,7276,6528,7646,7167,7862,6931↙
```

```
,↙
7030,6815,6989,7013,6922,7594,7065,7348,7964,7905,6816,6457,7111,7250,7517,6400,6963,7↙
288,7857,6529,7437,6805,6997,6698,7221,7650,6464,7199,7532,6866,6389,6736,7048,7147,69↙
02,6845,7736,7817,6927,7048,6638,7347,7959,7015,6506,6964,7468,7114,6425,7406,6907,638↙
7,7916,7167,6583,6627,6832,7268,6788,7059,7423,7195,7610,6979,6686,6502,6989,7039,6376↙
,6315,6259,7524,6825,6917,7773,6460,6546,6764,6402,6710,6866,7558,7108,6645,6359,6604,↙
6971,6307,6822,7388,7430,6713,6401,7238,7665,7076,7337,6856,7482,7597,6326,6469,6580,6↙
942,6214,6613,6445,6534,6652,7371,6594,6810,6404,6867,7461,6402,6291,7071,6977,6751,64↙
94,7229,7203,7134,6229,6692,6189,6834,7533,7052,6435,6866,7630,6322,6369,7052,6291,694↙
3,6541,7239,7417,6718,7586,6680,7353,6628,6792,7651,6591,7474,6914,7151,6829,7318,7033↙
,6389,7500,6334,7389,6868,7690,7222,6740,7236,6789,6423,7563,7177,6483,7114,7476,7029,↙
7158,7635,6681,7315,6646,6512,7291,6973,7703,6665,6884,7412,6572,7554,6552,6491,7619,6↙
832,6930,6707,7721,7436,7069,6430,7359,7032,7118,6901,7645,6455,7177,6534,6355,6398,72↙
43,6988,7599,8101,6772,7198,7673,6673,7130,7387,6527,7266,7504,6553,6488,7710,6770,764↙
9,7338,7203,6897,6838,8095,7186,6512,6585,6416,7078,7549,8040,7614,7690,6907,6999,7125↙
,7042,7394,7263,7290,6440,7630,7451,6802,7420,6479,7716,7077,8069,6707,7830,8024,6911,↙
7574,7327,6522,6775,7207,7654,8095,6747,6678,6844,6893,7550,7883,6493,7401,7762,6816,7↙
981,6925,7947,7846,6599,7601,7475,7205,7014,7798,8123,7139,7717,7612,6783,6845,6711,75↙
55,7629,8040,7928,7382,7109,6665,7297,7344,7760,7253,7446,7980,6934,6877,7049,7549,813↙
0,8093,7283,7828,7857,7143,7824,6742,7241,7642,7598,7684,7735,6617,7,7494,7791,7994,80↙
87,6659,7766,7314,7947,7084,7591,8027,8174,6913,7920,6958,7869,7169,6760,6889,7407,769↙
4,8112,7209,7338,6683,7586,7938,6930,7775,7640,7910,7735,8065,8002,7311,7259,7073,7620↙
,6713,7827,7018,7873,7572,7911,7327,6974,7126,7233,6920,7673,7942,7067,7507,7980,7701,↙
7293,7192,7813,7635,7685,7129,6963,7759,7886,7719,7926,7157,7599,7808,7662,7835,7551,7↙
956,7366,6947,7025,7755,7530,7863,7634,7187,7236,7124,7393,7585,7267,7321,7005,8,7788,↙
7907,7066,7698,7847,7802,7536,7492,7958,7661,7743,7178,7636,7887,7929,7985,7869,7528,7↙
673,7864,7288,7220,7343,7318,7724,7474,7767,8026,7814,8073,7143,7490,7882,7364,7407,77↙
41,8017,7983,7246,7592,7643,7698,7313,7574,7828,8067,7083,7434,7190,7515,8085,7867,742↙
1,7114,7241,7344,7623,8141,7723,7063,7459,7284,7543,6992,7490,8133,7714,7744,7382,7595↙
,7074,7641,7681,7158,7186,7037,7231,7131,7614,7292,7473,7469,7094,7419,7064,7259,7404,↙
7172,7134,7560,7503,7226,7432,7473,7089,7192,7285,7358,7329,7134,7419,7285,7312,7234,7↙
358,7376];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Pcb442',
                    obj.dim=442;
                    obj.optima=50778;
                    x=↙
[200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,200,2↙
00,200,200,200,200,200,200,200,200,200,200,200,300,300,300,300,300,300,300,300,300,300↙
,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,3↙
00,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400↙
,400,400,400,400,400,400,400,400,400,400,400,400,500,500,500,600,700,700,700,700,700,7↙
00,700,700,700,700,700,800,800,800,800,800,800,800,800,800,800,800,800,900,900,900,900↙
,900,900,900,900,900,900,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1↙
000,1100,1100,1100,1100,1100,1100,1100,1100,1100,1100,1100,1100,1200,1200,1200,1200,12↙
00,1200,1200,1200,1200,1200,1200,1300,1300,1300,1300,1300,1300,1300,1300,1300,1300,130↙
0,1300,1300,1400,1400,1400,1400,1400,1400,1400,1400,1400,1400,1400,1400,1400,1400,1400↙
,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1600,1600,1600,1600,1600,1600,↙
1600,1600,1600,1600,1700,1700,1700,1700,1700,1700,1800,1800,1800,1800,1800,1800,1800,1↙
900,1900,1900,1900,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,20↙
00,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2100,2100,210↙
0,2200,2200,2200,2200,2300,2300,2300,2400,2400,2400,2500,2500,2600,2600,2600,2600,2600↙
```

```
,↙
2600,2600,2600,2600,2600,2600,2600,2600,2600,2600,2600,2600,2600,2600,2600,2600,2↙
600,2600,2600,2600,2600,2700,2700,2700,2700,2700,2700,2700,2700,2700,2700,2700,27↙
00,2700,2700,2700,2700,2700,2700,2700,2700,2700,2700,2700,2700,2700,2700,2700,270↙
0,2700,2800,2800,2900,2900,2900,2900,2900,3000,3000,3000,3000,3000,3000,3000,3000,3000↙
,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,3000,↙
3000,3000,3000,3000,150,150,469,469,469,540,540,620,620,750,850,850,850,939,950,910,10↙
50,1150,1170,1220,1350,1350,1350,1450,1550,1550,1550,1650,1690,1710,1710,1750,1790,172↙
0,1790,1720,1829,1829,1829,2060,2050,2170,2110,2120,2150,2290,2220,2280,2390,2320,2450↙
,2620,2750,2760,2850,2850,2850,2930,2950,2950,520,2300,2320,530,2550,750,0];
                    y=↙
[400,500,600,700,800,900,1000,1100,1200,1300,1400,1500,1600,1700,1800,1900,2000,2100,2↙
200,2300,2400,2500,2600,2700,2800,2900,3000,3100,3200,3300,3400,3500,3600,400,500,600,↙
700,800,900,1000,1100,1200,1300,1400,1500,1600,1700,1800,1900,2000,2100,2200,2300,2400↙
,2500,2600,2700,2800,2900,3000,3100,3200,3300,3400,3500,400,500,600,700,800,900,1000,1↙
100,1200,1300,1400,1500,1600,1700,1800,1900,2000,2100,2200,2300,2400,2500,2600,2700,28↙
00,2900,3000,3100,3200,3300,3400,3500,3600,1500,1829,3100,400,300,600,1500,1600,1800,2↙
100,2400,2700,3000,3300,3600,300,600,1030,1500,1800,2100,2400,2600,2700,3000,3300,3600↙
,300,600,1500,1800,2100,2400,2700,3000,3300,3600,300,600,1100,1500,1629,1800,2100,2400↙
,2600,2700,3000,3300,3600,300,600,700,900,1500,1800,2100,2400,2700,3000,3300,3600,300,↙
600,1500,1700,1800,2100,2400,2700,3000,3300,3600,300,600,700,1130,1500,1800,2100,2200,↙
2400,2700,3000,3300,3600,300,600,930,1500,1800,2000,2100,2400,2500,2700,2820,2900,3000↙
,3300,3600,1500,1800,1900,2100,2400,2700,2800,2860,3000,3300,3600,1100,1300,1500,1800,↙
2100,2400,2700,3000,3300,3600,1200,1500,1800,2100,2400,3600,300,600,1230,1500,1800,210↙
0,2400,300,600,3000,3520,300,370,600,800,900,1000,1100,1200,1300,1400,1500,1600,1700,1↙
800,1900,2000,2100,2200,2300,2400,2500,2600,2700,2800,2900,3000,3100,3500,300,600,3200↙
,300,469,600,3200,300,600,3400,300,600,2100,300,800,400,500,800,900,1000,1100,1200,130↙
0,1400,1500,1600,1700,1800,1900,2000,2100,2200,2300,2400,2500,2600,2700,2800,2900,3000↙
,3100,3400,700,800,900,1000,1100,1200,1300,1400,1500,1600,1700,1800,1900,2000,2100,220↙
0,2300,2500,2600,2700,2800,2900,3000,3100,3200,3300,3400,3500,3600,3700,3800,900,1130,↙
400,500,1400,2400,3000,700,800,900,1000,1100,1200,1300,1500,1600,1700,1800,1900,2000,2↙
100,2200,2300,2500,2600,2700,2800,2900,3000,3100,3200,3300,3400,3500,3600,3700,3800,35↙
00,3550,2550,3350,3450,2330,2430,3650,3709,2550,520,700,2280,740,2220,2600,1050,1350,2↙
280,2210,750,1700,2140,770,300,500,1850,1050,2680,310,510,750,2580,2610,3330,3409,2700↙
,2800,3450,1650,3150,1900,2000,2750,3250,1400,2820,3250,1300,1500,710,3650,520,2360,22↙
00,2700,3350,950,1750,2050,3200,3500,3150,2100,710,490,0];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Pcb1173',
                    obj.dim=1173;
                    obj.optima=56892;
                    x=↙
[2017,2017,2018,2017,2016,2019,2019,2019,2031,1963,1964,1941,1862,1863,1865,1865,1865,↙
1866,1864,1864,1972,1971,1972,1974,1973,1974,1974,1943,1915,1856,1856,1858,1857,1857,1↙
858,1857,1820,2034,1860,1781,1976,1976,1975,1978,1978,1977,1977,1976,1979,1979,1978,20↙
26,2026,2025,2036,1931,1901,1861,1859,1861,1859,1860,1861,1858,1861,1860,1865,1860,197↙
9,1979,1979,1978,1981,1980,1980,1982,1981,1983,1984,2031,1952,1922,1933,1923,1862,1864↙
,1864,1865,1865,1864,1866,1865,1866,1865,1868,1982,1983,1985,1985,1984,1986,1984,1988,↙
1937,1906,1867,1867,1869,1869,1870,1867,1868,1871,1870,1958,2034,2035,2037,2038,1988,1↙
989,1988,1990,1989,1989,1990,1990,1993,1921,1904,1870,1872,1872,1871,1874,1871,1873,18↙
73,1873,1874,1758,1709,567,570,511,462,248,1563,1524,1486,1447,1409,1370,1332,1293,125↙
7,1216,1180,1138,1101,1061,1024,985,946,908,874,832,947,908,872,789,753,716,677,638,15↙
64,1526,1489,1449,1409,1373,1334,1296,1258,1216,1178,1139,1096,1060,1023,986,947,910,8↙
```

```
75,833,792,751,714,678,638,1721,1681,1641,1602,1565,1527,1489,1450,1025,983,1025,638,5↙
98,561,521,1627,1625,1627,1627,1622,1627,1625,1626,1626,1606,1645,1683,1703,1720,1739,↙
1759,1743,1742,1740,1741,1745,1744,1743,1743,1513,1580,1581,1540,1511,1512,1511,1558,1↙
512,1510,1510,1508,1474,1471,1450,1434,1491,1528,1568,1395,1395,1396,1394,1395,1396,14↙
25,1396,1397,1398,1398,1340,1329,1280,1282,1282,1278,1281,1280,1280,1281,1281,1257,124↙
2,1165,1164,1163,1165,1166,1166,1165,1165,1167,1088,1107,1108,1087,1049,1049,1049,1051↙
,1048,1050,1052,1050,1050,1019,1021,1019,1014,974,934,935,936,935,936,936,936,936,935,↙
896,898,899,900,860,859,861,860,859,883,817,816,817,819,816,815,816,817,788,768,777,73↙
0,695,702,703,699,700,701,699,701,702,702,642,602,564,525,542,640,583,585,585,641,585,↙
584,586,645,623,586,585,540,473,406,414,354,354,354,354,353,354,354,354,358,326,326,32↙
6,327,315,274,313,316,295,296,234,237,238,238,235,237,238,239,241,198,201,200,170,166,↙
167,166,166,1816,1804,1806,1746,1747,1748,1747,1746,1747,1749,1743,1751,1744,1749,1748↙
,1627,1633,1630,1628,1631,1634,1632,1633,1632,1629,1632,1631,1653,1633,1580,1592,1514,↙
1514,1514,1509,1514,1513,1513,1516,1515,1516,1515,1527,1475,1436,1400,1397,1399,1399,1↙
396,1398,1401,1398,1398,1399,1400,1401,1402,1403,1360,1361,1340,1358,1358,1331,1282,12↙
81,1284,1283,1282,1282,1283,1283,1285,1285,1285,1253,1218,1215,1166,1166,1166,1165,116↙
7,1167,1167,1168,1169,1168,1172,1168,1127,1131,1092,1052,1050,1047,1049,1051,1051,1050↙
,1052,1052,1051,1052,995,967,933,935,934,934,935,935,936,935,936,937,939,880,815,813,8↙
18,816,817,819,822,849,823,771,739,703,703,703,702,701,702,702,705,704,674,643,626,646↙
,588,586,586,588,588,587,589,610,591,547,548,510,473,471,473,472,469,470,473,473,475,4↙
43,383,326,356,355,356,352,356,357,357,237,240,240,237,241,241,240,241,472,478,427,397↙
,354,326,287,238,238,240,240,1790,1792,1790,1790,1748,1750,1751,1749,1752,1753,1751,17↙
52,1750,1752,1752,1632,1634,1634,1635,1633,1637,1636,1634,1636,1635,1635,1588,1851,183↙
2,1793,1793,1752,1753,1751,1753,1754,1754,1757,1756,1757,1756,1756,1727,1706,1702,1723↙
,1638,1637,1636,1636,1638,1637,1639,1639,1638,1639,1640,1619,1652,1640,1833,1812,1826,↙
1825,1795,1816,1777,1757,1758,1755,1755,1754,1757,1752,1755,1726,1709,1687,1608,1638,1↙
640,1640,1639,1643,1642,1641,1641,1730,1730,1681,1613,1585,1587,1517,1517,1514,1516,15↙
16,1517,1515,1516,1520,1518,1519,1520,1489,1459,1460,1440,1432,1400,1399,1401,1403,140↙
3,1404,1403,1403,1403,1404,1517,1518,1519,1519,1518,1519,1519,1520,1521,1523,1521,1492↙
,1473,1451,1454,1403,1402,1405,1403,1403,1404,1403,1405,1405,1406,1405,1591,1593,1419,↙
1554,1514,1477,1434,1402,1552,1515,1476,1439,1398,1556,1515,1476,1440,1396,1555,1514,1↙
476,1435,1399,1352,1364,1365,1364,1345,1346,1286,1323,1284,1284,1286,1286,1284,1286,12↙
90,1259,1369,1327,1368,1351,1232,1259,1291,1288,1289,1287,1290,1289,1290,1291,1321,123↙
5,1272,1320,1293,1295,1292,1296,1266,1295,1331,1362,1293,1295,1294,1345,1297,1248,1210↙
,1167,1169,1171,1171,1171,1172,1170,1172,1174,1132,1104,1150,1116,1118,1105,1104,1064,↙
1025,986,947,911,1005,1062,1026,988,948,913,1063,1028,992,947,911,1065,1029,992,952,91↙
8,1045,941,820,819,819,820,820,821,823,822,781,733,705,706,703,705,705,708,708,708,593↙
,592,589,591,591,591,593,516,504,475,473,473,475,474,475,475,446,416,357,357,360,359,3↙
60,358,359,242,242,241,245,244,241,243,1171,1170,1172,1171,1173,1171,1172,1176,1172,11↙
75,1173,1177,1177,1177,1178,1178,1177,1179,1178,1178,1145,1113,1114,1113,1114,1116,111↙
4,1117,1112,1118,1117,1087,1099,1101,1102,1057,1057,1057,1057,1056,1059,1059,1061,1060↙
,1059,1062,1061,1062,1062,1063,1062,1066,1064,1019,982,972,1002,974,1005,941,939,942,9↙
42,943,943,942,942,943,945,946,946,945,948,947,948,948,948,950,914,938,931,910,912,880↙
,891,869,842,821,618,628,821,783,743,708,668,823,784,747,707,669,630,824,783,747,707,6↙
70,823,786,748,707,671,727,755,788,766,591,592,592,593,593,594,596,595,598,554,537,477↙
,478,478,479,478,480,481,480,481,440,440,441,399,400,404,399,358,363,362,362,361,364,3↙
63,363,273,307,279,245,246,246,245,246,247,247,248,247,247,277,230,805,748,709,826,827↙
,826,828,829,829,829,711,712,715,710,712,713,715,714,714,672,647,685,273,771,831,760,6↙
63,664,596,627,629,594,597,595,599,598,598,598,599,509,561,561,541,521,571,599,558,532↙
,485,522,345,403,443,460,489,493,440,479,481,481,483,479,481,484,485,470,443,443,454,4↙
45,402,402,402,403,342,365,364,365,365,365,366,365,367,367,368,311,246,248,247,247,250↙
,252,253,251,250,253,213,221,263,253,215];
                    y=↙
[663,703,741,779,817,856,896,935,966,877,901,934,666,705,742,780,820,859,895,938,1051,↙
1090,1128,1168,1205,1245,1284,1110,1150,1053,1090,1130,1169,1207,1245,1286,1246,1360,1↙
```

392,1390,1456,1494,1536,1573,1611,1651,1688,1728,1767,1806,1852,1573,1648,1729,1854,17↙
75,1887,1456,1498,1537,1576,1614,1651,1691,1731,1768,2287,1853,1978,2015,2055,2097,213↙
3,2092,2210,2248,2284,2326,2377,2376,2058,2093,2209,2250,1930,1979,2017,2056,2093,2136↙
,2174,2214,2250,2291,2329,2459,2579,2541,2576,2616,2653,2671,2734,2696,2616,2414,2462,↙
2502,2544,2578,2619,2657,2697,2736,2830,2857,2906,3218,3361,2985,3020,3061,3099,3139,3↙
178,3214,3257,3365,2983,3103,2859,2938,2987,3027,3066,3104,3138,3180,3219,3258,540,559↙
,535,578,558,558,561,707,707,709,710,708,709,709,709,707,709,710,709,712,710,710,711,7↙
10,709,1111,1111,713,711,1110,627,712,713,712,710,745,746,745,745,746,745,746,745,747,↙
748,744,748,751,747,748,748,751,747,1146,1147,1148,752,750,750,751,782,782,784,784,785↙
,784,786,784,803,807,845,788,792,790,790,1323,1282,1243,1205,685,1129,1088,1051,984,93↙
5,934,933,985,934,891,934,1052,1090,1129,1164,1206,1247,1284,1322,1373,1373,1323,1284,↙
1285,1246,1206,1204,1169,1127,1091,1052,1043,983,935,906,936,935,933,987,1055,1092,112↙
7,1168,1207,1228,1246,1285,1339,1458,1295,1169,1372,1322,1286,1243,1209,1170,1132,1092↙
,1054,958,984,988,1057,1095,1132,1170,1208,1246,1289,1327,1049,1151,1286,1329,1058,109↙
3,1134,1172,1207,1249,1289,1328,1378,1323,1269,1226,988,1172,989,1058,1097,1134,1175,1↙
210,1250,1287,1327,1466,1043,1284,1326,1327,1371,1533,1683,1325,1876,1458,1496,1539,15↙
68,1117,1648,1684,1807,1357,1195,990,1139,913,991,1058,1095,1137,1176,1215,1252,1291,1↙
325,943,945,945,943,993,1059,1058,1098,1138,1135,1176,1215,1254,1250,1290,1289,1340,13↙
75,1342,1255,1079,1056,1100,1141,1179,1216,1260,1292,1345,1383,792,1312,870,909,947,94↙
6,995,1051,1273,1383,994,1062,1103,1140,1179,1216,1259,1295,1344,1207,1141,1024,996,90↙
5,863,826,790,1652,1728,1805,1456,1494,1533,1570,1611,1650,1690,1724,1768,1807,1856,19↙
02,1407,1460,1495,1533,1573,1612,1651,1692,1731,1766,1806,1855,1886,1931,1456,1536,145↙
8,1495,1538,1575,1617,1653,1692,1730,1770,1807,1855,1888,1748,1538,1411,1457,1499,1537↙
,1575,1615,1651,1691,1729,1768,1808,1855,1808,1936,1895,1825,1769,1576,1457,1458,1459,↙
1499,1538,1578,1616,1654,1692,1730,1770,1809,1860,1690,1482,1541,1411,1547,1500,1536,1↙
576,1617,1655,1694,1733,1768,1812,1861,1595,1810,1794,1459,1499,1536,1578,1617,1656,16↙
96,1733,1772,1812,1861,1636,1656,1411,1460,1498,1538,1580,1615,1655,1692,1734,1773,181↙
4,1463,1463,1418,1938,1576,1617,2133,2092,1729,1864,1674,1522,1412,1463,1504,1542,1579↙
,1619,1658,1698,1864,1599,1407,1483,1775,1466,1504,1542,1583,1621,1659,1696,1912,1861,↙
1757,1520,1714,1415,1469,1506,1546,1583,1137,1661,1699,1759,1642,1601,1449,1464,1506,1↙
545,1591,1624,1660,1700,1417,1467,1506,1544,1585,1622,1662,1700,1862,1939,1910,1892,18↙
60,1812,1792,1756,1804,1861,1942,1996,2058,2096,2233,1975,2016,2057,2096,2134,2174,221↙
3,2249,2289,2329,2376,1980,2020,2055,2096,2133,2174,2212,2251,2290,2327,2416,2247,2771↙
,2829,2791,2579,2459,2498,2539,2580,2615,2655,2695,2733,2771,2813,2858,2825,2755,2713,↙
2503,2461,2500,2542,2578,2615,2655,2699,2733,2776,2811,2859,2892,2911,2941,2997,3115,3↙
176,3212,3253,3295,3293,3251,3214,3176,3136,3095,3062,3022,2984,3002,3157,3013,3002,29↙
85,3024,3063,3101,3141,3180,3217,3254,3333,3441,3358,3360,3331,3441,3333,1979,2017,205↙
7,2093,2133,2172,2213,2251,2290,2330,2378,2058,2135,2172,2211,2021,1980,2017,2058,2094↙
,2135,2177,2216,2251,2291,2328,2462,2505,2542,2581,2621,2659,2695,2733,2775,2810,2861,↙
2736,2677,2532,2500,2417,2466,2503,2544,2583,2620,2658,2699,2738,2777,2813,3121,3219,3↙
034,2939,2938,2941,2940,2938,3092,3093,3094,3097,3092,3160,3160,3164,3160,3161,3200,31↙
97,3198,3198,3200,1988,2097,2155,2290,2331,2249,2380,2133,1980,2015,2056,2096,2135,217↙
3,2217,2076,2559,2598,2736,2773,2463,2481,2464,2500,2540,2581,2619,2659,2699,2737,2863↙
,2891,2909,2940,2985,3028,3066,3106,3124,3142,3140,3084,3179,3219,3255,3337,3335,3396,↙
2062,1939,1984,2022,2060,2098,2138,2176,2217,2339,1991,1994,2291,2313,2345,2274,2235,1↙
938,1935,1934,1934,1936,1975,2089,2089,2090,2089,2085,2236,2236,2238,2237,2232,2272,22↙
74,2275,2279,2276,2380,2343,1846,1985,2021,2058,2095,2134,2179,2214,2197,2017,1933,198↙
5,2020,2060,2097,2224,2182,2217,1987,2019,2058,2099,2142,2177,2217,2045,2103,1984,2026↙
,2063,2103,2141,2179,2219,2066,2181,1985,2025,2066,2104,2141,2180,2222,1987,2028,2064,↙
2105,2146,2182,2220,2418,2466,2505,2544,2583,2620,2661,2719,2738,2787,2862,2943,2990,3↙
025,3068,3105,3143,3186,3223,3264,2679,2467,2503,2539,2600,2640,2680,2730,2766,2806,28↙
63,2698,3027,3064,3104,2466,2505,2545,2583,2626,2660,2697,2737,2865,2988,3032,3068,310↙
7,3144,3182,3225,3261,3339,2448,2528,2700,2786,3051,3130,2418,2464,2505,2546,2586,2621↙
,2662,2701,2741,2860,2940,2989,3030,3068,3108,3148,3182,3222,3261,2768,2910,3299,3334,↙
3441,2903,3066,3009,2808,2341,2295,2381,2415,2417,2420,2422,2421,2571,2573,2574,2578,2↙

```
575,2626,2641,2642,2644,2644,2646,2682,2681,2680,2681,2683,2712,2711,2728,2759,2347,24↙
23,2473,2509,2549,2588,2624,2667,2705,2585,2743,2346,2423,2475,2512,2550,2590,2628,267↙
0,2710,2472,2630,2706,2471,2628,2712,2319,2346,2471,2512,2550,2589,2629,2668,2706,2590↙
,2631,2768,2348,2425,2473,2518,2552,2675,2627,2668,2705,2738,2761,2795,2823,2836,2863,↙
2985,3025,3151,3104,3143,2700,3217,2942,2987,3031,3069,3106,3148,3185,3218,3257,2993,3↙
015,3052,3149,3277,3414,3376,3343,3452,2865,3088,3242,2990,3028,3069,3106,3147,3185,32↙
25,3340,3034,3070,3166,3194,3245,3296,3341,3379,3369,3418,3445,2917,2865,2898,2837,286↙
7,2946,2944,2992,3031,3072,3109,3151,3185,3223,3259,3344,3050,3091,3174,3247,3021,3071↙
,3153,3226,2911,2992,3033,3070,3111,3,3187,3229,3265,3342,3381,3185,2869,2944,2994,304↙
1,3074,3109,3153,3191,3229,3270,3273,3313,3346,3386,3385];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Pr76',
                    obj.dim=76;
                    obj.optima=108159;
                    x=↙
[3600,3100,4700,5400,5608,4493,3600,3100,4700,5400,5610,4492,3600,3100,4700,5400,6650,↙
7300,7300,6650,7300,6650,5400,8350,7850,9450,10150,10358,9243,8350,7850,9450,10150,103↙
60,9242,8350,7850,9450,10150,11400,12050,12050,11400,12050,11400,10150,13100,12600,142↙
00,14900,15108,13993,13100,12600,14200,14900,15110,13992,13100,12600,14200,14900,16150↙
,16800,16800,16150,16800,16150,14900,19800,19800,19800,19800,200,200,200];
                    y=↙
[2300,3300,5750,5750,7103,7102,6950,7250,8450,8450,10053,10052,10800,10950,11650,11650↙
,10800,10950,7250,6950,3300,2300,1600,2300,3300,5750,5750,7103,7102,6950,7250,8450,845↙
0,10053,10052,10800,10950,11650,11650,10800,10950,7250,6950,3300,2300,1600,2300,3300,5↙
750,5750,7103,7102,6950,7250,8450,8450,10053,10052,10800,10950,11650,11650,10800,10950↙
,7250,6950,3300,2300,1600,800,10000,11900,12200,12200,1100,800];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Pr144',
                    obj.dim=144;
                    obj.optima=58537;
                    x=↙
[4350,4500,4300,4300,4300,4300,4350,4500,4300,4300,4300,4300,4950,5150,5525,5525,5525,↙
5525,4950,5250,5550,4950,5150,5525,5525,5525,5525,4950,5250,5550,5875,5875,5875,5875,5↙
675,5675,5875,5875,5875,5875,5675,5675,8125,8225,8325,8125,8225,8325,8675,8675,8675,86↙
75,8675,8675,8675,8675,8675,8675,8425,8525,8675,8675,8675,8675,8675,8675,8675,8675,867↙
5,8675,8425,8525,10850,10850,10850,10850,10900,11050,10850,10850,10850,10850,10900,110↙
50,11500,11800,11500,11700,11500,11800,11500,11700,12075,12075,12225,12225,12075,12075↙
,12100,12425,12425,12425,12425,12075,12075,12225,12225,12075,12075,12100,12425,12425,1↙
2425,12425,14675,14675,14775,14875,14975,15075,15225,15225,15225,15225,15225,15225,152↙
25,15225,15225,15225,14775,14875,14975,15075,15225,15225,15225,15225,15225,15225,15225↙
,15225,15225,15225];
                    y=↙
[4425,4425,4725,4825,4950,5050,8875,8875,9175,9275,9400,9500,10600,10600,9525,9425,922↙
5,9125,8875,8875,8875,6150,6150,5075,4975,4775,4675,4425,4425,4425,2325,2475,2625,2775↙
,4825,4925,6775,6925,7075,7225,9275,9375,10150,10150,10150,5700,5700,5700,3150,3250,33↙
50,3450,3550,3650,3750,3850,3950,4050,5700,5700,7600,7700,7800,7900,8000,8100,8200,830↙
0,8400,8500,10150,10150,9500,9400,9275,9175,8875,8875,5050,4950,4825,4725,4425,4425,44↙
25,4425,6150,6150,8875,8875,10600,10600,9525,9425,9375,9275,9225,9125,8875,7225,7075,6↙
925,6775,5075,4975,4925,4825,4775,4675,4425,2775,2625,2475,2325,5700,10150,10150,10150↙
```

```
,↙
10150,10150,8500,8400,8300,8200,8100,8000,7900,7800,7700,7600,5700,5700,5700,5700,4050↙
,3950,3850,3750,3650,3550,3450,3350,3250,3150];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Pr152',
                    obj.dim=152;
                    obj.optima=73682;
                    x=↙
[2100,2100,2100,2100,2100,2100,2100,2100,2348,2350,2348,2350,2348,2350,2348,2350,2625,↙
2775,2625,2634,2607,2625,2775,2625,2607,2634,2625,2775,2625,2607,2634,2625,2775,2625,2↙
634,2607,2825,2825,2825,2825,8349,8353,8349,8353,8349,8353,8349,8353,8474,8576,8575,84↙
74,8576,8575,8474,8576,8575,8474,8576,8575,8625,8675,8675,8669,8625,8675,8675,8669,862↙
5,8675,8675,8669,8625,8675,8675,8669,9250,9250,9250,9250,9250,9250,9250,9250,9498,9500↙
,9498,9500,9498,9500,9498,9500,9784,9757,9775,9775,9784,9757,9775,9775,9784,9757,9775,↙
9775,9784,9757,9775,9775,9925,9975,9925,9975,9925,9975,9925,9975,15499,15503,15499,155↙
03,15499,15503,15499,15503,15624,15726,15775,15725,15624,15726,15775,15725,15624,15726↙
,15775,15725,15624,15726,15775,15725,15825,15825,15819,15825,15825,15819,15825,15825,1↙
5819,15825,15825,15819];
                    y=↙
[1850,3000,4400,5550,6950,8100,9500,10650,11205,10050,8655,7500,6105,4950,3555,2400,11↙
175,10995,10025,9748,9831,8625,8445,7475,7281,7198,6075,5895,4925,4731,4648,3525,3345,↙
2375,2098,2181,3025,5575,8125,10675,10106,9397,7556,6847,5006,4297,2456,1747,1777,1803↙
,2325,4327,4353,4875,6877,6903,7425,9427,9453,9975,9875,9675,9525,9450,7325,7125,6975,↙
6900,4775,4575,4425,4350,2225,1875,2025,1800,1850,3000,4400,5550,6950,8100,9500,10650,↙
11205,10050,8655,7500,6105,4950,3555,2400,2098,2181,2375,3525,4648,4731,4925,6075,7198↙
,7281,7475,8625,9748,9831,10025,11175,10995,10675,8445,8125,5895,5575,3345,3025,10106,↙
9397,7556,6847,5006,4297,2456,1747,1777,1803,2225,2325,4327,4353,4775,4875,6877,6903,7↙
325,7425,9427,9453,9875,9975,9675,9525,9450,7125,6975,6900,4575,4425,4350,1875,2025,18↙
00];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Pr226',
                    obj.dim=226;
                    obj.optima=80369;
                    x=↙
[15625,14625,14525,14425,14125,14025,13925,13975,13625,13475,13475,13475,13475,13325,1↙
3175,13100,12675,12025,11425,9725,7975,6975,6875,6775,6475,6375,6275,6325,5975,5825,58↙
25,5825,5825,5675,5525,5450,5025,4375,3775,2075,2075,3775,4375,5025,5450,5475,5475,582↙
5,6325,7975,8725,8725,8725,8725,8725,8725,8725,8725,8725,8725,8725,8725,8725,9725,1142↙
5,12025,12675,13100,13125,13125,13475,13975,15625,16375,16375,16375,16375,16375,16375,↙
16375,16375,16375,16375,16375,16375,16375,4875,4875,4875,4875,4875,4875,4875,4875,4975↙
,4975,4975,4975,4975,4975,4975,4975,12525,12525,12525,12525,12525,12525,12525,12525,12↙
625,12625,12625,12625,12625,12625,12625,12625,12625,12625,12625,12625,12625,12625,1262↙
5,12525,12525,12525,12525,12525,12525,12525,10075,9975,9875,9775,4975,4975,4975,4975,4↙
975,4975,4975,4875,4875,4875,4875,4875,4875,4875,2425,2325,2225,2125,1875,1975,2075,24↙
25,2525,2625,2725,2825,2925,3025,3125,3375,3475,3575,3675,3775,4075,4175,4275,4375,447↙
5,4575,4675,4775,4875,4975,5375,5475,5575,5675,5775,5875,5975,6075,6175,6275,9525,9625↙
,9725,10075,10175,10275,10375,10475,10575,10675,10775,11025,11125,11225,11325,11425,11↙
725,11825,11925,12025,12125,12225,12325,12425,12525,12625,13025,13125,13225,13325,1342↙
5,13525,13625,13725,13825,13925];
```

```
                    y=↙
[1150,1200,1200,1200,1200,1200,1200,1500,1200,1200,1600,1750,1900,1200,1200,1725,1725,↙
1300,1600,1450,1150,1200,1200,1200,1200,1200,1200,1500,1200,1200,1600,1750,1900,1200,1↙
200,1725,1725,1300,1600,1450,3850,2300,3300,2150,2150,2500,3000,2050,2200,2150,2550,26↙
50,2750,2850,2950,3050,3200,3300,3400,3500,3600,3700,3800,3850,2300,3300,2150,2150,250↙
0,3000,2050,2200,2150,2550,2650,2750,2850,2950,3050,3200,3300,3400,3500,3600,3700,3800↙
,6500,6700,6600,6800,6900,7000,7100,7200,6500,6600,6700,6800,6900,7000,7100,7200,6500,↙
6700,6600,6800,6900,7000,7100,7200,6500,6600,6700,6800,6900,7000,7100,7200,8450,8550,8↙
650,8750,8850,8950,9050,8450,8650,8550,8750,8950,8850,9050,8800,8800,8800,8800,8450,85↙
50,8650,8750,8850,8950,9050,8450,8650,8550,8750,8950,8850,9050,8800,8800,8800,8800,118↙
50,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850↙
,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,1↙
1850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,118↙
50,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850↙
,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850,11850];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Pr299',
                    obj.dim=299;
                    obj.optima=48191;
                    x=↙
[2156,2456,2355,2256,2656,2556,2975,3275,3375,3475,3575,3700,3900,3875,4275,4275,4375,↙
4500,4450,4550,4560,4775,4710,4860,5010,5160,5310,5460,5610,6375,6525,6675,6675,6525,6↙
875,7075,7260,7259,7625,7410,7560,7409,7560,7825,7710,7711,7875,7975,8075,8175,8275,83↙
75,8475,8825,8820,8825,7725,7725,7475,7625,7575,7075,6825,6775,6975,6675,6200,6425,600↙
0,6025,5675,5275,5425,5525,5025,4975,5075,5075,4775,4925,4675,4775,4425,4525,4175,4125↙
,4275,3875,3875,3975,3975,2925,2925,4175,4325,4275,4275,4425,4525,4600,4600,4775,5175,↙
5450,6775,7275,7075,7425,7525,7475,7775,7825,7875,8825,8725,8703,8825,8575,8725,8625,8↙
375,8475,7975,8225,8225,8125,7975,7750,5350,5450,5450,5250,5175,5175,5075,4975,5025,48↙
75,4175,4175,3925,3725,3325,2925,2575,2675,2275,2375,2175,2148,2249,2352,2451,2552,265↙
1,2925,3325,3725,3925,4175,4175,4975,5075,5175,5450,5450,6125,6025,7575,7425,7925,7875↙
,7825,7750,8125,8075,8025,8125,7975,8775,8775,8775,8775,8775,8625,8525,8075,7975,7975,↙
7825,7575,7600,7425,7425,7425,7325,6125,6025,5375,5275,5450,5175,5075,4525,4575,4575,4↙
425,4325,4175,4275,4275,3975,3875,2925,2925,3925,3825,3975,4325,4275,4175,4125,4500,44↙
00,4725,4675,4775,5175,4975,5175,5025,5375,5275,6050,5908,6450,6675,6575,6975,6875,677↙
5,7150,7150,7625,7575,7475,7425,7825,7725,7825,7725,8225,8225,8475,8325,8575,8775,8675↙
,8575,8425,8325,8375,8175,8225,8075,8075,7925,7875,7075,7150,6975,6575,6375,5975,5325,↙
5275,5375,5075,5204,4675,4925,4925,4775,4575,4425,4325,3825,3625,3475,3225,3325,3125,2↙
925,2575,2675,2275,2375,2175,4775];
                    y=↙
[1639,1639,1640,1640,1640,1639,1725,1600,1600,1600,1600,1600,1625,1775,1625,1725,1675,↙
1675,1750,1750,1565,1725,1565,1565,1565,1565,1565,1565,1565,1575,1575,1575,1725,1725,1↙
575,1575,1575,1576,1775,1575,1575,1576,1576,1675,1575,1575,1575,1575,1575,1575,1575,15↙
75,1575,1625,1928,1825,1825,1975,1975,1925,2075,1975,1925,2075,2075,1925,1975,2075,197↙
5,2075,1825,1825,1825,1825,1825,2025,1925,2075,1875,1925,1950,1975,2050,2050,1950,2050↙
,2050,1900,2075,1975,2075,1875,2150,2150,2150,2225,2325,2150,2150,2225,2325,2125,2325,↙
2375,2225,2275,2175,2325,2375,2175,2350,2175,2350,2225,2375,2197,2425,2450,2525,2600,2↙
600,2600,2475,2500,2600,2575,2625,2650,2400,2525,2675,2400,2475,2675,2575,2525,2675,26↙
75,2425,2625,2450,2450,2450,2425,2525,2525,2525,2525,2525,2786,2786,2786,2785,2785,278↙
7,2925,2900,2900,2900,2925,2725,2825,2775,2875,2825,2925,2875,2875,2875,2875,2875,2775↙
,2925,2800,2775,2875,2775,2975,2975,2825,2925,3025,3125,3275,3225,3225,3075,3275,3075,↙
3125,3075,3175,3275,3125,3025,3225,3025,3175,3250,3250,3025,3150,3275,3200,3125,3025,3↙
200,3200,3200,3125,3025,3275,3275,3198,3475,3475,3475,3375,3575,3300,3400,3300,3300,33↙
```

```
00,3525,3400,3375,3300,3325,3475,3450,3525,3525,3375,3574,3375,3325,3525,3525,3325,342↙
5,3325,3475,3525,3425,3525,3425,3525,3525,3375,3375,3425,3325,3525,3525,3325,3700,3800↙
,3700,3625,3625,3775,3625,3775,3775,3675,3775,3675,3675,3750,3725,3750,3750,3750,3775,↙
3675,3675,3800,3772,3675,3800,3675,3675,3675,3675,3725,3650,3750,3750,3750,3750,3725,3↙
725,3675,3675,3675,3675,3675,3225];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Rat99',
                    obj.dim=99;
                    obj.optima=1211;
                    x =↙
[6,15,24,33,48,57,67,77,86,6,17,23,32,43,55,65,78,87,3,12,28,33,47,55,64,71,87,4,15,22↙
,34,42,54,66,78,87,7,17,26,32,43,57,64,78,83,5,13,25,38,46,58,67,74,88,2,17,23,36,42,5↙
3,63,72,87,2,16,25,38,42,57,66,73,86,5,13,25,35,46,54,65,73,86,2,14,28,38,46,57,63,77,↙
85,8,12,22,34,47,58,66,78,85];
                    y =↙
[4,15,18,12,12,14,10,10,15,21,26,25,35,23,35,36,39,35,53,44,53,49,46,52,50,57,57,72,78↙
,70,71,79,77,79,67,73,81,95,98,97,88,89,85,83,98,109,111,102,119,107,110,110,113,110,1↙
24,134,129,131,137,123,135,134,129,146,147,153,155,158,154,151,151,149,177,162,169,177↙
,172,166,174,161,162,195,196,189,187,195,194,188,193,194,211,217,210,216,203,213,206,2↙
10,204];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Rat195',
                    obj.dim=195;
                    obj.optima=2323;
                    x=↙
[3,17,23,34,47,54,66,75,86,94,107,115,123,3,15,26,33,42,53,64,74,85,95,104,113,125,3,1↙
5,26,36,48,54,64,75,88,98,103,115,127,6,15,27,36,47,54,66,74,85,94,107,117,125,6,13,25↙
,37,47,53,63,77,83,94,103,115,123,7,15,24,36,43,56,64,73,86,98,104,117,126,6,17,27,35,↙
44,54,63,77,82,96,103,116,126,7,16,24,35,45,55,63,75,87,93,104,117,127,3,16,25,35,44,5↙
3,64,76,87,95,106,114,125,3,16,25,37,44,54,66,77,85,93,106,113,125,5,15,24,33,43,53,64↙
,74,84,94,104,115,127,6,13,26,34,43,55,67,75,87,95,105,117,127,6,15,26,33,47,58,65,73,↙
87,94,104,113,125,5,16,24,37,45,54,67,74,87,95,106,116,127,7,17,23,33,43,53,67,73,87,9↙
4,104,114,127];
                    y=↙
[12,12,9,11,11,12,16,7,6,8,9,14,15,32,32,34,34,34,25,32,32,34,28,25,31,34,48,46,50,54,↙
50,46,54,44,49,50,54,47,49,75,75,73,73,68,72,68,67,65,74,65,65,68,84,95,94,84,87,95,86↙
,93,89,95,92,95,93,114,111,112,108,112,105,112,112,107,108,113,115,109,127,125,134,126↙
,131,132,124,127,134,128,126,130,134,152,147,153,151,154,146,155,151,154,156,151,153,1↙
48,164,172,165,175,169,174,168,171,173,174,168,169,169,190,188,195,186,189,194,192,192↙
,188,185,192,193,195,207,213,209,214,206,211,213,212,212,209,215,206,209,229,227,235,2↙
25,227,225,229,234,230,235,228,225,230,249,246,255,246,248,252,248,247,249,245,256,246↙
,253,266,274,267,266,267,266,267,265,264,271,264,271,273,287,294,287,284,288,295,288,2↙
86,293,284,291,294,290];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Rat783',
                    obj.dim=783;
```

```
                obj.optima=8806;
                x =↙
[13,49,105,239,110,270,53,131,88,97,244,78,254,174,10,200,29,62,123,141,201,167,39,219↙
,14,200,204,220,100,127,151,188,88,231,167,140,26,253,101,270,44,181,245,211,30,54,69,↙
116,7,159,189,13,73,175,213,2,25,77,142,222,143,165,94,125,67,237,46,150,200,38,132,17↙
9,228,107,243,82,113,207,261,37,51,163,71,255,140,154,248,8,104,208,222,180,196,68,96,↙
232,24,84,138,40,267,121,180,217,120,259,58,120,250,20,59,107,218,246,31,98,15,140,226↙
,75,154,240,20,160,0,149,45,113,177,185,204,200,89,260,262,60,89,136,17,110,190,240,90↙
,164,221,3,24,231,254,154,63,77,47,181,31,60,126,207,170,36,81,61,111,140,144,217,188,↙
230,119,237,266,17,151,220,42,103,252,210,99,20,249,137,170,59,0,20,76,174,195,121,194↙
,212,267,248,46,52,182,100,259,109,170,78,200,70,31,120,226,82,140,158,138,137,171,240↙
,2,13,113,255,141,201,175,60,232,84,124,170,58,190,187,36,46,107,116,269,3,14,158,249,↙
21,95,213,11,76,179,224,28,236,51,60,119,148,218,140,266,30,76,163,94,103,188,228,152,↙
193,246,252,1,73,86,125,7,206,207,44,55,158,30,25,44,231,260,120,175,185,228,265,68,16↙
6,191,15,109,82,130,97,37,147,238,243,219,68,113,171,23,130,243,6,86,106,193,224,186,1↙
45,160,14,258,270,97,47,155,58,210,119,130,67,220,80,152,56,30,254,140,230,92,117,220,↙
50,145,194,184,71,175,250,226,15,36,120,83,104,170,208,99,162,9,82,152,253,21,267,44,1↙
6,141,210,220,9,175,79,50,267,196,239,113,68,127,187,243,140,89,108,219,229,150,160,32↙
,178,11,200,156,30,201,58,23,130,67,100,257,3,237,46,221,246,181,28,178,200,266,73,266↙
,39,108,109,115,137,215,259,188,190,167,133,143,126,94,238,60,14,85,41,73,221,245,6,78↙
,118,183,67,152,126,6,156,55,113,208,137,248,46,266,17,177,218,149,83,193,231,24,31,22↙
3,66,90,102,26,166,77,258,268,63,140,198,30,113,220,130,133,238,5,80,179,200,150,97,22↙
3,253,181,246,16,103,30,40,58,169,200,180,266,100,64,187,146,80,159,161,220,1,103,115,↙
210,248,259,131,35,122,56,210,238,15,50,85,182,94,266,44,74,80,191,130,169,258,177,210↙
,13,160,209,102,30,67,227,4,35,118,123,205,250,60,68,131,141,166,212,230,245,71,12,240↙
,251,27,92,183,38,266,127,83,152,192,7,55,42,107,114,150,105,230,13,56,146,174,185,194↙
,241,164,9,141,96,43,25,201,230,257,132,263,33,119,86,178,64,129,231,216,32,239,11,46,↙
52,73,140,145,259,119,207,162,215,106,245,123,180,30,61,77,151,200,222,99,182,83,265,1↙
12,127,264,8,255,25,58,105,200,141,165,197,153,66,242,135,218,12,174,9,79,188,239,80,4↙
8,140,221,32,99,186,133,30,154,164,57,249,96,250,27,70,108,174,237,198,10,123,207,210,↙
120,224,40,10,187,208,76,214,262,29,48,63,90,262,10,112,56,135,246,30,155,67,86,90,171↙
,226,127,233,2,82,121,154,160,208,231,1,40,58,103,200,28,143,252,260,95,253,230,198,24↙
3,42,109,117,130,76,188,216,14,142,167,233,263,180,180,64,96,15,39,52,107,9,72,143,193↙
,208,68,247,47,183,22,138,157,167,228,213,252,118,121,207,84,70,83,50,30,157,171,215,5↙
0,141,260,94,8,169,251,38,130,132,240,76,114,12,105,184,199,229,231];
                y =↙
[6,6,6,6,7,7,8,8,9,9,9,10,10,11,13,13,14,14,14,15,15,17,18,18,19,19,19,19,20,20,21,21,↙
22,23,24,25,28,30,33,33,35,35,35,37,38,38,38,38,39,39,39,40,40,40,40,41,41,41,41,41,42↙
,45,46,46,49,49,51,53,53,55,55,55,55,56,56,57,57,58,58,59,59,59,60,60,62,62,62,63,63,6↙
3,64,65,65,66,68,68,69,69,70,71,72,73,73,75,76,78,79,80,80,81,81,82,82,83,86,89,90,90,↙
91,92,92,93,94,94,95,95,96,97,97,97,97,98,99,99,99,101,101,101,103,103,104,104,105,105↙
,106,107,107,107,107,109,110,110,111,111,114,115,117,117,118,119,120,121,121,121,121,1↙
21,122,122,124,127,128,129,129,129,130,130,131,133,134,135,135,137,137,138,139,139,140↙
,140,140,141,141,141,141,142,146,146,146,147,147,148,149,151,151,152,154,154,155,156,1↙
56,156,157,159,159,160,161,161,161,163,164,165,166,167,167,168,168,169,170,171,173,174↙
,174,174,174,174,175,175,178,178,179,179,180,181,181,181,182,184,184,186,186,186,187,1↙
87,188,188,190,191,191,192,193,193,193,194,197,197,197,198,198,198,198,199,199,199,201↙
,201,201,203,204,204,204,204,207,207,207,207,209,210,212,212,215,215,216,217,218,219,2↙
19,219,219,220,221,221,222,223,223,224,225,225,225,226,226,227,228,228,229,231,231,232↙
,233,233,235,235,237,238,239,239,240,240,242,243,243,244,244,246,246,246,248,249,250,2↙
51,252,252,252,253,255,255,256,257,257,257,258,259,259,260,260,260,260,261,261,262,263↙
,264,264,264,268,268,270,271,271,272,272,273,274,274,275,276,277,278,278,279,279,280,2↙
80,281,282,283,283,284,285,285,289,290,290,291,291,291,292,292,293,294,296,297,299,299↙
,299,299,300,300,301,301,301,301,301,301,302,304,304,307,308,309,310,312,312,315,316,3↙
16,317,317,318,318,319,319,319,319,320,321,323,324,324,325,325,325,326,326,328,329,330↙
```

```
,↙
330,330,331,333,333,333,334,335,336,337,337,337,339,339,340,340,344,345,346,346,348,34↙
8,348,349,349,349,350,350,350,350,352,354,354,354,356,356,358,359,360,360,360,360,361,↙
362,362,365,366,366,367,368,368,368,368,369,369,373,373,374,374,375,376,376,378,378,37↙
8,379,379,379,379,381,383,384,384,384,384,385,385,385,389,391,392,392,392,393,394,395,↙
397,398,398,398,398,399,399,400,400,400,400,400,400,400,400,401,404,404,404,405,406,40↙
6,410,410,411,412,413,413,415,416,418,418,419,419,420,420,421,421,421,421,421,421,421,↙
422,423,423,424,425,431,431,432,432,434,434,435,435,436,436,437,437,437,438,439,439,44↙
0,440,440,440,442,442,442,443,443,444,445,447,447,448,450,451,451,453,453,453,453,455,↙
456,457,458,459,459,459,460,460,462,462,462,462,463,464,464,465,466,466,467,467,468,46↙
8,469,469,469,475,476,478,479,480,481,481,481,483,484,484,485,486,486,487,487,488,489,↙
489,489,489,491,492,492,492,493,494,495,498,499,499,499,500,500,500,501,501,501,501,50↙
1,503,504,506,506,508,509,509,510,510,512,513,513,517,518,519,519,519,519,519,519,519,↙
520,520,520,520,520,521,521,521,521,525,525,527,528,528,533,533,533,534,535,536,536,53↙
8,538,539,539,539,540,541,542,543,544,544,544,545,546,546,546,546,547,548,549,550,552,↙
553,553,553,553,553,555,557,558,558,559,560,561,562,564,565,565,565,565,566,568,569,57↙
0,571,571,574,576,577,577,578,579,579,580,580,580,580,580,580];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Rl1304',
                    obj.dim=1304;
                    obj.optima=252948;
                    x=↙
[15440,15440,15440,15440,15440,15440,15536,15616,15824,16048,16176,16384,16512,16624,1↙
6704,16784,16880,17024,17312,17440,17600,17872,17936,6320,6320,6432,6480,6480,6480,648↙
0,1232,1232,1232,1232,1456,1520,4272,4272,4304,4416,4496,4528,4528,4592,4704,4784,4880↙
,4944,5120,5200,5360,5504,5616,5648,5648,5680,5808,5952,6096,6160,6240,5648,6288,6288,↙
6288,6288,6688,6800,6912,7072,7120,7120,7120,7120,18432,18720,18912,19040,14128,14128,↙
14128,14128,14320,14464,14608,14672,14736,14800,14896,15024,15280,15280,15280,15280,35↙
04,3504,3504,3648,3760,3872,4096,4304,4784,4880,4960,5136,5216,5344,5776,5808,5808,580↙
8,784,784,816,912,1056,1104,1104,9104,9104,9104,9104,9104,9216,9296,9456,9568,9664,977↙
6,9840,9904,10080,10336,10448,10544,10640,10768,10864,10960,11088,11168,11344,11568,11↙
696,11792,11824,11824,11824,11824,11824,10768,10768,10768,10768,11024,11088,11184,1128↙
0,11584,11760,11760,11760,11760,11760,10064,10064,10064,10064,10064,10128,10192,10336,↙
10448,10640,10864,10960,11024,11088,11088,9072,10480,10544,10640,10736,11216,11280,114↙
08,11536,11936,12064,12112,12112,12112,12112,12112,8688,8688,8688,8688,8720,8800,8944,↙
9488,9712,9904,9968,10112,10192,10544,10640,11600,11728,11968,12048,12048,12192,12288,↙
12368,12400,12400,12400,12400,12400,12400,8528,8528,8528,8528,8528,8560,8720,8800,1233↙
6,12336,12336,12336,12336,8464,8464,8464,8560,8560,8560,8560,8560,7216,7216,7216,7216,↙
7216,7216,7408,7504,7568,7664,7728,7760,7760,7760,7760,7760,12560,12560,12560,12560,12↙
560,12640,12752,12848,13056,13152,13232,13328,13392,13584,13696,13808,14032,13680,1368↙
0,13680,13680,13680,13680,13680,13712,13808,14064,14320,14448,14512,14608,14672,14736,↙
14800,14896,15024,15408,15536,15616,15824,16048,16176,16384,16512,16672,16768,17072,17↙
168,17312,17440,17536,18128,17712,17712,17968,18112,12784,12784,12784,12784,12784,1419↙
2,14896,14896,14896,13712,4208,4208,4208,4592,4704,4784,5792,5968,6080,6224,6528,6608,↙
6800,6912,6992,7056,7136,7376,7440,7568,7712,7792,7856,7920,8,8112,8272,8368,8400,8400↙
,10480,10544,10640,10752,10832,10912,11008,11200,11280,11408,12304,12304,12304,12656,1↙
2656,12656,12656,12656,12656,14864,14864,14864,15024,15024,15024,4592,4656,4768,4880,4↙
976,5056,5488,5952,6096,6160,6560,7664,7792,7792,7792,7792,7792,16496,16496,16624,1670↙
4,16784,16880,16944,17008,17104,17296,17424,17616,17952,18080,18208,18368,18416,18416,↙
1264,16400,17904,3984,3984,3984,8272,8272,16336,16336,16336,16976,1008,5808,5808,5808,↙
5936,5936,4432,4656,4656,4656,4656,16848,17296,17424,18208,18384,18384,8240,8400,8432,↙
8432,8432,8432,1584,1584,1584,1616,1680,2576,2576,2672,2672,2672,2672,2672,17008,880,8↙
```

```
80,1104,5520,5520,5680,5680,3888,3888,16592,18096,8016,8016,8016,8144,8144,8144,8144,8↙
144,16784,16784,17072,17168,17312,17440,17536,17664,17776,784,1104,7728,7728,7728,7920↙
,7920,7920,7920,7920,7920,7920,5232,5232,5520,5520,5520,3856,4720,4976,4976,4976,18496↙
,7376,7376,7376,7376,7504,7504,7504,7504,1808,1808,1808,1840,1920,1968,12528,12528,125↙
28,12528,12528,12752,12848,12848,12848,12848,13056,13136,13328,13392,13456,13584,13808↙
,14064,14320,14448,14512,14608,14672,14736,14800,15536,15616,15824,16048,16176,16304,1↙
7088,17296,17456,1680,1776,1808,5936,5936,6160,6160,6160,6160,6160,18512,18512,18512,1↙
0320,11280,11408,11552,11600,11600,11600,4880,4944,5040,5120,5392,6064,6160,6224,6224,↙
6256,6384,6384,6384,6384,6384,1936,1936,1936,2064,6672,6672,6672,6896,6992,7056,7136,7↙
440,7568,7856,7920,8128,8800,8880,8944,9216,9296,9488,9552,9632,9712,9776,9936,10128,1↙
0192,10336,10448,10544,10864,10960,11024,11168,11280,11376,11440,11728,11808,11952,119↙
84,11984,11984,11984,11984,12080,12224,12464,8784,8784,8784,8848,8976,8976,8976,8976,8↙
976,3280,3280,3280,3280,3344,3408,3536,3648,4400,4768,4880,5040,7568,7568,10544,10544,↙
10544,10640,7824,7824,7824,7824,7920,7920,10608,10864,10960,11024,11168,11280,11376,11↙
440,11776,11952,12048,9520,9632,9712,9776,9936,10128,10192,10336,10448,11440,11472,115↙
36,12592,12768,13056,13136,13136,13136,6,6160,6432,6512,8880,8944,9216,9296,9488,9552,↙
9632,9712,9776,9936,10352,10352,10544,656,656,16304,17648,6608,208,10864,11024,11184,1↙
1280,11376,12592,12768,13056,13232,13328,13392,13456,13584,9520,9552,9632,9712,9776,99↙
36,10128,10192,10448,10960,11024,11168,11280,11376,11728,12096,12384,12480,12720,13072↙
,13152,13232,13296,13392,13584,13808,13808,13808,13808,13808,13808,3120,3120,3120,3120↙
,3120,3184,3248,3280,3280,3280,3280,3280,11376,3152,3152,3152,3152,1392,1392,1392,1392↙
,1424,1520,464,528,7664,7664,7664,5360,5360,5360,5360,5392,10032,10192,10192,3184,3344↙
,3408,3536,3648,3744,4080,4592,4768,4880,5056,5136,5216,9936,10096,1328,1504,2144,2224↙
,2384,2480,2544,2816,2976,3088,3360,3440,3664,3744,4016,4592,4784,4880,6992,7056,7136,↙
7312,9232,9232,9296,9328,9328,4880,4048,8304,9776,12592,12752,12784,12784,7056,14320,1↙
4320,14320,14512,14512,14512,14704,14704,14704,14736,14736,2928,2928,3184,4592,4592,13↙
072,13072,13072,13072,13072,13072,10480,10480,10480,11200,11280,11536,12592,6896,6896,↙
7120,7120,3408,3408,3408,3408,3408,3728,7408,7408,7408,7408,3568,2896,2896,2976,4752,4↙
752,12720,12720,12720,12720,2800,2800,2800,2800,2832,2832,2832,2832,8112,8112,8208,820↙
8,1328,1328,1328,12464,12496,12496,9296,9296,9456,9584,9680,9744,9856,10336,10416,1075↙
2,5072,6512,7440,7440,12368,12368,13456,13584,14064,15536,15616,15824,16048,16176,1451↙
2,14512,17520,14032,14192,14192,12080,12496,16656,13232,13232,13232,13232,13456,10192,↙
14320,14320,14320,14320,14448,14448,14768,14768,14768,14800,9872,9872,10128,16496,1649↙
6,15376,15376,15376,15376,15408,6096,6096,6096,6672,6672,7248,7248,7248,7248,8880,8944↙
,13152,13296,13392,13584,14032,14608,15520,15648,15760,15824,15920,15664,15664,15664,1↙
6144,14608,14608,15504,16192,5328,6064,9616,9680,9680,8368,10416,4336,5136,5216,5600,6↙
560,6784,8848,9456,9584,9744,9968,13168,13168,13168,3344,3344,6544,6576,6576,6736,8720↙
,8944,9456,3632,15536,15536,15600,6448,1504,2,2128,2224,2256,2256,2256,5600,6784,8848,↙
9200,9584,9744,13584,14064,14960,15248,15536,15824,16048,16208,16304,16592,16752,12880↙
,13232,13296,15248,16176,16176,16176,16208,13296,13392,13584,14032,14608,15760,15824,1↙
5920,16032,2384,2384,2384,2384,2224,2480,2544,2960,3088,1504,2544,2544,2544,2672,2672,↙
8848,8848,3088,3344,3184,3184,2352,8880,13296,13296,13328,17232,17312,17424,18080,1816↙
0,18240,18336,18640,13392,13584,14032,14608,15760,15824,15920,16032,16704,16944,17072,↙
17232,17264,15920,16064,9712,17328,17968,18192,18704,18928,13232,14032,13584,14672,133↙
92,13392,5600,17440,1888,2128,2128,2064,2480,2480,2160,2064,3744,18640,18704,15696,920↙
0,15824,8016,9040,9104,18192,19024,9200,6320,18224,14944,1024,5872,2624,8080,4848,6224↙
,6800,6800,10592,12016,9360,11440,6624,6048,304,688,7632,560,752,6992,7408,8352,14720,↙
12736,2816,8160,12608,11664,14496,608,12480,16640,14384,7712,14784,16512,15392,5712,58↙
40,6096,6688,15952,4560,4784,8416,6128,6576,15568,944,2160,15984,14224,14832,14992,153↙
76,16176,16176,16560,16720,16848,12944,14352,16304,2448,3024,8832,10992,3376,13312,171↙
36,17264,13184,16736,15968,15520,205,16032,16064,14048,14640,5232,5232,6992,12304,1230↙
4,13904,336,336,3792,3792,880,2224,2608,3440,17888,11232,18144,15856,15856,7152,7152,7↙
280,7296,7312,7568,8624,8912,9232,9520,10576,10864,7696,9072,9184,10992,11120];
                    y=↙
[8888,9328,9768,11264,13288,14058,8228,8228,8228,8228,8228,8228,8228,8228,8228,8228,82↙
```

28,8228,8228,8228,8228,8228,4400,10626,12012,9680,4180,4796,5544,6732,5918,7304,12012,↙
14432,5588,4730,11154,11858,10296,10296,10296,5544,7040,10296,10296,10296,10296,10296,↙
10296,10296,10296,10296,10296,7942,9020,6952,6952,6952,6952,6952,6952,14322,4180,4796,↙
5544,8052,9020,9020,9020,9020,5280,6732,7172,7942,4422,4422,4422,4422,5610,7480,8888,9↙
328,9548,9548,9548,9548,9548,9548,9548,9548,9768,11264,13288,14058,9240,10142,12012,90↙
20,9020,9020,9020,9020,9020,9020,9020,9020,9020,9020,9020,4488,5544,8052,4400,5500,677↙
6,6776,7524,12012,14432,10472,11704,12540,13508,14058,9548,9548,9548,9548,9548,9548,95↙
48,9548,9548,9548,9548,9548,9548,9548,9548,9548,9548,9548,9548,9548,9548,9548,5324,651↙
2,7172,7612,8492,11484,12540,13068,13970,8932,8932,8932,8932,8932,5324,6512,7172,7612,↙
8382,11704,12540,13068,13508,14190,11484,11484,11484,11484,11484,11484,11484,11484,532↙
4,6842,8976,8492,8492,8492,8492,8492,8492,8492,8492,8492,8492,5104,5764,6512,7172,7612↙
,11594,12540,13508,14058,10956,10956,10956,10956,10956,10956,10956,10956,10956,10956,1↙
0956,10956,10956,10956,9284,10362,10736,10736,10736,4796,5874,6512,7172,7612,8272,1036↙
2,11594,12540,13508,14058,9988,9988,9988,4796,6512,7172,7612,8272,10516,11594,12540,51↙
70,6732,7282,7832,8866,9350,9988,10516,11374,12430,14432,8932,8932,8932,8932,8932,4906↙
,5544,6732,7172,7722,10516,11264,11814,13288,14058,9768,9768,9768,9768,9768,9768,9768,↙
9768,9768,9768,9768,9768,8888,9328,10516,11264,11814,13288,14058,7480,7480,7480,7480,7↙
480,7480,7480,7480,7480,7480,7480,7480,7480,7480,7480,7480,7480,7480,7480,7480,7480,74↙
80,7480,7480,7480,7480,7480,7480,10626,14058,8448,8448,10406,11264,11814,13288,14058,9↙
988,4906,5874,8888,5390,4290,7040,8052,11264,11264,11264,11264,11264,11264,11264,11264↙
,11264,11264,11264,11264,11264,11264,11264,11264,11264,11264,11264,11264,11264,11264,1↙
1264,11264,11264,12540,14058,4796,5104,5104,5104,5104,5104,5104,5104,5104,5104,11814,1↙
3288,14058,4906,5874,7172,7612,8272,8888,11264,13288,14058,4906,5874,8888,7612,7612,76↙
12,7612,7612,7612,7612,7612,7612,7612,7612,7612,9350,9988,10516,12430,14168,4906,9922,↙
10736,10736,10736,10736,10736,10736,10736,10736,10736,10736,10736,10736,10736,10736,11↙
176,14168,4620,4906,14058,4180,7150,10076,12540,14058,10296,11264,13288,5126,4840,9768↙
,12320,14322,4488,5544,11858,4180,5544,7040,8162,13288,11484,11484,11484,7700,8668,105↙
16,9460,5170,6732,7282,7832,12012,13992,14432,6556,5214,4290,5170,9768,10296,12012,137↙
72,14212,13288,5390,12012,5060,12210,14322,4796,5544,7150,8052,13288,9768,10516,12430,↙
14058,5170,6732,7172,7832,8866,5236,13288,5016,5016,5016,5016,5016,5016,5016,12012,580↙
8,10516,12430,14168,4906,5544,6732,7172,7832,8602,9350,12430,14322,4796,5544,6732,1201↙
2,11858,5544,7040,8162,7700,9878,10516,12430,14300,4796,5390,6732,7172,12012,13882,144↙
32,10296,10296,5170,5874,6512,7172,7612,8272,8888,11264,11814,13288,14058,8888,8888,88↙
88,8888,8888,8888,8888,8888,8888,8888,8888,8888,8888,8888,8888,8888,8888,8888,8888,888↙
8,8888,9988,9988,9988,12012,6864,4840,12320,14322,4180,4796,5544,9240,9988,8668,9768,1↙
0428,5170,5852,5852,5852,12540,13068,13838,14432,14432,14432,14432,14432,14432,14432,9↙
988,12012,9460,4180,4796,5544,6732,7172,12012,13882,14432,5720,4796,5544,6732,12540,12↙
540,12540,12540,12540,12540,12540,12540,12540,12540,12540,12540,12540,12540,12540,1254↙
0,12540,12540,12540,12540,12540,12540,12540,12540,12540,12540,12540,12540,12540,12540,↙
12540,12540,12540,12540,12540,6512,7172,7612,9284,11704,12540,12540,9438,11594,13508,1↙
4058,9460,5170,6732,7282,7832,8976,4180,5170,7040,7832,8052,8052,8052,8052,8052,8052,8↙
052,8052,10516,14168,13068,13508,14190,7612,4906,5544,6732,7172,10516,14058,14190,1372↙
8,13728,13728,13728,13728,13728,13728,13728,13728,11704,14432,14432,14432,14432,14432,↙
14432,14432,14432,14432,13068,7612,7612,7612,7612,7612,5016,5874,7172,9240,11704,11704↙
,11704,11704,11704,11704,11704,11704,11704,11704,11704,11704,11704,13068,13508,7612,42↙
90,14212,9328,4400,14432,4422,13068,7172,7172,7172,7172,7172,7172,7172,7172,7172,7172,↙
7172,7172,13508,13068,13068,13068,13068,13068,13068,13068,13068,13068,13068,13068,1306↙
8,13068,13068,13068,11924,11924,11924,11924,11924,11924,11924,11924,11924,4796,5610,93↙
28,10516,11264,11704,4180,5170,7040,7832,8272,8492,8492,9350,10076,10516,12012,14212,6↙
402,9240,10098,12012,14212,7304,12012,13992,14432,6336,6336,13992,14212,5390,6732,7172↙
,4796,5544,7040,9658,12100,6732,5170,6292,7040,7040,7040,7040,7040,7040,7040,7040,7040↙
,7040,7040,7040,7040,13508,5170,12012,12012,12012,12012,12012,12012,12012,12012,12012,↙
12012,12012,12012,12012,12012,12012,12012,12012,12012,12012,10516,14432,14432,14432,13508,14↙
058,8052,5170,6732,5544,4180,10516,13508,8272,8272,4906,5874,10516,4796,5236,5874,1126↙
4,13288,14058,11264,13288,14058,4906,5874,4290,5060,7832,4180,5544,5016,5874,10516,112↙

```
64,13288,14058,6842,7612,13508,6512,6512,6512,6512,5170,6732,9988,10516,4180,5280,8712↙
,9240,9922,10516,4796,5390,6732,7172,5280,8272,9240,10076,4180,5544,10406,11264,13288,↙
14058,9768,10296,13772,14212,4290,5060,8272,9240,10516,14058,7832,8866,7304,13992,1443↙
2,11044,13288,14058,13508,14058,7612,7612,7612,7612,7612,7612,7612,7612,5544,14432,105↙
16,14168,13288,14058,9328,9328,9328,9328,9328,9328,9328,9328,4906,5874,4290,11704,1051↙
6,11264,14058,10516,4796,10516,11264,13288,14058,5390,13508,10516,11264,13288,14058,49↙
06,5874,11264,13288,14058,5874,5170,6732,13508,11264,13288,9768,11264,13288,14058,6138↙
,4796,5544,9240,12320,14432,4796,5390,6732,7172,14168,14168,14168,14168,14168,14168,14↙
168,14168,14168,14168,14168,14168,14168,9768,11264,13288,6358,4906,5874,9768,9768,1232↙
0,12320,13508,5170,6732,10516,6292,4620,5544,5544,5544,5544,5544,5544,5544,5544,5544,5↙
544,10516,11264,13288,10516,14212,4180,4796,6732,14432,8976,13508,6732,5280,11264,1328↙
8,6138,14432,14432,14432,14432,10604,4840,6160,9768,6732,6732,6732,6732,6732,6732,5456↙
,5456,4796,4796,6204,6204,6204,4796,4796,4796,4796,4906,5236,4906,5874,10296,11264,132↙
88,6512,11264,11264,11264,11264,11264,11264,11264,11264,11264,6160,9768,10296,13992,13↙
772,13772,13772,14212,9240,13992,9768,10296,14212,4290,5060,7282,7832,14212,5170,4180,↙
5170,4730,13508,10516,13288,5874,14168,14168,14168,14168,14168,14168,14168,14168,13288↙
,13288,13288,13288,13288,13288,13288,13288,13288,13288,13288,13288,13288,4290,10296,10296,13↙
508,8976,9768,9768,9768,8668,5874,10516,10516,5874,5874,10516,4796,4400,5500,6160,9768↙
,6160,9768,10296,4840,9768,5280,10428,8668,5764,7392,5764,4906,4796,4796,8668,8668,783↙
2,9020,4213,10208,5808,9328,5808,9460,14520,14520,13750,12694,11704,12848,14520,14520,↙
9680,4400,4213,14410,9680,4092,4092,14520,14520,9680,10208,9988,9460,9680,8492,9768,91↙
08,4708,10736,7700,10208,9680,10208,11044,9548,4092,4092,4092,11924,14256,4092,4092,99↙
00,4092,4092,9988,14520,14520,14146,4708,4708,4708,4708,4708,5478,4708,4708,4708,5236,↙
5104,5742,5500,9460,9680,9152,14410,10208,14256,13596,7920,7700,9988,9548,13992,10516,↙
10824,9548,10208,4224,4642,4708,4224,4620,4708,4213,4598,4466,5104,4092,4092,4092,4092↙
,8888,9152,9988,5016,5588,4708,4884,4708,4796,4708,4708,4708,4708,4708,4708,4708,4708,↙
5104,5016,5016,4708,4708];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Rl1323',
                    obj.dim=1323;
                    obj.optima=270199;
                    x=↙
[18192,18192,18192,18272,18416,18416,1200,1200,1200,1200,17904,17904,17904,17904,17904↙
,8272,8272,8272,8272,8272,8272,8336,8464,8496,8496,8496,15888,15888,15888,15888,15936,↙
16048,16208,16336,16336,16336,16336,16336,16336,16336,1040,1040,1040,1040,4464,4640,48↙
16,4944,5168,5440,5616,5728,5808,5808,5808,5808,5808,5808,5936,5936,16848,16848,16848,↙
16848,16848,16848,16848,16848,16848,16848,8240,8400,8400,8400,1520,1520,1520,1584,1584↙
,1584,5168,5280,5456,5584,5712,2672,2784,2976,3104,3264,3360,3600,3856,3920,4016,4144,↙
4208,4352,4464,4560,4688,4720,4720,4720,4720,4720,17008,17008,17008,17008,17008,17072,↙
17136,17280,17456,17520,17520,17520,17520,880,880,880,4144,4144,4304,4416,4576,4656,48↙
64,5040,5296,5456,5456,5520,5520,5520,5520,5520,16560,16560,16592,16592,16592,16592,16↙
592,16592,7728,7728,7728,7792,7856,7952,8016,8016,8016,8016,8016,8016,8016,16272,16272↙
,16272,16272,16640,16720,16784,16784,16784,464,528,640,784,784,7664,7664,7664,7664,772↙
8,7728,7728,7728,7728,7728,7728,5232,5232,5232,5232,5232,5520,3856,3920,4016,4176,4176↙
,4176,4176,4176,5984,6096,6096,6096,6160,6224,6320,6496,6608,6608,6608,6608,6608,6608,↙
17200,17200,17200,17200,17200,17296,17440,17568,17616,17616,7248,7248,7376,7376,7376,7↙
376,7376,7376,7376,7376,7376,7376,1808,1840,1840,1840,1840,1840,12272,12464,12592,1273↙
6,12848,12848,12848,12848,12848,12848,12848,12848,12848,12848,12848,12848,1648,1648,1648,164↙
8,1680,1680,6256,6256,16976,16976,16976,16976,17072,17136,17440,17600,11600,11600,1160↙
0,11600,11600,11600,11600,11632,11632,11632,17376,17728,18416,18608,18704,18864,18992,↙
19088,6224,6224,6224,6224,6400,6768,6864,1936,1936,1968,1968,1968,1968,3984,4304,4416,↙
4496,15440,15440,15440,15440,15536,15648,15760,15840,15936,16048,16208,5584,5728,6,606↙
```

4,6160,6320,6320,6320,6320,6384,6384,16720,17088,17376,17728,18416,18608,18704,18848,1↙
5280,15280,15280,15280,15280,15280,15280,15280,15280,15280,15280,15280,3472,3472,3472,↙
3472,3472,3472,208,432,944,1104,1424,1424,1424,10096,10096,10096,10096,10096,10096,100↙
96,10096,10096,10256,10336,10448,10544,10608,10672,10752,10864,10992,11088,11168,11280↙
,11408,688,688,6160,6496,6592,6736,6848,6944,7104,7232,7552,7600,7600,7600,7600,7600,7↙
600,7600,7632,7792,7856,7920,8112,8384,8464,8624,8800,8912,9216,9504,9616,9744,9888,99↙
36,9936,9936,9936,9936,10032,10032,10032,6496,6736,6768,6768,9936,9936,5200,7408,7568,↙
7856,7936,8032,8144,8224,8320,8784,8976,9408,10288,10400,10560,10704,10864,10960,11136↙
,11264,11392,11472,11568,4880,4880,4880,4880,4944,4944,4944,4944,4944,4944,6944,7104,7↙
232,7792,7856,7920,8128,8384,8464,8528,8592,8688,8800,8912,9008,9104,9200,9232,9232,92↙
32,9344,9456,9456,3568,3664,3728,8240,8240,9376,9776,9776,9776,9776,6992,6992,6992,699↙
2,7056,7056,7056,7056,7056,7056,14512,14640,14640,14640,14640,14640,14640,14640,14640,↙
14640,14640,14768,14848,14960,15376,15536,15664,15760,15840,15936,16048,16208,16464,16↙
528,14736,14736,14736,14736,14736,14736,14736,14768,14880,14976,15376,15536,15648,1576↙
0,15840,15936,16048,16208,16464,16704,13808,13808,13808,13808,13808,13808,13808,14032,↙
14144,14320,14464,16944,3664,3728,3760,3760,3760,3760,3600,3600,4048,4336,4576,4656,49↙
76,8976,8976,8976,10176,10304,10448,10576,10736,10832,10896,10976,11088,11168,11264,11↙
424,11536,11664,11728,11792,11872,11968,12112,12272,12448,12608,12736,13008,13072,1307↙
2,13072,13072,13072,13072,13168,13264,13408,13568,13696,14016,14160,14320,15120,15376,↙
10480,10480,10480,7120,7120,7120,7120,7120,7120,7120,7120,7120,7120,7120,3728,4016,401↙
6,4016,6480,6896,7248,7248,7248,7248,7248,7248,7248,7248,3920,5056,3536,3536,3536,3536↙
,8784,10304,10560,10704,10768,10848,10976,11120,11264,11552,11760,11888,12080,12240,12↙
720,12720,12720,12720,12720,2800,2928,2992,3104,3264,4208,8112,8112,8112,8112,8112,811↙
2,9584,9584,9840,11440,11728,11824,11984,12112,12256,12384,12464,12496,12496,12608,558↙
4,5680,5760,6896,7568,8144,8208,9136,10544,11088,11168,11264,11312,11312,11312,10768,1↙
0768,10768,10768,10768,10768,10768,10768,5584,5616,6160,6496,6608,6736,6848,6944,7792,↙
7856,7920,8384,8464,8560,8688,8800,8912,9008,8688,8464,8528,8528,8528,8464,8464,8464,1↙
7456,17456,17456,11728,11824,11984,12112,12256,12384,12464,12560,12560,12560,17552,176↙
32,17776,18080,18256,10448,10560,10704,10864,10976,11136,11760,11888,12032,12144,12880↙
,13008,13168,13312,13472,13584,13680,13680,13680,13680,13680,13680,17760,18048,18128,1↙
2640,12784,12784,10976,11088,11264,11664,11728,11792,11888,12912,13008,13184,13328,134↙
56,13568,8400,8400,8400,9328,9520,9632,9840,10160,10256,10336,10544,10608,10672,10864,↙
10992,11088,11168,12304,14864,14864,14864,14864,14864,14864,14864,14864,14864,7792,779↙
2,7792,7792,7792,7792,15376,15376,15376,15376,6864,6864,6864,6928,10544,10672,10704,12↙
464,12592,12736,12880,13008,13168,13312,13456,13584,14144,14224,14336,14416,14992,1544↙
0,15632,15760,15824,16160,16464,15952,15952,16048,16208,16464,16704,17072,17136,17296,↙
15664,15664,15664,15664,15664,15664,15664,15664,16432,8144,8144,9616,12112,12112,12112↙
,8304,8304,12912,13008,13184,13328,13456,13552,13552,13552,13552,13552,11920,11920,119↙
52,11952,11952,10320,10448,11824,11824,11824,12656,5008,5008,13168,13168,13168,13168,1↙
3168,13232,13376,13488,14032,14144,14224,14320,14464,14976,16704,3280,3280,6736,6736,6↙
736,6736,6736,6736,8784,10288,10512,11792,8720,6800,6800,6912,7856,8896,8944,8944,7568↙
,7856,8784,14144,14224,14336,14416,14480,15440,15504,15536,15536,15536,15568,15760,158↙
40,16048,16208,16704,17104,17296,18,18080,11728,11728,11728,11728,11760,12272,12272,10↙
512,7856,7920,8784,8784,8784,7920,7920,7920,7920,8880,10544,10544,10608,9008,9312,1144↙
0,11440,11440,11536,15824,15824,15824,15824,15824,6608,6608,6608,6896,112,10672,10832,↙
10864,10864,11088,9520,9520,12240,3280,3280,3632,3632,3632,11568,14128,14160,14160,141↙
60,14160,14160,14160,4080,4336,4576,4976,528,16016,16016,16016,16016,16160,16464,16640↙
,16720,17088,1744,2112,2368,2544,2944,3056,3056,3088,4080,4336,4432,13456,14224,14336,↙
14416,14448,14448,14448,1280,1744,1904,2160,2224,2368,2544,2704,14032,14032,14032,1497↙
6,14320,14768,13392,13392,13392,13392,16688,13360,16208,16208,1504,2064,2160,2224,2384↙
,2544,4560,4304,4384,4464,4816,4880,14352,14352,14352,14352,5744,4336,6160,6160,6160,6↙
512,6512,6928,6928,6928,11280,12592,13488,14768,14960,15760,15840,15920,12368,12368,12↙
368,12368,11664,17776,12464,13488,13232,10192,944,10848,17088,17344,4592,4592,11024,11↙
024,2384,2384,2384,2928,2928,2928,2544,2544,2544,9648,2048,2160,2224,3440,1744,1744,20↙
96,2096,3056,10960,2992,2992,10896,4080,16144,16144,14224,14224,14224,17104,17136,1544↙

```
0,15440,15440,15760,15856,4208,18320,15760,9712,17072,17072,16464,7856,7856,7856,7856,↙
16720,15504,4368,6,6064,4688,4816,4880,9008,15760,17424,18512,4880,4880,4880,4688,4752↙
,4816,18608,14416,14416,15760,18224,13424,18704,11088,11168,12464,15760,15760,15760,43↙
2,11168,13488,13584,13488,11184,4464,4528,4816,4816,11088,10672,10256,9840,4432,5808,1↙
552,5136,5552,5968,5488,16576,6096,6096,1824,1664,5936,11616,1952,6304,6368,16688,3488↙
,1392,1392,1408,11728,11728,6896,9232,8272,7024,14720,13072,15408,8080,10016,7776,1534↙
4,12400,12400,16496,16496,11936,11840,12672,3312,10544,11808,15536,18128,7600,11744,78↙
88,10992,10992,12,4992,5888,2704,16672,3408,3408,5760,6528,6944,7408,12608,12352,11648↙
,12416,12096,13248,912,912,928,14800,6640,6832,2944,9664,560,560,576,1728,5120,2080,30↙
72,2976,10848,8848,16112,17120,12160,13552];
                    y=↙
[8954,9856,11319,8624,7348,8294,7370,8184,10912,11440,572,2684,6908,7348,8624,308,2068↙
,2508,3080,3608,5302,7260,7260,9548,10582,11440,9064,9504,10252,11,8184,8184,8184,572,↙
1012,1452,2992,4818,6028,7348,7370,8184,10912,11440,308,308,308,308,308,308,308,308,30↙
80,3608,4664,5368,6072,7128,9658,11286,572,1452,2992,4708,6028,7348,8514,9284,9922,111↙
98,7480,9548,10582,11440,9152,10912,11440,6446,7590,8294,11440,11440,11440,11440,11440↙
,6336,6556,6556,6556,6556,6556,6556,6556,6556,6556,6556,6556,6556,6556,6556,6556,7260,↙
8140,9702,10582,11132,572,1452,2992,5874,7348,8096,8096,8096,8096,8404,8954,10142,1131↙
9,7370,8184,11440,4840,7700,8360,8360,8360,8360,8360,8360,8360,8844,9812,3608,4664,536↙
8,5918,6930,8954,9922,572,1452,2992,4818,6138,7348,7744,9548,10846,7040,7040,7040,308,↙
2068,2508,3080,3608,4268,5852,9064,9504,9922,11,8844,8844,6028,7348,8514,11440,8184,81↙
84,7370,7964,6446,8074,9548,10846,308,2068,2508,3080,3608,4048,4488,3608,4664,5368,591↙
8,6776,7920,7480,7920,7920,9152,9922,10472,10912,11440,748,3080,3608,4664,5368,5368,53↙
68,5368,5698,6336,7502,9064,10142,11022,5874,7348,9284,9922,11198,5192,5192,5192,572,2↙
992,10362,11286,308,968,2068,2508,3080,3608,4488,5588,6556,7744,6446,7436,8404,9042,10↙
912,11440,9878,9064,9064,9064,902,1980,2640,3168,3608,4158,5368,6556,6996,8008,8844,84↙
04,9152,10912,11440,6446,7590,9658,11022,8514,9284,9922,11198,5720,5720,5720,5720,572,↙
1760,3388,5258,6556,6996,7898,8602,9284,10208,10142,10142,10142,10142,10142,10142,1014↙
2,10142,968,3080,3608,4664,10912,10912,11286,6336,7436,8404,9152,10912,11440,8712,8932↙
,8932,10802,1232,3608,4180,4818,5808,5808,5808,5808,5808,5808,5808,10252,10252,10252,1↙
0252,10252,968,3080,3608,4664,9548,10472,11198,11198,11198,11198,11198,11198,11198,111↙
98,572,1232,3608,4180,4818,6138,7348,8228,9064,9504,10252,11,7920,8712,9372,10472,1091↙
2,11440,11440,7964,7964,7370,6446,7370,8294,4378,5478,6556,7480,8008,8602,9284,10318,1↙
1440,3608,3608,3608,3608,3608,3608,3608,3608,3608,3608,3608,3608,3608,7370,11440,968,9↙
68,968,968,968,968,968,968,968,968,308,2068,2508,3080,3608,4048,4488,5148,5148,5148,5148,5↙
148,5148,5148,5148,5148,5148,5148,5148,5148,5148,5148,6556,7480,8382,9284,11440,572,14↙
52,2728,7084,7084,9064,10142,572,1452,10032,10362,10692,10692,10692,10692,10692,10692,↙
10692,10692,10692,10692,10692,10692,10692,10692,10692,10692,10692,10692,10692,10692,10↙
692,9218,10032,10582,11132,3608,4664,5368,5918,6776,7436,308,308,308,308,308,308,308,3↙
08,308,308,308,308,308,308,308,308,308,308,1452,2288,3388,7260,8602,11440,7920,7920,7920,9↙
548,11440,6776,572,1452,2508,3388,7854,9174,9812,10362,2068,3080,3608,4664,5588,6556,6↙
82,3608,4048,4598,6138,7568,8228,9064,9504,10252,11,1012,1012,1012,1012,1012,1012,1012↙
,1012,1012,1012,1012,1012,572,6138,7568,8228,9064,9504,10252,10890,4928,4928,4928,4928↙
,4928,4928,4928,4928,4928,4928,4928,4928,4928,6028,6776,7788,8228,9504,10362,11110,470↙
8,4708,4708,4708,4708,8712,9284,9702,10472,10912,11440,8712,9482,9812,9812,9812,9812,8↙
844,8118,9394,11440,6556,6556,6556,6556,6556,6556,6556,6556,6556,6556,6556,6556,6556,6↙
556,6556,6556,6556,6556,6556,6556,6556,6556,6556,6556,1012,1980,2640,3608,4268,5368,57↙
2,572,572,572,572,572,572,572,572,572,572,1452,6006,2508,3080,3608,4664,5588,6556,7744↙
,9064,9812,10362,11286,8712,10472,10912,11440,9064,9064,2068,2508,3080,3608,4664,5588,↙
6556,7480,8712,9372,9372,10472,10912,11440,8448,8448,8448,8448,8448,8448,8448,8448,844↙
8,8448,8448,8448,8448,8448,572,1980,3718,5368,6996,7326,7700,7700,7700,7700,7700,2068,↙
2508,3080,3608,4268,7480,2728,3388,2288,2288,2288,2288,2288,2288,2288,2288,2288,572,12↙
32,1012,7920,7920,7920,7920,7920,7920,7920,7590,6116,6116,6116,6116,572,1496,5258,572,↙
1342,4488,5038,5588,6996,9284,9878,8844,3080,3080,3080,3080,3080,3080,3080,3080,3080,3↙
080,3080,3080,3080,3080,3080,3080,3080,2288,6116,2068,2508,3608,2068,2508,3608,572,299↙
```

```
2,7238,2948,2948,2948,2948,2948,2948,2948,572,1980,2728,7238,8184,8184,8184,9064,10032↙
,10032,10032,10032,10032,10032,10032,10032,10032,10032,10032,10032,10032,10032,10032,1↙
0032,3608,4268,6028,6776,8008,9504,6908,6908,7348,3388,2640,3168,8008,8008,8008,8008,8↙
008,8008,8008,8008,8008,8008,8008,8008,8008,2068,2508,3608,1452,1452,1452,1452,1452,14↙
52,1452,1452,1452,1452,1452,1452,1452,1452,572,3608,4048,6138,7568,8228,9064,9504,1025↙
2,10890,2068,2508,3608,4048,4488,6446,3608,4180,6138,7348,2068,3608,4774,5368,5368,536↙
8,9284,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,572,1452,2992,2992,↙
2992,2992,2992,2992,2992,572,3608,4180,7348,7788,9064,9504,10252,1452,9548,11440,572,5↙
72,3938,5258,9548,11440,8844,8844,8844,8844,8844,3608,4378,5368,6028,6776,6996,9284,57↙
2,3938,5258,572,11440,572,3938,5258,5368,10692,11132,1980,2640,3608,4378,5478,6028,602↙
8,6028,6028,6028,6028,6028,6028,6028,6028,10912,11440,308,2068,3608,4774,5698,6336,114↙
40,11440,11440,6996,4158,5808,6336,6556,6556,6556,2288,3608,9504,9504,9504,8228,8228,8↙
228,8228,8228,8228,8228,3608,4180,7348,572,572,572,572,572,572,572,572,572,572,572,393↙
8,5258,6996,9284,3938,5258,9284,4048,4048,2288,3608,4268,2068,2508,3608,4488,4268,572,↙
6996,4488,2288,2288,572,5258,6996,6996,7348,7788,9064,9504,10252,2068,3608,4664,5808,1↙
1198,4488,4488,572,5038,6996,572,2068,6996,8712,9152,10472,10912,11440,9284,10472,1342↙
,3608,4048,6776,7678,9504,10472,10472,10472,10032,7370,1452,7348,9064,9504,9922,9922,9↙
922,9922,9922,10912,10912,10912,10912,10912,8712,9152,10912,10912,10912,10912,9504,778↙
8,7788,7788,1342,3608,4048,8404,8404,8404,8404,8404,8404,8404,7546,1342,3608,6776,7568↙
,4048,4048,1980,3608,4378,5368,1452,6776,1452,7348,6336,6336,6336,6336,6336,6336,7370,↙
7260,7260,7260,7260,7260,1342,3608,9504,10252,8844,11440,3608,4664,11132,3608,4664,206↙
8,3608,4664,6996,6996,3608,3608,3608,3608,3608,7348,572,1232,3938,5258,6996,2684,3938,↙
4268,6666,572,7370,9284,9284,9284,4840,6028,572,5038,7436,9152,11440,8712,9152,11440,7↙
546,9152,11440,7480,7172,7172,7172,7172,9042,11440,9152,11440,11440,6996,8712,9152,699↙
6,11440,9064,9504,6776,9504,10252,8624,7348,9064,9504,10252,7788,1452,4840,7348,4180,7↙
480,1452,7348,9504,2068,2508,3608,4488,7348,9064,4840,11132,11132,5808,5808,5808,3608,↙
9064,9064,8294,4664,5368,6776,4488,4488,4488,8866,9504,10252,7348,7348,6776,8866,4928,↙
4928,5368,1452,9504,10252,7480,6996,6776,9504,5368,572,4840,4840,5368,6776,572,572,572↙
,572,220,220,8932,11528,11528,11528,7304,8184,264,594,6864,8184,330,8228,8184,8536,884↙
4,11407,7480,2904,5852,6028,11330,10868,220,220,8756,7260,4928,484,484,7700,9064,8448,↙
7788,11484,11176,11484,11176,6776,6336,6776,10692,11528,8228,484,352,6116,8228,9064,11↙
550,11308,6776,9812,5060,8250,6248,3058,6160,8624,8844,7260,10912,3608,6116,6336,6116,↙
9504,9284,3410,6864,7040,8008,220,220,6952,7788,3410,6864,7040,8184,7920,8932,11220,10↙
692,6776,9064,8844,8404,9064,11418];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'St70';
                    obj.dim=70;
                    obj.optima=675;
                    x=↙
[64,80,69,72,48,58,81,79,30,42,7,29,78,64,95,57,40,68,92,62,28,76,67,93,6,87,30,77,78,↙
55,82,73,20,27,95,67,48,75,8,20,54,63,44,52,12,25,58,5,90,41,25,37,56,10,98,16,89,48,8↙
1,29,17,5,79,9,17,74,10,48,83,84];
                    y=↙
[96,39,23,42,67,43,34,17,23,67,76,51,92,8,57,91,35,40,34,1,43,73,88,54,8,18,9,13,94,3,↙
88,28,55,43,86,99,83,81,19,18,38,36,33,18,13,5,85,67,9,76,76,64,63,55,7,74,60,82,76,60↙
,22,45,70,100,82,67,68,19,86,94];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));
                    obj.x=x;
                    obj.y=y;
                    obj.D=D;
                case 'Ts225';
```

```
                    obj.dim=225;
                    obj.optima=126643;
                    x=↙
[4000,4000,4000,4000,4000,4000,4000,4000,4000,4000,4000,4000,4000,4000,4000,4000,↙
4000,4000,4000,4000,4000,4000,4000,4000,7000,7000,7000,7000,7000,7000,7000,7000,7000,7↙
000,7000,7000,7000,7000,7000,7000,7000,7000,7000,7000,7000,7000,7000,7000,7000,10000,1↙
0000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,100↙
00,10000,10000,10000,10000,10000,10000,10000,10000,10000,13000,13000,13000,13000,13000↙
,13000,13000,13000,13000,13000,13000,13000,13000,13000,13000,13000,13000,13000,13000,1↙
3000,13000,13000,13000,13000,13000,16000,16000,16000,16000,16000,16000,16000,16000,160↙
00,16000,16000,16000,16000,16000,16000,16000,16000,16000,16000,16000,16000,16000,16000↙
,16000,16000,4500,5000,5500,6000,6500,4500,5000,5500,6000,6500,4500,5000,5500,6000,650↙
0,4500,5000,5500,6000,6500,4500,5000,5500,6000,6500,7500,8000,8500,9000,9500,7500,8000↙
,8500,9000,9500,7500,8000,8500,9000,9500,7500,8000,8500,9000,9500,7500,8000,8500,9000,↙
9500,10500,11000,11500,12000,12500,10500,11000,11500,12000,12500,10500,11000,11500,120↙
00,12500,10500,11000,11500,12000,12500,10500,11000,11500,12000,12500,13500,14000,14500↙
,15000,15500,13500,14000,14500,15000,15500,13500,14000,14500,15000,15500,13500,14000,1↙
4500,15000,15500,13500,14000,14500,15000,15500];
                    y=↙
[4000,4500,5000,5500,6000,6500,7000,7500,8000,8500,9000,9500,10000,10500,11000,11500,1↙
2000,12500,13000,13500,14000,14500,15000,15500,16000,4000,4500,5000,5500,6000,6500,700↙
0,7500,8000,8500,9000,9500,10000,10500,11000,11500,12000,12500,13000,13500,14000,14500↙
,15000,15500,16000,4000,4500,5000,5500,6000,6500,7000,7500,8000,8500,9000,9500,10000,1↙
0500,11000,11500,12000,12500,13000,13500,14000,14500,15000,15500,16000,4000,4500,5000,↙
5500,6000,6500,7000,7500,8000,8500,9000,9500,10000,10500,11000,11500,12000,12500,13000↙
,13500,14000,14500,15000,15500,16000,4000,4500,5000,5500,6000,6500,7000,7500,8000,8500↙
,9000,9500,10000,10500,11000,11500,12000,12500,13000,13500,14000,14500,15000,15500,160↙
00,4000,4000,4000,4000,4000,7000,7000,7000,7000,7000,10000,10000,10000,10000,10000,130↙
00,13000,13000,13000,13000,16000,16000,16000,16000,16000,4000,4000,4000,4000,4000,7000↙
,7000,7000,7000,7000,10000,10000,10000,10000,10000,13000,13000,13000,13000,13000,16000↙
,16000,16000,16000,16000,4000,4000,4000,4000,4000,7000,7000,7000,7000,7000,10000,10000↙
,10000,10000,10000,13000,13000,13000,13000,13000,16000,16000,16000,16000,16000,4000,40↙
00,4000,4000,4000,7000,7000,7000,7000,7000,10000,10000,10000,10000,10000,13000,13000,1↙
3000,13000,13000,16000,16000,16000,16000,16000];
                    T(:,1)=x;T(:,2)=y;
                    D=round(pdist2(T,T));
                    obj.x=x;
                    obj.y=y;
                    obj.D=D;

                otherwise,
                  disp('fitness function not defined');
               end
        end

        function fitness = evaluation(obj, tour)
            global eval;
            eval = eval+1;

            fitness = TourLength_0(tour,obj.D);

        end
    end
end
```

Appendix H

MATLAB code of Combinatorial Bees Algorithm with BNSN using seed (Domino Sequence Heuristic) for PCB insertion sequence optimisation

```
clc;
close all;

model = IntCreateModel_C_PCB();
nFeed = 25;
A = zeros(nFeed, model.n);
for i = 1:nFeed
 A(i,:)  = DominoAlgo( 2, model.D,model.n);

while A(i,end) ~= 1 % because comp 1 is the min in number

    A(i,:) = circshift(A(i,:),[0,1]);


end
%A

end

for j = 1:nFeed
    BestSol(j)  = BA_PCB_BNSN( A(j,:) );
end

BestSol
```

```
function [ JL ] = DominoAlgo( pl,y,noc )
%% Shuffle the dominoes
c = randperm (noc);                     % The collection of shuffled tiles is often↙
called the "bone yard".
%% Draw an opening hand
sc = vec2mat (c, ceil(noc/pl));         % Take 1/pl (1/2 or half for 2 players) from↙
bone yard. 7 tiles each player for the original game rule. The first player's tile set↙
is in the first row of the sc matrix and so on. (sc) matrix represent the sets of↙
players' card.
%% Decide the order of play
I=randi(noc);
y2=[zeros(noc,1) y];                    % Generate the dummy matrix to assist the↙
calculation.
y2=[zeros(1,noc+1); y2];                % Since the selected tile will be "0" value↙
then the (y) matrix should added with 1 zeros row and column in upper and left edge of↙
the matrix.
%% Lay the first domino
E=I;                                    % The sequence is devided to 2 section so it↙
can be fitted for Assymetric TSP problem.
T=I;                                    % The Early section called (E) and the Tardy↙
called (T).
sc(sc==I)=0;                            % Replace the selected first tile in the↙
matrix with "0".
%% Take turn adding the dominoes
while sum(sum(sc))>0                                % The turn will be stop if there↙
is not a tile anymore.
         for i=1:pl
            Etem.d=sc(i,:);                         % Generate the vector of the↙
destination from player 1's set of tiles for calculate (E)'s path.
            Etem.o=zeros(1,ceil(noc/pl))+E(end);    % Generate the vector of the↙
destination from the end side of (E) --> E(end).
            for j=1:ceil(noc/pl)
            Etem.p(j)=y2(Etem.d(j)+1,Etem.o(j)+1);  % Calculate the (E)'s path from↙
(y2) matrix.
            end
            Ttem.d=Etem.d;                          % Generate the vector of the↙
destination from player 1's set of tiles for evaluate (T)'s path.
            Ttem.o=zeros(1,ceil(noc/pl))+T(1);      % Generate the vector of the↙
destination from the first side of (T)--> T(1).
            for k=1:ceil(noc/pl)
            Ttem.p(k)=y2(Ttem.d(k)+1,Ttem.o(k)+1);  % Calculate the (T)'s path from↙
(y2) matrix.
            end
            ValE=min(Etem.p(Etem.p>0));             % Evaluate the (E)'s path
            if isempty(ValE)
                continue
            end
            Etemp = Etem.d(find(Etem.p==ValE));
            Etemp = Etemp(1);
            ValT=min(Ttem.p(Ttem.p>0));             % Evaluate the (T)'s path
            if isempty(ValT)
                continue
            end
            Ttemp = Ttem.d(find(Ttem.p==ValT));
            Ttemp = Ttemp(1);
```

```
            if ValE <= ValT
                E=[E Etemp];                        % Adding the domino to the (E) or↙
(T)
                sc(sc==Etemp)=0;
            else
                T=[Ttemp T];
                sc(sc==Ttemp)=0;
            end
            clearvars Etem Ttem;
        end
end
%% End the round
T(end)=[];
JL=[E T];                                   % Combine the (E) and (T) as the↙
sequence
```

```
function [ BestSol ] = BA_PCB_BNSN( A )
DataModel= Model_PCB2_r(A);
CostFunction=@(tour) TourLength_PCB(tour,DataModel);    % Cost Function
nVar= DataModel.n;      % Number of Decision Variables
VarSize=[1 nVar];       % Decision Variables Matrix Size
MaxIt=999;              % Maximum Number of Iterations
n=50;                   % Number of Scout Bees
m=14;                   % Number of Potential Sites
e=5;                    % Number of Best Sites
nsp=25;                 % Number of Recruited Bees for Potential Sites
nep=50;                 % Number of Recruited Bees for Best Sites

empty_bee.Position=[];
empty_bee.Cost=[];
Patch=repmat(empty_bee,n,1);

for i=1:1000
    Patch(i).Position= BNSN_0(randi(nVar),randi(3),rand(),DataModel);
    Patch(i).Cost=CostFunction(Patch(i).Position);
end

[~, SortOrder]=sort([Patch.Cost]);
Patch=Patch(1:1000/n:1000);
BestSol=Patch(1);

BestCost=zeros(MaxIt,1);
%% Main Loop
for it=1:MaxIt
    for i=1:e
        bestnewbee.Cost=inf;
        for j=1:nep
            newbee.Position= Foraging(Patch(i).Position); %
            newbee.Cost=CostFunction(newbee.Position);
            if newbee.Cost<bestnewbee.Cost
                bestnewbee=newbee;
            end
        end
        if bestnewbee.Cost<Patch(i).Cost
            Patch(i)=bestnewbee;
        end
    end
    for i=e+1:m
        bestnewbee.Cost=inf;
        for j=1:nsp
            newbee.Position= Foraging(Patch(i).Position); %
            newbee.Cost=CostFunction(newbee.Position);
            if newbee.Cost<bestnewbee.Cost
                bestnewbee=newbee;
            end
        end
        if bestnewbee.Cost<Patch(i).Cost
            Patch(i)=bestnewbee;
        end
    end
    for i=m+1:n
```

```
        Patch(i).Position= Foraging(Patch(i).Position);
        Patch(i).Cost=CostFunction(Patch(i).Position);
    end
    % Sort
    [~, SortOrder]=sort([Patch.Cost]);
    Patch=Patch(SortOrder);
    % Update Best Solution Ever Found
    BestSol=Patch(1);
    BestCost(it)=BestSol.Cost;
    disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(BestCost(it))]);
    figure(1);
    PlotSolution(BestSol.Position,DataModel);
    t = text(0,0,['feeder arrangement: ' num2str(A) ' --> Fittness: ' num2str(BestSol.↙
Cost) ' seconds']);
    pause(0.01);
end
end
```

```
function L=TourLength_PCB(tour,model)

    n=numel(tour);

    tour=[tour tour(1) tour(2)];

    L=0;
    nct = numel(model.ct);
    for i = 1:nct
        FN(i) = find(model.A==model.ct(tour(i)));
    end

    FN=[FN FN(1) FN(2)];

    for i=1:n
        L=L+(max([model.D1(tour(i),tour(i+1)), model.D2(FN(i+1),FN(i+2)), model.D3
(tour(i),tour(i+1))]));
    end

end
```

```
function p = BNSN_0 (s,m,t,model)
x=model.x;
y=model.y;
D=model.D;
n = size(D,1);
p = zeros(1,n,'uint16');

p(1) = s;
D(s,:) = inf;

%m=2;
for k = 2:2
    D(s,:) = inf;
    [junk,s] = min(D(:,s)); %#ok
    p(k) = s;

end
for k = 3:n    %-(m-1)

    D(s,:) = inf;
    Temp=D(:,s);

    %m=5;

    R=cell(m,1);
    PQ_PR=cell(m,1);
    PRd=cell(m,1);
    angd=cell(m,1);
    valZ=zeros(m,1);

    val = zeros(m,1);
    idx = zeros(m,1);
    for i=1:m
      [val(i),idx(i)] = min(Temp);
      % remove for the next iteration the last smallest value:
      Temp(idx(i)) = inf;
    end
    if ((val(end)-val(1))/val(1)) <= t %*rand()
        Q=[x(p(k-2)) y(p(k-2))];
        P=[x(p(k-1)) y(p(k-1))];
        PQd= norm(P-Q);
        for a= 1:m
            R{a}=[x(idx(a)) y(idx(a))];
            PQ_PR{a}=sum((Q-P).*(R{a}-P));
            PRd{a}= norm(P-R{a});
            angd{a}=acosd(PQ_PR{a}/(PQd*PRd{a}));
            valZ(a)=(1+(2-(angd{a}/90)))*PRd{a};
        end

        [~,idz] = min(valZ);
        s=idx(idz);

        p(k) = s;
    else
        [junk,s] = min(D(:,s)); %#ok
```

```
        p(k) = s;
    end


end
```

```
function NewPatch = Foraging(sol)

    m=randi([1 3]);

    switch m
        case 1
            NewPatch = Swap2(sol);
        case 2
            NewPatch = Reverse2(sol);
        case 3
            NewPatch = Insert2(sol);
    end
end

function NewPatch = Swap2(sol)
    n=numel(sol);
    i=randsample(n,2);
    i1=i(1);
    i2=i(2);
    NewPatch=sol;
    NewPatch([i1 i2])=sol([i2 i1]);
end

function NewPatch = Reverse2(sol)
    n=numel(sol);
    i=randsample(n,2);
    i1=min(i(1),i(2));
    i2=max(i(1),i(2));
    NewPatch = sol;
    NewPatch(i1:i2)=sol(i2:-1:i1);
end

function NewPatch = Insert2(sol)
    n=numel(sol);
    i=randsample(n,2);
    i1=i(1);
    i2=i(2);
    if i1<i2
        NewPatch = [sol(1:i1-1) sol(i1+1:i2) sol(i1) sol(i2+1:end)];
    else
        NewPatch = [sol(1:i2) sol(i1) sol(i2+1:i1-1) sol(i1+1:end)];
    end
end
```

```
function model=IntCreateModel_C_PCB()

    x=[240  160 150 162.8571429 168 153.3333333 208 200 180 180];

    y=[120  132.5   115 154.2857143 142 130 196 130 112.5   130];

    n=numel(x);

    D=zeros(n,n);

    for i=1:n-1
        for j=i+1:n

            D(i,j)=round(sqrt((x(i)-x(j))^2+(y(i)-y(j))^2));

            D(j,i)=D(i,j);

        end
    end

    model.n=n;
    model.x=x;
    model.y=y;
    model.D=D;

end
```

```
function model = Model_PCB2_r( FAct )
v = 60;
nF = numel(FAct);
FAno = 1:nF;

xf = [300   300 300 300 300 300 300 300 300 300];
yf = [15    30  45  60  75  90  105 120 135 150];

x = [100    120 100 140 160 120 100 120 100 140 120 140 160 180 200 220 240 240 240
240 220 200 160 180 200 200 240 240 220 240 220 200 180 160 140 120 140 180 200 220
240 240 240 220 200 180 160 140 120 100];
y = [90 130 130 140 140 150 180 230 230 220 190 180 220 220 180 220 220 210 200 180
200 220 180 180 170 140 80  60  60  40  40  60  60  60  40  90  100 140 130 160 140
120 100 100 100 100 100 80  50  60];
ct = [3 6   2   4   5   9   10  9   4   7   5   10  9   8   10  5   2   7   6   7   7
4   4   4   7   3   4   9   9   10  9   8   9   5   9   10  9   8   9   9   10  1   8
10  9   5   2   2   4   6];

n = numel(x);

p = [1  2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  19  20  21
22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43
44  45  46  47  48  49  50];


nct = numel(ct);
for i = 1:nct
    FN(i) = find(FAct==ct(p(i)));
    yfi(i) = yf(FN(i));
    t3(i) = 0.25;
end
yfj= circshift(yfi,[0,-1]);

D1=zeros(n,n);
D2=zeros(nF,nF);
D3=zeros(n,n);
Dm=zeros(n,n);
    for i=1:n-1
        for j=i+1:n
            D(i,j)=sqrt((x(i)-x(j))^2+(y(i)-y(j))^2);
            D1(i,j) = max([abs((x(j)-x(i))/v) abs((y(j)-y(i))/v)]);
            D3(i,j) = 0.25;
            D(j,i)=D(i,j);
            D1(j,i)=D1(i,j);
            D3(j,i)=D3(i,j);
         end
    end
    for k = 1:nF-1
        for l = k+1:nF
            D2(k,l) = (((abs(yf(k)-yf(l)))^2)^0.5)/v;
            D2(l,k)=D2(k,l);
        end
    end

    model.n = n;
```

```
    model.A = FAct;
    model.ct = ct;
    model.x = x;
    model.y = y;
    model.D = D;
    model.D1 = D1;
    model.D2 = D2;
    model.D3 = D3;
    model.yfi = yfi;
    model.yfj = yfj;
    model.FN = FN;
end
```

Appendix I

MATLAB code of Combinatorial Bees Algorithm with BNSN without seed for PCB insertion sequence optimisation

```
clc;
clear;
close all;

%% Problem Definition
A=randperm(10);
Instance= Model_PCB2_r(A);
model = IntCreateModel_C_PCB();
Dims=Instance.n;
VarSize=[1 Dims];
%% Bees Algorithm Parameters
n= 50; nep = n; m=14; e=5; nsp=0.5*n; ngh=0.1;
stlim1=100; stlim2=10;
accuracy=0.005; MaxEval = 500000; MaxIt=10000;
ColonySize=(e*nep)+((m-e)*nsp)+(n-m);
%% Initialization
Empty_Patch.A=[]; Empty_Patch.Model=[]; Empty_Patch.Position=[];
Empty_Patch.Cost=[]; Empty_Patch.Stagnated =[]; Empty_Patch.Counter=[];
Patch=repmat(Empty_Patch,n,1);
counter=0;
% Create Initial Solutions
for i=1:n
    A = DominoAlgo( 2, model.D,model.n);
    while A(end) ~= 1 % because component 1 is the min in number
        A = circshift(A,[0,1]);
    end
    Patch(i).A = A;
    Patch(i).Model= Model_PCB2_r(Patch(i).A);
    Patch(i).Position= BNSN_0(randi(Dims),randi(3),rand(),Instance);
    Patch(i).Cost= TourLength_PCB(Patch(i).Position,Instance);
    Patch(i).Stagnated = 0;
    counter = counter + 1;
    Patch(i).Counter = counter;
end
% Sort
[~, SortOrder]=sort([Patch.Cost]);
Patch=Patch(SortOrder);
% Update Best Solution Ever Found
BestSol=Patch(1);
% Array to Hold Best Cost Values
BestCost=zeros(MaxIt,1);
Counter=zeros(MaxIt,1);
OptSol.Cost=inf;
%% Bees Algorithm Main Loop
P=1;
for it=1:MaxIt
    if counter >= MaxEval
        break;
    end
    for i=1:e % ELITE SITES
        BestForager.Cost=inf;
        for j=1:nep
            ForagerBees.A = Patch(i).A;
            ForagerBees.Model = Patch(i).Model;
            ForagerBees.Position= Foraging(Patch(i).Position);
```

```
            ForagerBees.Cost = TourLength_PCB(ForagerBees.Position,ForagerBees.Model);
            ForagerBees.Stagnated = Patch(i).Stagnated;
            counter = counter + 1;
            ForagerBees.Counter = counter;
            if ForagerBees.Cost<BestForager.Cost
                BestForager=ForagerBees;
            end
        end
        if BestForager.Cost<Patch(i).Cost
            Patch(i)=BestForager;
            Patch(i).Stagnated=0;
        else
            Patch(i).Stagnated=Patch(i).Stagnated+1;
        end
        %site abandonment procedure
        if(Patch(i).Stagnated>stlim1)
            A = Foraging_A (A);
            while A(end) ~= 1 % because comp 1 is the min in number
                A = circshift(A,[0,1]);
            end
            Patch(i).A = A;
            Patch(i).Size=range;
            Patch(i).Stagnated=0;
            counter = counter + 1;
            Patch(i).Counter = counter;
        end
    end
    for i=e+1:m % SELECTED SITES
        BestForager.Cost=inf;
        for j=1:nsp
            ForagerBees.A = Patch(i).A;
            ForagerBees.Model = Patch(i).Model;
            ForagerBees.Position= Foraging(Patch(i).Position);
            ForagerBees.Cost = TourLength_PCB(ForagerBees.Position,ForagerBees.Model);
            ForagerBees.Stagnated = Patch(i).Stagnated;
            counter = counter + 1;
            ForagerBees.Counter = counter;
            if ForagerBees.Cost<BestForager.Cost
                BestForager=ForagerBees;
            end
        end
        if BestForager.Cost<Patch(i).Cost
            Patch(i)=BestForager;
            Patch(i).Stagnated=0;
        else
            Patch(i).Stagnated=Patch(i).Stagnated+1;
        end
        %site abandonment procedure
        if(Patch(i).Stagnated>stlim2)
            A = Foraging (A);
            while A(end) ~= 1 % because comp 1 is the min in number
                A = circshift(A,[0,1]);
            end
            Patch(i).A = A;
            Patch(i).Stagnated=0;
```

```
            counter = counter + 1;
            Patch(i).Counter = counter;
        end
    end
    for i=m+1:n % NON-SELECTED SITES
        A = DominoAlgo( 2, model.D,model.n);
        while A(end) ~= 1 % because comp 1 is the min in number
            A = circshift(A,[0,1]);
        end
        Patch(i).A = A;
        Patch(i).Model = Patch(i).Model;
        Patch(i).Position= BNSN_0(randi(Dims),randi(2),rand(),Instance);
        Patch(i).Cost = TourLength_PCB(Patch(i).Position,Patch(i).Model);
        Patch(i).Stagnated = 0;
        counter = counter + 1;
        Patch(i).Counter = counter;
    end
    % Sort
    [~, SortOrder]=sort([Patch.Cost]);
    Patch=Patch(SortOrder);
    % Update Best Solution Ever Found
    BestSol=Patch(1);
    if BestSol.Cost < OptSol.Cost
        OptSol=BestSol;
    end
    % Store Best Cost Ever Found
    BestCost(it)=OptSol.Cost;
    Counter(it)=OptSol.Counter;
    % Display Iteration Information
    disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(BestCost(it)) ': Best Cost↙
= ' num2str(Counter(it))]);
    figure(1);
    PlotSolution(OptSol.Position,OptSol.Model);
    t = text(0,0,['feeder arrangement: ' num2str(OptSol.A) ' --> Fittness: ' num2str↙
(OptSol.Cost) ' seconds']);
    pause(0.01);
end
%% Results
figure(1);
PlotSolution(BestSol.Position,Instance);
figure;
semilogy(BestCost,'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
```

```
function p = BNSN_0 (s,m,t,model)
x=model.x;
y=model.y;
D=model.D;
n = size(D,1);
p = zeros(1,n,'uint16');

p(1) = s;
D(s,:) = inf;

%m=2;
for k = 2:2
    D(s,:) = inf;
    [junk,s] = min(D(:,s)); %#ok
    p(k) = s;

end
for k = 3:n    %-(m-1)

    D(s,:) = inf;
    Temp=D(:,s);

    %m=5;

    R=cell(m,1);
    PQ_PR=cell(m,1);
    PRd=cell(m,1);
    angd=cell(m,1);
    valZ=zeros(m,1);

    val = zeros(m,1);
    idx = zeros(m,1);
    for i=1:m
      [val(i),idx(i)] = min(Temp);
      % remove for the next iteration the last smallest value:
      Temp(idx(i)) = inf;
    end
    if ((val(end)-val(1))/val(1)) <= t %*rand()
        Q=[x(p(k-2)) y(p(k-2))];
        P=[x(p(k-1)) y(p(k-1))];
        PQd= norm(P-Q);
        for a= 1:m
            R{a}=[x(idx(a)) y(idx(a))];
            PQ_PR{a}=sum((Q-P).*(R{a}-P));
            PRd{a}= norm(P-R{a});
            angd{a}=acosd(PQ_PR{a}/(PQd*PRd{a}));
            valZ(a)=(1+(2-(angd{a}/90)))*PRd{a};
        end

        [~,idz] = min(valZ);
        s=idx(idz);

        p(k) = s;
    else
        [junk,s] = min(D(:,s)); %#ok
```

```
        p(k) = s;
    end


end
```

```
function [ JL ] = DominoAlgo( pl,y,noc )
%% Shuffle the dominoes
c = randperm (noc);                     % The collection of shuffled tiles is often↙
called the "bone yard".
%% Draw an opening hand
sc = vec2mat (c, ceil(noc/pl));         % Take 1/pl (1/2 or half for 2 players) from↙
bone yard. 7 tiles each player for the original game rule. The first player's tile set↙
is in the first row of the sc matrix and so on. (sc) matrix represent the sets of↙
players' card.
%% Decide the order of play
I=randi(noc);
y2=[zeros(noc,1) y];                    % Generate the dummy matrix to assist the↙
calculation.
y2=[zeros(1,noc+1); y2];                % Since the selected tile will be "0" value↙
then the (y) matrix should added with 1 zeros row and column in upper and left edge of↙
the matrix.
%% Lay the first domino
E=I;                                    % The sequence is devided to 2 section so it↙
can be fitted for Assymetric TSP problem.
T=I;                                    % The Early section called (E) and the Tardy↙
called (T).
sc(sc==I)=0;                            % Replace the selected first tile in the↙
matrix with "0".
%% Take turn adding the dominoes
while sum(sum(sc))>0                                % The turn will be stop if there↙
is not a tile anymore.
         for i=1:pl
            Etem.d=sc(i,:);                         % Generate the vector of the↙
destination from player 1's set of tiles for calculate (E)'s path.
            Etem.o=zeros(1,ceil(noc/pl))+E(end);    % Generate the vector of the↙
destination from the end side of (E) --> E(end).
            for j=1:ceil(noc/pl)
            Etem.p(j)=y2(Etem.d(j)+1,Etem.o(j)+1);  % Calculate the (E)'s path from↙
(y2) matrix.
            end
            Ttem.d=Etem.d;                          % Generate the vector of the↙
destination from player 1's set of tiles for evaluate (T)'s path.
            Ttem.o=zeros(1,ceil(noc/pl))+T(1);      % Generate the vector of the↙
destination from the first side of (T)--> T(1).
            for k=1:ceil(noc/pl)
            Ttem.p(k)=y2(Ttem.d(k)+1,Ttem.o(k)+1);  % Calculate the (T)'s path from↙
(y2) matrix.
            end
            ValE=min(Etem.p(Etem.p>0));             % Evaluate the (E)'s path
            if isempty(ValE)
                continue
            end
            Etemp = Etem.d(find(Etem.p==ValE));
            Etemp = Etemp(1);
            ValT=min(Ttem.p(Ttem.p>0));             % Evaluate the (T)'s path
            if isempty(ValT)
                continue
            end
            Ttemp = Ttem.d(find(Ttem.p==ValT));
            Ttemp = Ttemp(1);
```

```
            if ValE <= ValT
                E=[E Etemp];                        % Adding the domino to the (E) or↙
(T)
                sc(sc==Etemp)=0;
            else
                T=[Ttemp T];
                sc(sc==Ttemp)=0;
            end
            clearvars Etem Ttem;
        end
end
%% End the round
T(end)=[];
JL=[E T];                                   % Combine the (E) and (T) as the↙
sequence
```

```
function L=TourLength_PCB(tour,model)

    n=numel(tour);

    tour=[tour tour(1) tour(2)];

    L=0;
    nct = numel(model.ct);
    for i = 1:nct
        FN(i) = find(model.A==model.ct(tour(i)));
    end

    FN=[FN FN(1) FN(2)];

    for i=1:n
        L=L+(max([model.D1(tour(i),tour(i+1)), model.D2(FN(i+1),FN(i+2)), model.D3↙
(tour(i),tour(i+1))]));
    end

end
```

```
function NewPatch = Foraging(sol)

    m=randi([1 3]);

    switch m
        case 1
            NewPatch = Swap2(sol);
        case 2
            NewPatch = Reverse2(sol);
        case 3
            NewPatch = Insert2(sol);
    end
end

function NewPatch = Swap2(sol)
    n=numel(sol);
    i=randsample(n,2);
    i1=i(1);
    i2=i(2);
    NewPatch=sol;
    NewPatch([i1 i2])=sol([i2 i1]);
end

function NewPatch = Reverse2(sol)
    n=numel(sol);
    i=randsample(n,2);
    i1=min(i(1),i(2));
    i2=max(i(1),i(2));
    NewPatch = sol;
    NewPatch(i1:i2)=sol(i2:-1:i1);
end

function NewPatch = Insert2(sol)
    n=numel(sol);
    i=randsample(n,2);
    i1=i(1);
    i2=i(2);
    if i1<i2
        NewPatch = [sol(1:i1-1) sol(i1+1:i2) sol(i1) sol(i2+1:end)];
    else
        NewPatch = [sol(1:i2) sol(i1) sol(i2+1:i1-1) sol(i1+1:end)];
    end
end
```

```
function PlotSolution(tour,model)

    tour=[tour tour(1)];

    plot(model.x(tour),model.y(tour),'k-o',...
        'MarkerSize',3,...
        'MarkerFaceColor','y',...
        'LineWidth',0.5);

    axis equal;

    alpha = 0.1;

    xmin = min(model.x);
    xmax = max(model.x);
    dx = xmax - xmin;
    xmin = floor((xmin - alpha*dx)/10)*10;
    xmax = ceil((xmax + alpha*dx)/10)*10;
    xlim([xmin xmax]);

    ymin = min(model.y);
    ymax = max(model.y);
    dy = ymax - ymin;
    ymin = floor((ymin - alpha*dy)/10)*10;
    ymax = ceil((ymax + alpha*dy)/10)*10;
    ylim([ymin ymax]);


end
```

```
function model = Model_PCB2_r( FAct )
v = 60;
nF = numel(FAct);
FAno = 1:nF;

xf = [300   300 300 300 300 300 300 300 300 300];
yf = [15    30  45  60  75  90  105 120 135 150];

x = [100    120 100 140 160 120 100 120 100 140 120 140 160 180 200 220 240 240 240
240 220 200 160 180 200 200 240 240 220 240 220 200 180 160 140 120 140 180 200 220
240 240 240 220 200 180 160 140 120 100];
y = [90 130 130 140 140 150 180 230 230 220 190 180 220 220 180 220 220 210 200 180
200 220 180 180 170 140 80  60  60  40  40  60  60  60  40  90  100 140 130 160 140
120 100 100 100 100 100 80  50  60];
ct = [3 6   2   4   5   9   10  9   4   7   5   10  9   8   10  5   2   7   6   7   7
4   4   4   7   3   4   9   9   10  9   8   9   5   9   10  9   8   9   9   10  1   8
10  9   5   2   2   4   6];

n = numel(x);

p = [1  2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  19  20  21
22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43
44  45  46  47  48  49  50];


nct = numel(ct);
for i = 1:nct
    FN(i) = find(FAct==ct(p(i)));
    yfi(i) = yf(FN(i));
    t3(i) = 0.25;
end
yfj= circshift(yfi,[0,-1]);

D1=zeros(n,n);
D2=zeros(nF,nF);
D3=zeros(n,n);
Dm=zeros(n,n);
    for i=1:n-1
        for j=i+1:n
            D(i,j)=sqrt((x(i)-x(j))^2+(y(i)-y(j))^2);
            D1(i,j) = max([abs((x(j)-x(i))/v) abs((y(j)-y(i))/v)]);
            D3(i,j) = 0.25;
            D(j,i)=D(i,j);
            D1(j,i)=D1(i,j);
            D3(j,i)=D3(i,j);
         end
    end
    for k = 1:nF-1
        for l = k+1:nF
            D2(k,l) = (((abs(yf(k)-yf(l)))^2)^0.5)/v;
            D2(l,k)=D2(k,l);
        end
    end

    model.n = n;
```

```
    model.A = FAct;
    model.ct = ct;
    model.x = x;
    model.y = y;
    model.D = D;
    model.D1 = D1;
    model.D2 = D2;
    model.D3 = D3;
    model.yfi = yfi;
    model.yfj = yfj;
    model.FN = FN;
end
```

```
function model=IntCreateModel_C_PCB()

    x=[240  160 150 162.8571429 168 153.3333333 208 200 180 180];

    y=[120  132.5   115 154.2857143 142 130 196 130 112.5   130];

    n=numel(x);

    D=zeros(n,n);

    for i=1:n-1
        for j=i+1:n

            D(i,j)=round(sqrt((x(i)-x(j))^2+(y(i)-y(j))^2));

            D(j,i)=D(i,j);

        end
    end

    model.n=n;
    model.x=x;
    model.y=y;
    model.D=D;

end
```

Appendix J

MATLAB Code of Combinatorial Bees Algorithm with Bi-BA (Clustering) and BRO for capacitated vehicle routing problem

```
clc;
clear;
close all;
%% Problem Definition
[typeOfFunction] = 'EilD76';
Instance=TsplibVRP(typeOfFunction);
Dims=Instance.dim;
ObjFunction=@(x) Instance.evaluation( x );
VarSize=[1 Dims];
%% Bees Algorithm Parameters
n= 10; nep = 40; m=5; e=2; nsp=20; stlim=Instance.J*2;
ColonySize=(e*nep)+((m-e)*nsp)+(n-m);
MaxIt=3000; MaxEval = 1000000;
range=ceil(log10(Dims));
ngh=10^-(range-1);
%% Initialization
Empty_Patch.Position=[]; Empty_Patch.Cost=[]; Empty_Patch.Sol=[];
Empty_Patch.Stagnated =[]; Empty_Patch.Counter=[];
Patch=repmat(Empty_Patch,n,1);
counter=0;
for i=1:n
    [Patch(i).Position,Patch(i).Cost, Patch(i).Sol]=BiBA_Clustering_VRP(Instance);
    Patch(i).Stagnated = 0;
    counter = counter + 1;
    Patch(i).Counter = counter;
end
% Sort
[~, SortOrder]=sort([Patch.Cost]);
Patch=Patch(SortOrder);
BestSol=Patch(1);
% Array to Hold Best Cost Values
BestCost=zeros(MaxIt,1);
Counter=zeros(MaxIt,1);
OptSol.Cost=inf;
%% Bees Algorithm Main Loop
P=1;
for it=1:MaxIt
    if counter >= MaxEval
        break;
    end
    % Elite Sites
    for i=1:e
        BestForager.Cost=inf;
        for j=1:nep
            ForagerBees.Position= Foraging(Patch(i).Position);
            [ForagerBees.Cost,ForagerBees.Sol]=ObjFunction(ForagerBees.Position);
            ForagerBees.Stagnated = Patch(i).Stagnated;
            counter = counter + 1;
            ForagerBees.Counter = counter;
            if ForagerBees.Cost<BestForager.Cost
                BestForager=ForagerBees;
            end
        end
        if BestForager.Cost<Patch(i).Cost
            Patch(i)=BestForager;
```

```
            Patch(i).Stagnated=0;
        else
            Patch(i).Stagnated=Patch(i).Stagnated+1;
        end
        %site abandonment procedure
        if(Patch(i).Stagnated>stlim)
            Patch(i).Stagnated=0;
            Patch(i).Position= BRO_0_VRP(Patch(i),Instance,1,1);
            counter = counter + Dims;
            Patch(i).Counter = counter;
        end
    end
    % Selected Non-Elite Sites
    for i=e+1:m
        BestForager.Cost=inf;
        for j=1:nsp
            ForagerBees.Position= Foraging(Patch(i).Position);
            [ForagerBees.Cost,ForagerBees.Sol]=ObjFunction(ForagerBees.Position);
            ForagerBees.Stagnated = Patch(i).Stagnated;
            counter = counter + 1;
            ForagerBees.Counter = counter;
            if ForagerBees.Cost<BestForager.Cost
                BestForager=ForagerBees;
            end
        end
        if BestForager.Cost<Patch(i).Cost
            Patch(i)=BestForager;
            Patch(i).Stagnated=0;
        else
            Patch(i).Stagnated=Patch(i).Stagnated+1;
        end
        %site abandonment procedure
        if(Patch(i).Stagnated>stlim)
            Patch(i).Stagnated=0;
            Patch(i).Position= BRO_0_VRP(Patch(i),Instance,1,1);
            counter = counter + 1;
            Patch(i).Counter = counter;
        end
    end
    % Non-Selected Sites
    for i=m+1:n
        Patch(i).Position= BRO_0_VRP(Patch(i),Instance,1,1);
        [Patch(i).Cost,Patch(i).Sol] =ObjFunction(Patch(i).Position);
        Patch(i).Stagnated = 0;
        counter = counter + 1;
        Patch(i).Counter = counter;
        if(Patch(i).Stagnated>stlim)
            Patch(i).Stagnated=0;
            [Patch(i).Position,Patch(i).Cost, Patch(i).Sol]=BiBA_Clustering_VRP↙
(Instance);
            counter = counter + 1;
            Patch(i).Counter = counter;
        end
    end
    % Sort
```

```
    [~, SortOrder]=sort([Patch.Cost]);
    Patch=Patch(SortOrder);
    % Update Best Solution Ever Found
    BestSol=Patch(1);
    if BestSol.Cost < OptSol.Cost
        OptSol=BestSol;
    end
    % Store Best Cost Ever Found
    BestCost(it)=OptSol.Cost;
    Counter(it)=OptSol.Counter;
    OPTSol(it)=OptSol;
    %% Display Iteration Information
    if BestSol.Sol.IsFeasible
        FLAG=' *';
    else
        FLAG='';
    end
    disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(BestCost(it)) FLAG ';↙
Fittness Evaluations = ' num2str(Counter(it))]);
    figure(1);
    PlotSolution(BestSol.Sol,Instance);
    pause(0.01);
end
%% Results
figure;
semilogy(BestCost,'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
```

```
function [ z, L,sol] = BiBA_Clustering_VRP( model )
I=model.I;
J=model.J;
C=model.c;
X=[];
X(1,:)= model.x;
X(2,:)= model.y;
X=transpose(X);
[idx,~] = BiBA_Clust_VRP( X ,J ,model );
q=[];

for i=1:J
    q{i,1}=transpose(find(idx==i));
end
for j=1: J-1
    q{j}=[q{j} I+j];
end

z=q{1};
for j=2: J
    z =[z q{j}];
end

[L,sol] = MyCost_VRP(z,model);

end

function [ id,pos ] = BiBA_Clust_VRP( X,k,model )
    CostFunction=@(m) ClusteringCost_VRP(m, X, model);     % Cost Function
    VarSize=[k size(X,2)];  % Decision Variables Matrix Size
    nVar=prod(VarSize);     % Number of Decision Variables
    VarMin= repmat(min(X),k,1);      % Lower Bound of Variables
    VarMax= repmat(max(X),k,1);      % Upper Bound of Variables
    range=VarMax(1)-VarMin(1);
    %% SBA
    MaxEval = 5000;
    n=7;
    nep =7;
    Shrink = 1;
    accuracy=0.001;
    stlim=50;
    %recruitment = round(logspace(0,-1,n)*nep);
    recruitment = round(linspace(nep,1,n));
    ColonySize=sum(recruitment);  % total number of foragers
    MaxIt=round(MaxEval/ColonySize);
    %% Initialization
    Empty_Bees.Position=[];
    Empty_Bees.Cost=[];
    Empty_Bees.Out=[];
    Empty_Bees.Size=[];
    Empty_Bees.Stagnated =[];
    Empty_Bees.counter=[];
    Bees=repmat(Empty_Bees,n,1);
    counter=0;
     % Generate Initial Solutions
```

```
    for i=1:n
        Bees(i).Position=unifrnd(VarMin,VarMax,VarSize);
        [Bees(i).Cost,Bees(i).Out]=CostFunction(Bees(i).Position);
        Bees(i).Size = range;
        Bees(i).Stagnated = 0;
        counter=counter+1;
        Bees(i).counter= counter;
    end
    sz= linspace(0,1,n);
    %% Sites Selection
    [~, RankOrder]=sort([Bees.Cost]);
    Bees=Bees(RankOrder);
    P=1;
    BestCost=zeros(MaxIt,1);
    %% Bees Algorithm Local and Global Search
    for it=1:MaxIt
        if counter >= MaxEval
            break;
        end
        % All Sites (Exploitation and Exploration)
        for i=1:n
            bestnewbee.Cost=inf;
            assigntment=D_Tri_real_array(0,sz(i),1,1,recruitment(i));
            for j=1:recruitment(i)
                if P==1
                    newbee.Position= Integrated_Foraging_stlim_unif(Bees(i).Position,↙
assigntment(j),VarMax(1),VarMin(1),Bees(i).Size);
                else
                    newbee.Position= Integrated_Foraging_stlim(Bees(i).Position,↙
assigntment(j),VarMax(1),VarMin(1),Bees(i).Size);
                end
                [newbee.Cost,newbee.Out]=CostFunction(newbee.Position);
                newbee.Size= Bees(i).Size;
                newbee.Stagnated = Bees(i).Stagnated;
                counter=counter+1;
                newbee.counter= counter;
                if newbee.Cost<bestnewbee.Cost
                    bestnewbee=newbee;
                end
            end
            if bestnewbee.Cost<Bees(i).Cost
                Bees(i)=bestnewbee;
                Bees(i).Stagnated=0;
            else
                Bees(i).Stagnated=Bees(i).Stagnated+1;
                Bees(i).Size=Bees(i).Size*Shrink;
            end
            %site abandonment procedure
            if(Bees(i).Stagnated>stlim)
                Bees(i)=Bees(end);
                Bees(i).Size=range;
                Bees(i).Stagnated=0;
                Bees(i).Position=unifrnd(VarMin,VarMax,VarSize);
                P=P*-1;
            end
```

```
        end
        % SORTING
        [~, RankOrder]=sort([Bees.Cost]);
        Bees=Bees(RankOrder);
        % Update Best Solution Ever Found
        OptSol=Bees(1);
        BestCost(it)=OptSol.Cost;
    end
    figure(1);
    PlotSolution_C(X, OptSol);
    pause(0.01);
    id=OptSol.Out.ind;
    pos=OptSol.Position;
end

function [z, out] = ClusteringCost_VRP(m, X, model)
    J=model.J;
    C=model.r;
    maxC=model.c(1);

    d = pdist2(X, m);

    [dmin, ind] = min(d, [], 2);

    WCD = sum(dmin);

        [~, SortOrder]=sort([ind]);
    dmin=dmin(SortOrder);
    tes=zeros(1,J);
    VC=zeros(1,J);
    for i=1:J
        tes(i)=sum(C(find(ind==i)));
    end
    for j=1:J
        if tes(j) <= maxC
            VC(j)=0;
        else
            VC(j)=(tes(j) - maxC)*10;
        end
    end
    WVC=sum(VC);

    z=WCD+WVC*10^2;

    out.d=d;
    out.dmin=dmin;
    out.ind=ind;
    out.WCD=WCD;
end


function y=Integrated_Foraging_stlim_unif(x,ass,Vmx,Vmn,size)
    r=ass*size;

    y=x;
```

```
    y = y + (random('unif',-r,r)); %.*pert);
    y(y>Vmx)=Vmx;
    y(y<Vmn)=Vmn;
end


function y=Integrated_Foraging_stlim(x,ass,Vmx,Vmn,size)
    r=ass*size;

    nVar=numel(x);

    k=randi([1 nVar]);

    y=x;
    y(k)=y(k)+ r*((-1)^randi(2));
    y(y>Vmx)=Vmx;
    y(y<Vmn)=Vmn;
end

function PlotSolution_C(X, sol)
    m = sol.Position;
    k = size(m,1);
    ind = sol.Out.ind;
    Colors = hsv(k);
    for j=1:k
        Xj = X(ind==j,:);
        plot(Xj(:,1),Xj(:,2),'x','LineWidth',1,'Color',Colors(j,:));
        hold on;
    end
    plot(m(:,1),m(:,2),'ok','LineWidth',2,'MarkerSize',12);
    hold off;
    grid on;
end
```

```
function NewPatch = Foraging(sol)

    m=randi([1 3]);

    switch m
        case 1
            NewPatch = Swap2(sol);
        case 2
            NewPatch = Reverse2(sol);
        case 3
            NewPatch = Insert2(sol);
    end
end

function NewPatch = Swap2(sol)
    n=numel(sol);
    i=randsample(n,2);
    i1=i(1);
    i2=i(2);
    NewPatch=sol;
    NewPatch([i1 i2])=sol([i2 i1]);
end

function NewPatch = Reverse2(sol)
    n=numel(sol);
    i=randsample(n,2);
    i1=min(i(1),i(2));
    i2=max(i(1),i(2));
    NewPatch = sol;
    NewPatch(i1:i2)=sol(i2:-1:i1);
end

function NewPatch = Insert2(sol)
    n=numel(sol);
    i=randsample(n,2);
    i1=i(1);
    i2=i(2);
    if i1<i2
        NewPatch = [sol(1:i1-1) sol(i1+1:i2) sol(i1) sol(i2+1:end)];
    else
        NewPatch = [sol(1:i2) sol(i1) sol(i2+1:i1-1) sol(i1+1:end)];
    end
end
```

```
function q = BRO_0_VRP( p, model, BatasB, BatasA)
I=model.I;
J=model.J;
l=randi([1 J]);
x0=model.x0;
y0=model.y0;

PTemp=p.Sol.L{l};
x=[model.x(PTemp) x0];
y=[model.y(PTemp) y0];
PTemp=[PTemp I+l];
t(:,1)=x;t(:,2)=y;
D=round(pdist2(t,t));
n=numel(PTemp);

if n<=1
    q=p.Position;

else
    u=1:n;

    idrem = randperm(n,[randi([BatasB BatasA])]);

    rem = u(idrem);
    u(idrem)=[];

    u = TwoOpt_0(u,D);

    [u,~] = Insert_Forgotten( rem,u,D,x,y);

    u(u==max(u))=[];

    sequence=PTemp(u);
    p.Sol.L{l}=sequence;
    for i=1:J-1
        p.Sol.L{i}=[p.Sol.L{i} I+i];
    end
    Temp=transpose(p.Sol.L);
    q=cell2mat(Temp);

end
end

function p = TwoOpt_0(p,D)
n = numel(p);
if n==0
    return
end
zmin = -1;
% Iterate until the tour is 2-optimal
while zmin < 0
    zmin = 0;
    i = 0;
    b = p(n);
    % Loop over all edge pairs (ab,cd)
```

```
    while i < n-2
        a = b;
        i = i+1;
        b = p(i);
        Dab = D(a,b);
        j = i+1;
        d = p(j);
        while j < n
            c = d;
            j = j+1;
            d = p(j);
            z = (D(a,c) - D(c,d)) + D(b,d) - Dab;
            % Keep best exchange
            if z < zmin
                zmin = z;
                imin = i;
                jmin = j;
            end
        end
    end
    % Apply exchange
    if zmin < 0
        p(imin:jmin-1) = p(jmin-1:-1:imin);
    end
end
end

function [p,L] = Insert_Forgotten( rem,init,D,x,y )
p=init;
nrem = numel (rem);
for i = 1 : nrem
    np = numel (p);

    Dis = zeros(np,1);
    center.x(1) = (x(p(1))+x(p(end)))*0.5;
    center.y(1) = (y(p(1))+y(p(end)))*0.5;
    Dis(1) = (sqrt((x(rem(i))-center.x(1))^2+(y(rem(i))-center.y(1))^2));
    for j=2:np
        center.x(j) = (x(p(j))+x(p(j-1)))*0.5;
        center.y(j) = (y(p(j))+y(p(j-1)))*0.5;
        Dis(j) = (sqrt((x(rem(i))-center.x(j))^2+(y(rem(i))-center.y(j))^2));
    end
   [a,b] = min (Dis);
    s = p(b);
    idx=find(p==s);
    if idx==1
        p = [rem(i) p];
    else
        p = [p(1:idx-1) rem(i) p(idx:end)];
    end
end
L=TourLength_0(p,D);
end

function L=TourLength_0(tour,D)
```

```
    n=numel(tour);

    tour=[tour tour(1)];

    L=0;
    for i=1:n
        L=L+D(tour(i),tour(i+1));
    end

end
```

```
function [z, sol]=MyCost_VRP(q,model)

    sol=ParseSolution_VRP(q,model);

    beta=10^9;

    z=sol.TotalD + (beta*sol.MeanCV);

end

function sol=ParseSolution_VRP(q,model)
    I=model.I;
    J=model.J;
    d=model.d;
    d0=model.d0;
    r=model.r;
    c=model.c;

    DelPos=find(q>I);

    From=[0 DelPos]+1;
    To=[DelPos I+J]-1;

    L=cell(J,1);
    D=zeros(1,J);
    UC=zeros(1,J);
    for j=1:J
        L{j}=q(From(j):To(j));

        if ~isempty(L{j})

            D(j)=d0(L{j}(1));

            for k=1:numel(L{j})-1
                D(j)=D(j)+d(L{j}(k),L{j}(k+1));
            end

            D(j)=D(j)+d0(L{j}(end));


            UC(j)=sum(r(L{j}));
        end
    end

    CV=max(UC./c-1,0);

    MeanCV=mean(CV);

    sol.L=L;
    sol.D=D;
    sol.MaxD=max(D);
    sol.TotalD=sum(D);
    sol.UC=UC;
    sol.CV=CV;
    sol.MeanCV=MeanCV;
```

```
    sol.IsFeasible=(MeanCV==0.00);

end
```

```
function [ M ] = D_Tri_real_array(k,t,b,baris,kolom)
    M=zeros(baris,kolom);
    for i=1:baris
        for j=1:kolom
            M(i,j)=D_Tri_real(k,t,b);
        end
    end

end

function [ angka ] = D_Tri_real(k,t,b)
m=randi([1 10]);
    a=(t-k)/10;
    b=(b-t)/10;
    switch m
        case 1
            angka=lapis1(t,a,b);
        case 2
            angka=lapis2(t,a,b);
        case 3
            angka=lapis3(t,a,b);
        case 4
            angka=lapis4(t,a,b);
        case 5
            angka=lapis5(t,a,b);
        case 6
            angka=lapis6(t,a,b);
        case 7
            angka=lapis7(t,a,b);
        case 8
            angka=lapis8(t,a,b);
        case 9
            angka=lapis9(t,a,b);
        case 10
            angka=lapis10(t,a,b);
    end
end

function angka=lapis1(t,a,b)
    angka=unifrnd((t-a),(t+b),1);
end

function angka=lapis2(t,a,b)
    angka=unifrnd((t-2*a),(t+2*b),1);
end

function angka=lapis3(t,a,b)
    angka=unifrnd((t-3*a),(t+3*b),1);
end

function angka=lapis4(t,a,b)
    angka=unifrnd((t-4*a),(t+4*b),1);
end

function angka=lapis5(t,a,b)
```

```
    angka=unifrnd((t-5*a),(t+5*b),1);
end

function angka=lapis6(t,a,b)
    angka=unifrnd((t-6*a),(t+6*b),1);
end

function angka=lapis7(t,a,b)
    angka=unifrnd((t-7*a),(t+7*b),1);
end

function angka=lapis8(t,a,b)
    angka=unifrnd((t-8*a),(t+8*b),1);
end

function angka=lapis9(t,a,b)
    angka=unifrnd((t-9*a),(t+9*b),1);
end

function angka=lapis10(t,a,b)
    angka=unifrnd((t-10*a),(t+10*b),1);
end
```

```
function PlotSolution(sol,model)
    J=model.J;
    xmin=model.xmin;
    xmax=model.xmax;
    ymin=model.ymin;
    ymax=model.ymax;
    x=model.x;
    y=model.y;
    x0=model.x0;
    y0=model.y0;
    L=sol.L;
    Colors=hsv(J);
    for j=1:J
        if isempty(L{j})
            continue;
        end

        X=[x0 x(L{j}) x0];
        Y=[y0 y(L{j}) y0];

        Color=0.8*Colors(j,:);

        plot(X,Y,'-o',...
            'Color',Color,...
            'LineWidth',1,...
            'MarkerSize',5,...
            'MarkerFaceColor','white');
        hold on;

    end

    plot(x0,y0,'ks',...
        'LineWidth',1,...
        'MarkerSize',15,...
        'MarkerFaceColor','yellow');

    hold off;
    grid on;
    axis equal;

    xlim([xmin xmax]);
    ylim([ymin ymax]);

end
```

```
classdef TsplibVRP
    properties
        type;
        I;
        J;
        dim;
        optima;
        r;
        c;
        xmin;
        xmax;
        ymin;
        ymax;
        x;
        y;
        x0
        y0;
        d;
        D;
        d0;
        eta;
        definedFunctions
    end
    methods
        function obj = TsplibVRP(typeOfFunction)
            obj.definedFunctions=↙
{'Eil33','Eil51','EilA76','EilB76','EilC76','EilD76','EilA101','EilB101'};
            obj.type=typeOfFunction;
            switch(obj.type)
                case 'Eil33',
                    obj.I=32;
                    obj.J=4;
                    obj.dim=obj.I+obj.J-1;
                    obj.optima=835;
                    obj.r=↙
[700,400,400,1200,40,80,2000,900,600,750,1500,150,250,1600,450,700,550,650,200,400,300↙
,1300,700,750,1400,4000,600,1000,500,2500,1700,1100];
                    obj.c=[8000,8000,8000,8000];
                    obj.x=↙
[298,309,307,336,320,321,322,323,324,323,314,311,304,293,296,261,297,315,314,321,321,3↙
14,313,304,295,283,279,271,264,277,290,319];
                    obj.y=↙
[427,445,464,475,439,437,437,433,433,429,435,442,427,421,418,384,410,407,406,391,398,3↙
94,378,382,402,406,399,401,414,439,434,433];
                    A= obj.x;
                    B= obj.y;
                    obj.x0=292;
                    obj.y0=495;
                    for i=1:obj.J
                        A=[A obj.x0];
                        B=[B obj.y0];
                    end
                    obj.xmin=min(min(obj.x), obj.x0);
                    obj.xmax=max(max(obj.x), obj.x0);
                    obj.ymin=min(min(obj.y), obj.y0);
```

```
                    obj.ymax=max(max(obj.y), obj.y0);;
                    t(:,1)=obj.x;
                    t(:,2)=obj.y;
                    T(:,1)=A;
                    T(:,2)=B;
                    obj.d=round(pdist2(t,t));
                    obj.D=round(pdist2(T,T));
                    O(:,1)=obj.x0;
                    O(:,2)=obj.y0;
                    obj.d0=transpose(round(pdist2(T,O)));
                    obj.eta = 0.5;
                case 'Eil51',
                    obj.I=50;
                    obj.J=5;
                    obj.dim=obj.I+obj.J-1;
                    obj.optima=521;
                    obj.r=↙
[7,30,16,9,21,15,19,23,11,5,19,29,23,21,10,15,3,41,9,28,8,8,16,10,28,7,15,14,6,19,11,1↙
2,23,26,17,6,9,15,14,7,27,13,11,16,10,5,25,17,18,10];
                    obj.c=[160,160,160,160,160];
                    obj.x=↙
[37,49,52,20,40,21,17,31,52,51,42,31,5,12,36,52,27,17,13,57,62,42,16,8,7,27,30,43,58,5↙
8,37,38,46,61,62,63,32,45,59,5,10,21,5,30,39,32,25,25,48,56];
                    obj.y=↙
[52,49,64,26,30,47,63,62,33,21,41,32,25,42,16,41,23,33,13,58,42,57,57,52,38,68,48,67,4↙
8,27,69,46,10,33,63,69,22,35,15,6,17,10,64,15,10,39,32,55,28,37];
                    A= obj.x;
                    B= obj.y;
                    obj.x0=30;
                    obj.y0=40;
                    for i=1:obj.J
                        A=[A obj.x0];
                        B=[B obj.y0];
                    end
                    obj.xmin=min(min(obj.x), obj.x0);
                    obj.xmax=max(max(obj.x), obj.x0);
                    obj.ymin=min(min(obj.y), obj.y0);
                    obj.ymax=max(max(obj.y), obj.y0);;
                    t(:,1)=obj.x;
                    t(:,2)=obj.y;
                    T(:,1)=A;
                    T(:,2)=B;
                    obj.d=round(pdist2(t,t));
                    obj.D=round(pdist2(T,T));
                    O(:,1)=obj.x0;
                    O(:,2)=obj.y0;
                    obj.d0=transpose(round(pdist2(T,O)));
                    obj.eta = 0.5;
                case 'EilA76',
                    obj.I=75;
                    obj.J=10;
                    obj.dim=obj.I+obj.J-1;
                    obj.optima=826;
                    obj.r=↙
[18,26,11,30,21,19,15,16,29,26,37,16,12,31,8,19,20,13,15,22,28,12,6,27,14,18,17,29,13,↙
```

```
22,25,28,27,19,10,12,14,24,16,33,15,11,18,17,21,27,19,20,5,22,12,19,22,16,7,26,14,21,2↙
4,13,15,18,11,28,9,37,30,10,8,11,3,1,6,10,20];
                    obj.c=[140,140,140,140,140,140,140,140,140,140];
                    obj.x=↙
[22,36,21,45,55,33,50,55,26,40,55,35,62,62,62,21,33,9,62,66,44,26,11,7,17,41,55,35,52,↙
43,31,22,26,50,55,54,60,47,30,30,12,15,16,21,50,51,50,48,12,15,29,54,55,67,10,6,65,40,↙
70,64,36,30,20,15,50,57,45,38,50,66,59,35,27,40,40];
                    obj.y=↙
[22,26,45,35,20,34,50,45,59,66,65,51,35,57,24,36,44,56,48,14,13,13,28,43,64,46,34,16,2↙
6,26,76,53,29,40,50,10,15,66,60,50,17,14,19,48,30,42,15,21,38,56,39,38,57,41,70,25,27,↙
60,64,4,6,20,30,5,70,72,42,33,4,8,5,60,24,20,37];
                    A= obj.x;
                    B= obj.y;
                    obj.x0=40;
                    obj.y0=40;
                    for i=1:obj.J
                        A=[A obj.x0];
                        B=[B obj.y0];
                    end
                    obj.xmin=min(min(obj.x), obj.x0);
                    obj.xmax=max(max(obj.x), obj.x0);
                    obj.ymin=min(min(obj.y), obj.y0);
                    obj.ymax=max(max(obj.y), obj.y0);;
                    t(:,1)=obj.x;
                    t(:,2)=obj.y;
                    T(:,1)=A;
                    T(:,2)=B;
                    obj.d=round(pdist2(t,t));
                    obj.D=round(pdist2(T,T));
                    O(:,1)=obj.x0;
                    O(:,2)=obj.y0;
                    obj.d0=transpose(round(pdist2(T,O)));
                    obj.eta = 0.5;
                case 'EilB76',
                    obj.I=75;
                    obj.J=14;
                    obj.dim=obj.I+obj.J-1;
                    obj.optima=1019;
                    obj.r=↙
[18,26,11,30,21,19,15,16,29,26,37,16,12,31,8,19,20,13,15,22,28,12,6,27,14,18,17,29,13,↙
22,25,28,27,19,10,12,14,24,16,33,15,11,18,17,21,27,19,20,5,22,12,19,22,16,7,26,14,21,2↙
4,13,15,18,11,28,9,37,30,10,8,11,3,1,6,10,20];
                    obj.c=[100,100,100,100,100,100,100,100,100,100,100,100,100,100];
                    obj.x=↙
[22,36,21,45,55,33,50,55,26,40,55,35,62,62,62,21,33,9,62,66,44,26,11,7,17,41,55,35,52,↙
43,31,22,26,50,55,54,60,47,30,30,12,15,16,21,50,51,50,48,12,15,29,54,55,67,10,6,65,40,↙
70,64,36,30,20,15,50,57,45,38,50,66,59,35,27,40,40];
                    obj.y=↙
[22,26,45,35,20,34,50,45,59,66,65,51,35,57,24,36,44,56,48,14,13,13,28,43,64,46,34,16,2↙
6,26,76,53,29,40,50,10,15,66,60,50,17,14,19,48,30,42,15,21,38,56,39,38,57,41,70,25,27,↙
60,64,4,6,20,30,5,70,72,42,33,4,8,5,60,24,20,37];
                    A= obj.x;
                    B= obj.y;
                    obj.x0=40;
                    obj.y0=40;
```

```
                    for i=1:obj.J
                        A=[A obj.x0];
                        B=[B obj.y0];
                    end
                    obj.xmin=min(min(obj.x), obj.x0);
                    obj.xmax=max(max(obj.x), obj.x0);
                    obj.ymin=min(min(obj.y), obj.y0);
                    obj.ymax=max(max(obj.y), obj.y0);;
                    t(:,1)=obj.x;
                    t(:,2)=obj.y;
                    T(:,1)=A;
                    T(:,2)=B;
                    obj.d=round(pdist2(t,t));
                    obj.D=round(pdist2(T,T));
                    O(:,1)=obj.x0;
                    O(:,2)=obj.y0;
                    obj.d0=transpose(round(pdist2(T,O)));
                    obj.eta = 0.5;
                case 'EilC76',
                    obj.I=75;
                    obj.J=8;
                    obj.dim=obj.I+obj.J-1;
                    obj.optima=735;
                    obj.r=↙
[18,26,11,30,21,19,15,16,29,26,37,16,12,31,8,19,20,13,15,22,28,12,6,27,14,18,17,29,13,↙
22,25,28,27,19,10,12,14,24,16,33,15,11,18,17,21,27,19,20,5,22,12,19,22,16,7,26,14,21,2↙
4,13,15,18,11,28,9,37,30,10,8,11,3,1,6,10,20];
                    obj.c=[180,180,180,180,180,180,180,180];
                    obj.x=↙
[22,36,21,45,55,33,50,55,26,40,55,35,62,62,62,21,33,9,62,66,44,26,11,7,17,41,55,35,52,↙
43,31,22,26,50,55,54,60,47,30,30,12,15,16,21,50,51,50,48,12,15,29,54,55,67,10,6,65,40,↙
70,64,36,30,20,15,50,57,45,38,50,66,59,35,27,40,40];
                    obj.y=↙
[22,26,45,35,20,34,50,45,59,66,65,51,35,57,24,36,44,56,48,14,13,13,28,43,64,46,34,16,2↙
6,26,76,53,29,40,50,10,15,66,60,50,17,14,19,48,30,42,15,21,38,56,39,38,57,41,70,25,27,↙
60,64,4,6,20,30,5,70,72,42,33,4,8,5,60,24,20,37];
                    A= obj.x;
                    B= obj.y;
                    obj.x0=40;
                    obj.y0=40;
                    for i=1:obj.J
                        A=[A obj.x0];
                        B=[B obj.y0];
                    end
                    obj.xmin=min(min(obj.x), obj.x0);
                    obj.xmax=max(max(obj.x), obj.x0);
                    obj.ymin=min(min(obj.y), obj.y0);
                    obj.ymax=max(max(obj.y), obj.y0);;
                    t(:,1)=obj.x;
                    t(:,2)=obj.y;
                    T(:,1)=A;
                    T(:,2)=B;
                    obj.d=round(pdist2(t,t));
                    obj.D=round(pdist2(T,T));
                    O(:,1)=obj.x0;
```

```
                    O(:,2)=obj.y0;
                    obj.d0=transpose(round(pdist2(T,O)));
                    obj.eta = 0.5;

                case 'EilD76',
                    obj.I=75;
                    obj.J=7;
                    obj.dim=obj.I+obj.J-1;
                    obj.optima=682;
                    obj.r=↙
[18,26,11,30,21,19,15,16,29,26,37,16,12,31,8,19,20,13,15,22,28,12,6,27,14,18,17,29,13,↙
22,25,28,27,19,10,12,14,24,16,33,15,11,18,17,21,27,19,20,5,22,12,19,22,16,7,26,14,21,2↙
4,13,15,18,11,28,9,37,30,10,8,11,3,1,6,10,20];
                    obj.c=[220,220,220,220,220,220,220];
                    obj.x=↙
[22,36,21,45,55,33,50,55,26,40,55,35,62,62,62,21,33,9,62,66,44,26,11,7,17,41,55,35,52,↙
43,31,22,26,50,55,54,60,47,30,30,12,15,16,21,50,51,50,48,12,15,29,54,55,67,10,6,65,40,↙
70,64,36,30,20,15,50,57,45,38,50,66,59,35,27,40,40];
                    obj.y=↙
[22,26,45,35,20,34,50,45,59,66,65,51,35,57,24,36,44,56,48,14,13,13,28,43,64,46,34,16,2↙
6,26,76,53,29,40,50,10,15,66,60,50,17,14,19,48,30,42,15,21,38,56,39,38,57,41,70,25,27,↙
60,64,4,6,20,30,5,70,72,42,33,4,8,5,60,24,20,37];
                    A= obj.x;
                    B= obj.y;
                    obj.x0=40;
                    obj.y0=40;
                    for i=1:obj.J
                        A=[A obj.x0];
                        B=[B obj.y0];
                    end
                    obj.xmin=min(min(obj.x), obj.x0);
                    obj.xmax=max(max(obj.x), obj.x0);
                    obj.ymin=min(min(obj.y), obj.y0);
                    obj.ymax=max(max(obj.y), obj.y0);;
                    t(:,1)=obj.x;
                    t(:,2)=obj.y;
                    T(:,1)=A;
                    T(:,2)=B;
                    obj.d=round(pdist2(t,t));
                    obj.D=round(pdist2(T,T));
                    O(:,1)=obj.x0;
                    O(:,2)=obj.y0;
                    obj.d0=transpose(round(pdist2(T,O)));
                    obj.eta = 0.5;

                case 'EilA101',
                    obj.I=100;
                    obj.J=8;
                    obj.dim=obj.I+obj.J-1;
                    obj.optima=817;
                    obj.r=↙
[10,7,13,19,26,3,5,9,16,16,12,19,23,20,8,19,2,12,17,9,11,18,29,3,6,17,16,16,9,21,27,23↙
,11,14,8,5,8,16,31,9,5,5,7,18,16,1,27,36,30,13,10,9,14,18,2,6,7,18,28,3,13,19,10,9,20,↙
25,25,36,6,5,15,25,9,8,18,13,14,3,23,6,26,16,11,7,41,35,26,9,15,3,1,2,22,27,20,11,12,1↙
0,9,17];
```

```
                    obj.c=[200,200,200,200,200,200,200,200];
                    obj.x=↙
[41,35,55,55,15,25,20,10,55,30,20,50,30,15,30,10,5,20,15,45,45,45,55,65,65,45,35,41,64↙
,40,31,35,53,65,63,2,20,5,60,40,42,24,23,11,6,2,8,13,6,47,49,27,37,57,63,53,32,36,21,1↙
7,12,24,27,15,62,49,67,56,37,37,57,47,44,46,49,49,53,61,57,56,55,15,14,11,16,4,28,26,2↙
6,31,15,22,18,26,25,22,25,19,20,18];
                    obj.y=↙
[49,17,45,20,30,30,50,43,60,60,65,35,25,10,5,20,30,40,60,65,20,10,5,35,20,30,40,37,42,↙
60,52,69,52,55,65,60,20,5,12,25,7,12,3,14,38,48,56,52,68,47,58,43,31,29,23,12,12,26,24↙
,34,24,58,69,77,77,73,5,39,47,56,68,16,17,13,11,42,43,52,48,37,54,47,37,31,22,18,18,52↙
,35,67,19,22,24,27,24,27,21,21,26,18];
                    A= obj.x;
                    B= obj.y;
                    obj.x0=35;
                    obj.y0=35;
                    for i=1:obj.J
                        A=[A obj.x0];
                        B=[B obj.y0];
                    end
                    obj.xmin=min(min(obj.x), obj.x0);
                    obj.xmax=max(max(obj.x), obj.x0);
                    obj.ymin=min(min(obj.y), obj.y0);
                    obj.ymax=max(max(obj.y), obj.y0);;
                    t(:,1)=obj.x;
                    t(:,2)=obj.y;
                    T(:,1)=A;
                    T(:,2)=B;
                    obj.d=round(pdist2(t,t));
                    obj.D=round(pdist2(T,T));
                    O(:,1)=obj.x0;
                    O(:,2)=obj.y0;
                    obj.d0=transpose(round(pdist2(T,O)));
                    obj.eta = 0.5;
                case 'EilB101',
                    obj.I=100;
                    obj.J=14;
                    obj.dim=obj.I+obj.J-1;
                    obj.optima=1077;
                    obj.r=↙
[10,7,13,19,26,3,5,9,16,16,12,19,23,20,8,19,2,12,17,9,11,18,29,3,6,17,16,16,9,21,27,23↙
,11,14,8,5,8,16,31,9,5,5,7,18,16,1,27,36,30,13,10,9,14,18,2,6,7,18,28,3,13,19,10,9,20,↙
25,25,36,6,5,15,25,9,8,18,13,14,3,23,6,26,16,11,7,41,35,26,9,15,3,1,2,22,27,20,11,12,1↙
0,9,17];
                    obj.c=[112,112,112,112,112,112,112,112,112,112,112,112,112,112];
                    obj.x=↙
[41,35,55,55,15,25,20,10,55,30,20,50,30,15,30,10,5,20,15,45,45,45,55,65,65,45,35,41,64↙
,40,31,35,53,65,63,2,20,5,60,40,42,24,23,11,6,2,8,13,6,47,49,27,37,57,63,53,32,36,21,1↙
7,12,24,27,15,62,49,67,56,37,37,57,47,44,46,49,49,53,61,57,56,55,15,14,11,16,4,28,26,2↙
6,31,15,22,18,26,25,22,25,19,20,18];
                    obj.y=↙
[49,17,45,20,30,30,50,43,60,60,65,35,25,10,5,20,30,40,60,65,20,10,5,35,20,30,40,37,42,↙
60,52,69,52,55,65,60,20,5,12,25,7,12,3,14,38,48,56,52,68,47,58,43,31,29,23,12,12,26,24↙
,34,24,58,69,77,77,73,5,39,47,56,68,16,17,13,11,42,43,52,48,37,54,47,37,31,22,18,18,52↙
,35,67,19,22,24,27,24,27,21,21,26,18];
                    A= obj.x;
```

```
                    B= obj.y;
                    obj.x0=35;
                    obj.y0=35;
                    for i=1:obj.J
                        A=[A obj.x0];
                        B=[B obj.y0];
                    end
                    obj.xmin=min(min(obj.x), obj.x0);
                    obj.xmax=max(max(obj.x), obj.x0);
                    obj.ymin=min(min(obj.y), obj.y0);
                    obj.ymax=max(max(obj.y), obj.y0);;
                    t(:,1)=obj.x;
                    t(:,2)=obj.y;
                    T(:,1)=A;
                    T(:,2)=B;
                    obj.d=round(pdist2(t,t));
                    obj.D=round(pdist2(T,T));
                    O(:,1)=obj.x0;
                    O(:,2)=obj.y0;
                    obj.d0=transpose(round(pdist2(T,O)));
                    obj.eta = 0.5;
                otherwise,
                  disp('fitness function not defined');
                end
        end

        function [fitness, sol] = evaluation(obj, tour)
            global eval;
            eval = eval+1;
            [fitness, sol] = MyCost_VRP (tour,obj);
        end
    end
end
```

References

Aalizadeh, B. and Asnafi, A. (2016). Integrated bees algorithm and artificial neural network to propose an efficient controller for active front steering control of vehicles. *International Journal of Automotive and Mechanical Engineering*, 13:3476.

Abbass, H. A. (2001). Mbo: Marriage in honey bees optimization-a haplometrosis polygynous swarming approach. In *Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546)*, volume 1, pages 207–214. IEEE.

Abdelhakim, A. M., Saleh, H. I., and Nassar, A. M. (2016). Quality metric-based fitness function for robust watermarking optimisation with Bees algorithm. *IET Image Processing*, 10(3):247–252.

AbdelHamid, N. M., Halim, M. A., and Fakhr, M. W. (2013). BEES ALGORITHM-BASED DOCUMENT CLUSTERING. In *ICIT The 6th International Conference on Information Technology*.

Abdul-Razaq, T. S. and Ali, F. H. (2015). Hybrid Bees Algorithm to Solve Aircraft Landing Problem. *Journal of Zankoy Sulaimani - Part A*, 17(1):71–90.

Abdullah, S. and Alzaqebah, M. (2013). A hybrid self-adaptive bees algorithm for examination timetabling problems. *Applied Soft Computing*, 13-8(8):3608–3620.

Abirami, R., Andrea, L., and Diviya, M. (2018). Optimal detection of lung cancer using bees algorithm. *International Journal of Pure and Applied Mathematics*, 118(18):1775–1781.

Aboutorabi, S. J. S. and Rezvani, M. H. (2020). An optimized meta-heuristic bees algorithm for players' frame rate allocation problem in cloud gaming environments. *The Computer Games Journal*, 9(3):281–304.

Acar, O., Kalyoncu, M., and Hassan, A. (2019). Proposal of a harmonic bees algorithm for design optimization of a gripper mechanism. In *IFToMM World Congress on Mechanism and Machine Science*, pages 2829–2839. Springer.

References

Addeh, A., Khormali, A., and Golilarz, N. A. (2018). Control chart pattern recognition using rbf neural network with new training algorithm and practical features. *ISA transactions*, 79:202–216.

Addeh, J. and Ebrahimzadeh, A. (2013). Control Chart Patterns Recognition Using Fuzzy Rules and Improved Bees Algorithm. In *CONGRESS OF ELECTRIC INDUSTRY AUTOMATION*.

Ahangarpour, A., Farbod, M., Ghanbarzadeh, A., Moradi, A., and MirzakhaniNafchi, A. (2018). Optimization of continual production of CNTs by CVD method using radial basic function (RBF) neural network and the bees algorithm. *Journal of Nanostructures*, 8(3):225–231.

Ahmad, S. A., Pham, D. T., Ng, K. W., and Ang, M. C. (2012). TRIZ-inspired asymmetrical search neighborhood in the bees algorithm. In *Proceedings - 6th Asia International Conference on Mathematical Modelling and Computer Simulation, AMS 2012*, pages 29–33.

Ahmad, S. A. and Sunthiram, D. (2018). Optimization of diesel engine performance by the bees algorithm. In *IOP Conference Series: Materials Science and Engineering*, volume 319, page 012064. IOP Publishing.

Ahmed, J. A. and Brifcani, A. M. A. (2015). A new internal architecture based on feature selection for holonic manufacturing system. *International Journal of Industrial and Manufacturing Engineering*, 2(8):1431.

Akamine, Y. (2019). Optimal design of power system stabilizer based on multilayer perceptron neural networks using bee's algorithm. *Annals of Electrical and Electronic Engineering*, 2(9):6–11.

Akkar, H. A. (2010). Optimizing opto-electronic cellular neural networks using bees swarm intelligent. *Engineering and Technology Journal*, 28(21):6237–6252.

Akpinar, Ş. and Baykasoğlu, A. (2014a). Modeling and solving mixed-model assembly line balancing problem with setups. part ii: A multiple colony hybrid bees algorithm. *Journal of Manufacturing Systems*, 33(4):445–461.

Akpinar, Ş. and Baykasoğlu, A. (2014b). Multiple colony bees algorithm for continuous spaces. *Applied Soft Computing*, 24:829–841.

Akram Chaweshly, S. (2010). Proposal of Mutation-Based Bees Algorithm (MBA) to Solve Traveling Salesman & Jobs Scheduling Problems. *Engineering and Technology Journal*, 28(19):5833–5843.

Al-Araji, A. and Yousif, N. (2017). A cognitive nonlinear trajectory tracking controller design for wheeled mobile robot based on hybrid bees-pso algorithm. *Engineering and Technology Journal*, 35(6 Part A):609–616.

Al-Araji, A. S. (2017). Development of an on-line self-tuning fpga-pid-pwm control algorithm design for dc-dc buck converter in mobile applications. *Journal of Engineering*, 23(8):84–106.

Al-Araji, A. S. (2019). An adaptive swing-up sliding mode controller design for a real inverted pendulum system based on culture-bees algorithm. *European Journal of Control*, 45:45–56.

Al-dawoodi, A. G. M. and Mahmuddin, M. (2017). An empirical study of double-bridge search move on subset feature selection search of bees algorithm. *Journal of Telecommunication, Electronic and Computer Engineering*, 9(2-2):11–15.

Al-Negheimish, S., Alnuhait, F., Albrahim, H., Al-Mogherah, S., Alrajhi, M., and Hosny, M. (2018). An intelligent bio-inspired algorithm for the faculty scheduling problem. *Int. J. Adv. Comput. Sci. Appl*, 9:151–159.

Albayrak, M. and Allahverdi, N. (2011). Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms. *Expert Systems with Applications*, 38(3):1313–1320.

Alfi, A. and Khosravi, A. (2012). Constrained Nonlinear Optimal Control via a Hybrid BA-SD. *International Journal of Engineering*, 25(3 (C)):197–204.

Alhuwaishel, N. and Manar, H. (2015). A hybrid bees/demon optimization algorithm for solving the university course timetabling problem. In *Proceedings of the 3rd NAUN International Conference on Mathematical, Computational and Statistical Sciences. Dubai, United Arab Emirates, February*, pages 22–24.

Ali, F. H., Al-Safi, M. G., and Yousif, A. A. (2018). Analyzing cryptosystems by using artificial intelligence. *Al-Nahrain Journal of Science*, 1:100–108.

Ali, G. A. and Al Masud, S. M. R. (2018). Routing optimization in wban using bees algorithm for overcrowded hajj environment. *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, 9(5):75–79.

Ali, G. A. and Jantan, A. (2011). A new approach based on honeybee to improve intrusion detection system using neural network and bees algorithm. *Communications in Computer and Information Science*, 181 CCIS(PART 3):777–792.

Ali, I. K. (2013a). Automated Cryptanalysis of Transposition Cipher Using Bees Algorithm. *Journal of Baghdad College of Economic sciences University*, 2013(4):521–530.

Ali, I. K. (2013b). Cryptanalysis of Simple Substitution Ciphers Using Bees Algorithm. *Journal of Baghdad College of Economic sciences University*, 2013(36):373–382.

Ali, I. K. and Mahmod, A. G. (2015). Hybrid Bees Algorithm With Simulated Annealing for Cryptanalysis of Simple Substitution Cipher. *Journal of Babylon University / Pure and Applied Sciences*, 23(2):565–574.

References

Alimouri, P., Moradi, S., and Chinipardaz, R. (2017). Updating finite element model using frequency domain decomposition method and bees algorithm. *Journal of Computational Applied Mechanics*, 48(1):75–88.

Alimouri, P., Moradi, S., and Chinipardaz, R. (2018). Updating finite element model using stochastic subspace identification method and bees optimization algorithm. *Latin American Journal of Solids and Structures*, 15(2).

Almaneea, L. I. and Hosny, M. I. (2018). A two level hybrid bees algorithm for operating room scheduling problem. In *Science and Information Conference*, pages 272–290. Springer.

Almansob, S. M., Jalil, A. A., and Lomte, S. S. (2017). The use of k-nn and bees algorithm for big data intrusion detection system. *IOSR Journal of Computer Engineering*, 19(01):08–12.

Alomari, O. and Othman, Z. A. (2012). Bees algorithm for feature selection in network anomaly detection. *Journal of Applied Sciences Research*, 8(3):1748–1756.

Alzaqebah, M., Jawarneh, S., Sarim, H. M., and Abdullah, S. (2018). Bees algorithm for vehicle routing problems with time windows. *International Journal of Machine Learning and Computing*, 8(3):234–240.

Amirinejad, M., Eslami, M., and Noori, A. (2014). Automatic PID Controller Parameter Tuning Using Bees Algorithm. *International Journal of Scientific & Engineering Research*, 5(8):24–28.

Anantasate, S. and Bhasaputra, P. (2011). A multi-objective bees algorithm for multi-objective optimal power flow problem. In *The 8th Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI) Association of Thailand-Conference 2011*, pages 852–856.

Anantasate, S., Chokpanyasuwan, C., and Bhasaputra, P. (2010). Optimal power flow by using Bees algorithm. In *ECTI-CON 2010 - The 2010 ECTI International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, volume 8, pages 430–434.

Ananthi, J. and Ranganathan, V. (2016). Multilayer perceptron weight optimization using bee swarm algorithm for mobility prediction. *IIOAB Journal*, 7(9):47–63.

Ang, M. C., Ng, K. W., and Pham, D. T. (2013a). Combining the Bees Algorithm and shape grammar to generate branded product concepts. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 227(12):1860–1873.

Ang, M. C., Ng, K. W., Pham, D. T., and Soroka, A. (2013b). Simulations of PCB assembly optimisation based on the bees algorithm with TRIZ-inspired operators. In *International Visual Informatics Conference*, volume 8237 LNCS, pages 335–346. Springer.

Ang, M. C., Pham, D., and Ng, K. (2009a). Minimum-time motion planning for a robot arm using the bees algorithm. In *2009 7th IEEE International Conference on Industrial Informatics*, pages 487–492.

Ang, M. C., Pham, D. T., Pham, D., Eldukhri, E., Soroka, A., Ang, M., and Ng, K. (2009b). Application of the Bees Algorithm with TRIZ-inspired operators for PCB assembly planning. In *Proceedings of 5 th Virtual International Conference on Intelligent Production Machines and Systems*.

Ang, M. C., Pham, D. T., Soroka, A. J., and Ng, K. W. (2010). PCB assembly optimisation using the bees algorithm enhanced with TRIZ operators. In *IECON Proceedings (Industrial Electronics Conference)*, pages 2708–2713.

Anh, V. M., Quan, T. Q., and Tran, P. (2021). Nonlinear vibration and geometric optimization of nanocomposite multilayer organic solar cell under wind loading. *Thin-Walled Structures*, 158:107199.

Antil, P., Singh, S., Singh, S., Prakash, C., and Pruncu, C. I. (2019). Metaheuristic approach in machinability evaluation of silicon carbide particle/glass fiber–reinforced polymer matrix composites during electrochemical discharge machining process. *Measurement and Control*, 52(7-8):1167–1176.

Archana, C. and Rejith, K. (2014a). Rate Adaptive Resource Allocation in Ofdma Using Bees Algorithm. *International Journal of Research in Engineering and Technology*, 03(15):14–18.

Archana, C. and Rejith, K. N. (2014b). The use of Bees Algorithm for RA and MA based resource allocation in OFDMA. In *1st International Conference on Computational Systems and Communications, ICCSC 2014*, volume December, pages 339–343.

Arif Şen, M., Bakircioğlu, V., and Kalyoncu, M. (2016). Performances comparison of the bees algorithm and genetic algorithm for PID controller tuning. *ACM International Conference Proceeding Series*, pages 126–130.

Arif Şen, M., Tinkir, M., and Kalyoncu, M. (2018). Optimisation of a pid controller for a two-floor structure under earthquake excitation based on the bees algorithm. *Journal of Low Frequency Noise, Vibration and Active Control*, 37(1):107–127.

Arzeha, N. A., Mustafa, M. W., and Idris, R. M. (2015). Lead Lag Controller of TCSC Optimized by Bees Algorithm for Damping Low Frequency Oscillation Enhancement in SMIB. *Applied Mechanics and Materials*, 781:374–378.

Arzeha, N. A., Mustafa, M. W., and Idris, R. M. (2018). Damping low frequency oscillations via facts-pod controllers tuned by bees algorithm. *ELEKTRIKA-Journal of Electrical Engineering*, 17(2):6–14.

Assareh, E., Behrang, M. A., Ghalambaz, M., Noghrehabadi, A. R., and Ghanbarzadeh, A. (2011). A New Approach to Solve Blasius Equation using Parameter Identification of Nonlinear Functions based on the Bees Algorithm (BA). *International Journal of Mechanical and Mechatronics Engineering*, 5(1):255–257.

References

Assareh, E. and Biglari, M. (2016). A Novel Approach to Capture the Maximum Power Generation from Wind Turbines Using Hybrid MLP Neural Network and Bees Algorithm (HNNBA). *IETE Journal of Research*, 62(3):368–378.

Atashpaz-Gargari, E. and Lucas, C. (2007). Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In *2007 IEEE congress on evolutionary computation*, pages 4661–4667. Ieee.

Attaran, B. and Ghanbarzadeh, A. (2012). The Improved Bees Algorithm by Using the Fuzzy Logic Controller. In *20th Annual International Iranian Mechanical Engineering Conference*, pages 16–19.

Attaran, B. and Ghanbarzadeh, A. (2015). Bearing fault detection based on maximum likelihood estimation and optimized ANN using the bees algorithm. *Journal of Applied and Computational Mechanics*, 1(1):35–43.

Attaran, B., Ghanbarzadeh, A., and Ansari-Asl, K. (2012). Two New Feature Extraction Methods for Intelligent Fault Diagnosis Based On Optimized Neural Networks and the Bees Algorithm. In *The 6th Condition Monitoring & Fault Diagnosis Conference*, volume February.

Attaran, B., Ghanbarzadeh, A., Zaeri, R., and Moradi, S. (2011). Intelligent fault diagnosis of rolling bearing based on optimized complementary capability features and rbf neural network by using the bees algorithm. In *The 2nd International Conference on Control, Instrumentation and Automation*, pages 764–769.

Attaran, B., Zarchi, M., and Toloei, A. (2017). Numerical survey of vibrational model for third aircraft based on hr suspension system actuator using two bee algorithm objective functions. *International Journal of Engineering*, 30(6):887–894.

Aungkulanon, P. (2016). Comparison of Bee Algorithm and Scheduling Methodologies: A Case Study of Manufacturing in Thailand. *Applied Mechanics and Materials*, 835:864–868.

Aungkulanon, P. and Luangpaiboon, P. (2012). Stochastic search mechanisms on the bee algorithm for optimising noisy multi-response surfaces. In *4th International Conference on Applied Operational Research*, volume Lecture No, pages 154–164.

Aw, M. Y., Mohamad, M. S., Chong, C. K., Deris, S., Remli, M. A., Ismail, M. A., Corchado, J. M., and Omatu, S. (2018). Parameter estimation of essential amino acids in arabidopsis thaliana using hybrid of bees algorithm and harmony search. In *International Conference on Practical Applications of Computational Biology & Bioinformatics*, pages 9–16. Springer.

Aydogdu, I. and AKIN, A. (2011). Bees algorithm based optimum design of open canal sections. *International Journal of Engineering and Applied Sciences*, 3(4):21–31.

Azarbad, M., Azami, H., Sanei, S., and Ebrahimzadeh, A. (2014). New neural network-based approaches for GPS GDOP classification based on neuro-fuzzy inference system, radial basis function, and improved bee algorithm. *Applied Soft Computing Journal*, 25:285–292.

Azfanizam, A. S., Pham, D. T., and Faieza, A. A. (2014). Combination of adaptive enlargement and reduction in the search neighbourhood in the bees algorithm. *Applied Mechanics and Materials*, 564:614–618.

Azman, Z., Sahari, N., Ngadiron, Z., Yi, S. S., Noh, F. H. M., Mustafa, F., Zambri, N. A., and Norjali, R. (2020). Optimal sizing of static var compensator using bees algorithm for cost minimization. *Journal of Electrical Power and Electronic Systems*, 2(1).

Azzeh, M. (2011a). Adjusted case-based software effort estimation using bees optimization algorithm. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 315–324.

Azzeh, M. (2011b). Software effort estimation based on optimized model tree. In *Proceedings of the 7th International Conference on Predictive Models in Software Engineering*.

Bahamish, H. A. A., Abdullah, R., and Salam, R. A. (2008). Protein conformational search using bees algorithm. In *Proceedings - 2nd Asia International Conference on Modelling and Simulation, AMS 2008*, pages 911–916.

Bakırcıoğlu, V., Şen, M. A., and Kalyoncu, M. (2016). Optimization of pid controller based on the bees algorithm for one leg of a quadruped robot. In *MATEC Web of Conferences*, volume 42, page 03004. EDP Sciences.

Banooni, S., Zarea, H., and Molana, M. (2014). Thermodynamic and Economic Optimization of Plate Fin Heat Exchangers Using the Bees Algorithm. *Heat Transfer - Asian Research*, 43(5):427–446.

Baronti, L., Castellani, M., and Pham, D. T. (2020a). An analysis of the search mechanisms of the bees algorithm. *Swarm and Evolutionary Computation*, 59:100746.

Baronti, L., Zhang, B., Castellani, M., and Pham, D. T. (2020b). Machine learning of electro-hydraulic motor dynamics. *SN Applied Sciences*, 2(1):1–12.

Baykasoğlu, A., Özbakir, L., and Tapkan, P. (2009). The bees algorithm for workload balancing in examination job assignment. *European Journal of Industrial Engineering*, 3(4):424–435.

Behrang, M. A., Assareh, E., Assari, M. R., and Ghanbarzadeh, A. (2011a). Total energy demand estimation in Iran using bees algorithm. *Energy Sources, Part B: Economics, Planning and Policy*, 6(3):294–303.

Behrang, M. A., Assareh, E., Assari, M. R., and Ghanbarzadeh, A. (2011b). Using bees algorithm and artificial neural network to forecast world carbon dioxide emission. *Energy Sources, Part A: Recovery, Utilization and Environmental Effects*, 33(19):1747–1759.

References

Bellman, R. (1962). Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM*, 9(1):61–63.

Berger-Tal, O. and Bar-David, S. (2015). Recursive movement patterns: review and synthesis across species. *Ecosphere*, 6(9):1–12.

Bernardino, A. M., Bernardino, E. M., Sánchez-Pérez, J. M., Gómez-Pulido, J. A., and Vega-Rodríguez, M. A. (2011). Efficient Load Balancing Using the Bees Algorithm. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 469–479.

Bernardino, E. M., Bernardino, A. M., Sánchez-Pérez, J. M., Gómez-Pulido, J. A., and Vega-Rodríguez, M. A. (2010). Using the bees algorithm to assign terminals to concentrators. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, number PART 2 in 6097 LNAI, pages 267–276.

Bernardino, E. M., Bernardino, A. M., Sanchez-Perez, J. M., Gomez-Pulido, J. A., and Vega-Rodriguez, M. A. (2012). Solving large-scale SONET network design problems using bee-inspired algorithms. *Optical Switching and Networking*, 9(2):97–117.

Bhasaputra, P., Anantasate, S., and Pattaraprakorn, W. (2011). Multiobjective bees algorithm for optimal power flow problem. *ECTI Transactions on Electrical Engineering, Electronics, and Communications*, 9(1):56–64.

Bilgic, H. H., Sen, M. A., and Kalyoncu, M. (2016). Tuning of LQR controller for an experimental inverted pendulum system based on the bees algorithm. *Journal of Vibroengineering*, 18(6):3684–3694.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308.

Bonab, M. B. and Hashim, S. Z. (2014). Improved k-means clustering with Harmonic-Bee algorithms. In *4th World Congress on Information and Communication Technologies, WICT 2014*, pages 332–337.

Bonab, M. B., Hashim, S. Z. M., Bazin, N. E. N., and Alsaedi, A. K. Z. (2015). An Effective Hybrid of Bees Algorithm and Differential Evolution Algorithm in Data Clustering. *Mathematical Problems in Engineering*, 2015.

Bradford Jr, D. and Hung, C. (2012). Pollen-Based Bee Algorithm for Data Clustering - A Computational Model. *Progress in Intelligent Computing and Applications*, 1(1):16–36.

Braiwish, N. Y., Anayi, F. J., Fahmy, A. A., and E, E. E. (2014). Design Optimisation of Permanent Magnet Synchronous Motor for Electric Vehicles Traction using the Bees Algorithm Effective methods for detection of

internal faults in transformers View project Innovative production machines and systems View project Desig. In *49th International Universities Power Engineering Conference*, volume September.

Braiwish, N. Y., Anayi, F. J., Fahmy, A. A., and Eldukhri, E. E. (2015). Design optimization comparison of BLPM traction motor using Bees and Genetic Algorithms. In *Proceedings of the IEEE International Conference on Industrial Technology*, volume June, pages 702–707.

Buatois, A. and Lihoreau, M. (2016). Evidence of trapline foraging in honeybees. *Journal of Experimental Biology*, 219(16):2426–2429.

Cabrera G., G., Cabrera, E., Soto, R., Rubio, L. J. M., Crawford, B., and Paredes, F. (2012). A hybrid approach using an Artificial Bee algorithm with mixed integer programming applied to a large-scale capacitated facility location problem. *Mathematical Problems in Engineering*, 2012.

Camazine, S. and Sneyd, J. (1991). A model of collective nectar source selection by honey bees: self-organization through simple rules. *Journal of theoretical Biology*, 149(4):547–571.

Castellani, M., Otri, S., and Pham, D. T. (2019). Printed circuit board assembly time minimisation using a novel bees algorithm. *Computers & Industrial Engineering*, 133:186–194.

Castellani, M., Pham, Q. T., and Pham, D. T. (2012). Dynamic optimisation by a modified bees algorithm. *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, 226(7):956–971.

Chai-ead, N., Aungkulanon, P., and Luangpaiboon, P. (2011). Bees and firefly algorithms for noisy non-linear optimisation problems. In *IMECS 2011 - International MultiConference of Engineers and Computer Scientists 2011*, volume 2, pages 1449–1454.

Chamazi, M. A. and Motameni, H. (2019). Finding suitable membership functions for fuzzy temporal mining problems using fuzzy temporal bees method. *Soft Computing*, 23(10):3501–3518.

Chaweshly, S. A. (2010). Proposal of mutation-based bees algorithm (mba) to solve traveling salesman & jobs scheduling problems. *Engineering and Technology Journal*, 28(19):5833–5843.

Cheng, M. Y. and Lien, L. C. (2012). A hybrid AI-based particle bee algorithm for facility layout optimization. *Engineering with Computers*, 28(1):57–69.

Chmiel, W. and Szwed, P. (2016). Bees algorithm for the quadratic assignment problem on cuda platform. In *Man–Machine Interactions 4*, pages 615–625. Springer.

References

Choon, Y. W., Mohamad, M. B. S., Deris, S., Illias, R. M., Chai, L. E., and Chong, C. K. (2013a). Using bees hill flux balance analysis (BHFBA) for in silico microbial strain optimization. In *Asian Conference on Intelligent Information and Database Systems*, pages 375–384.

Choon, Y. W., Mohamad, M. S., Deris, S., Chong, C. K., Chai, L. E., Ibrahim, Z., and Omatu, S. (2012). Identifying gene knockout strategies using a hybrid of bees algorithm and flux balance analysis for in silico optimization of microbial strains. In *Distributed Computing and Artificial Intelligence*, pages 371–378. Springer.

Choon, Y. W., Mohamad, M. S., Deris, S., Chong, C. K., Omatu, S., and Corchado, J. M. (2015). Gene knockout identification using an extension of bees hill flux balance analysis. *BioMed Research International*, 2015.

Choon, Y. W., Mohamad, M. S., Deris, S., and Illias, R. M. (2014a). A hybrid of bees algorithm and flux balance analysis (BAFBA) for the optimisation of microbial strains. *International Journal of Data Mining and Bioinformatics*, 10(2):225–238.

Choon, Y. W., Mohamad, M. S., Deris, S., Illias, R. M., Chai, L. E., and Chong, C. K. (2013b). Identifying gene knockout strategy using Bees Hill Flux Balance Analysis (BHFBA) for improving the production of ethanol in bacillus subtilis. In *Proceedings of International Symposium on Biomedical Data Infrastructure (BDI 2013)*, pages 117–126.

Choon, Y. W., Mohamad, M. S., Deris, S., Illias, R. M., Chong, C. K., and Chai, L. E. (2014b). A hybrid of bees algorithm and flux balance analysis with optknock as a platform for in silico optimization of microbial strains. *Bioprocess and biosystems engineering*, 37(3):521–532.

Choon, Y. W., Mohamad, M. S., Deris, S., Illias, R. M., Chong, C. K., Chai, L. E., Omatu, S., and Corchado, J. M. (2014c). Differential bees flux balance analysis with OptKnock for in silico microbial strains optimization. *PLoS ONE*, 9(7):1–13.

Çil, Z. A., Li, Z., Mete, S., and Özceylan, E. (2020). Mathematical model and bee algorithms for mixed-model assembly line balancing problem with physical human–robot collaboration. *Applied Soft Computing*, 93:106394.

Climer, S. and Zhang, W. (2006). Cut-and-solve: An iterative search strategy for combinatorial optimization problems. *Artificial Intelligence*, 170(8-9):714–738.

ÇOBAN, R. and ERÇİN, Ö. (2012). Multi-objective bees algorithm to optimal tuning of pid controller. *Çukurova Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi*, 27(2):13–26.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT press.

Cox, M. D. and Myerscough, M. R. (2003). A flexible model of foraging by a honey bee colony: the effects of individual behaviour on foraging success. *Journal of Theoretical Biology*, 223(2):179–197.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations research*, 6(6):791–812.

Dagher Al-Khwarizi, K. E. and Ibraheem Abdulkareem, A. (2016). On-Line Tuning Sliding Mode Controller Design for Nonlinear Inverted Pendulum System based on Bees Algorithm. *Eng. &Tech.Journal*, 34(8).

Danaei, H. and Khajezadeh, A. (2015). Optimal design of PID controller using new version of bee's algorithm for quarter-car active suspension system. *Academie Royale Des Sciences D Outre-Mer Bulletin Des Seances*, 4(4):119–125.

Daoud, S., Yalaoui, F., Amodeo, L., Chehade, H., and Duperray, P. (2012). A hybrid bees algorithm for solving a robotic assembly line balancing problem. *World Scientific Proc. Series on Computer Engineering and Information Science 7; Uncertainty Modeling in Knowledge Engineering and Decision Making - Proceedings of the 10th International FLINS Conf.*, 7:1275–1280.

Darwish, A. H., Joukhadar, A., and Kashkash, M. (2018). Using the bees algorithm for wheeled mobile robot path planning in an indoor dynamic environment. *Cogent Engineering*, 5(1):1426539.

Dat, N. D., Anh, V. M., Quan, T. Q., Duc, P. T., and Duc, N. D. (2020a). Nonlinear stability and optimization of thin nanocomposite multilayer organic solar cell using bees algorithm. *Thin-Walled Structures*, 149:106520.

Dat, N. D., Quan, T. Q., Tran, P., Lam, P. T., and Duc, N. D. (2020b). A first-principle study of nonlinear large amplitude vibration and global optimization of 3d penta-graphene plates based on the bees algorithm. *Acta Mechanica*, 231(9):3799–3823.

De la Torre Gutiérrez, H. and Pham, D. T. (2018). Identification of patterns in control charts for processes with statistically correlated noise. *International Journal of Production Research*, 56(4):1504–1520.

Deghbouch, H. and Debbat, F. (2021). A hybrid bees algorithm with grasshopper optimization algorithm for optimal deployment of wireless sensor networks. *Inteligencia Artificial*, 24(67):18–35.

Dereli, T. and Das, G. S. (2011). A hybrid 'bee (s) algorithm'for solving container loading problems. *Applied Soft Computing*, 11(2):2854–2862.

Dey, P., Akhil, V., and Laskar, A. (2019). Application of smartphone and model updating technique in structural health monitoring. *Arabian Journal for Science and Engineering*, 44(5):4819–4828.

References

Dhote, C. A., Thakare, A. D., and Chaudhari, S. M. (2013). Data clustering using particle swarm optimization and bee algorithm. In *4th International Conference on Computing, Communications and Networking Technologies, ICCCNT 2013*, volume 1, pages 2–6.

Dhurandher, S. K., Misra, S., Pruthi, P., Singhal, S., Aggarwal, S., and Woungang, I. (2011). Using bee algorithm for peer-to-peer file searching in mobile ad hoc networks. *Journal of Network and Computer Applications*, 34(5):1498–1508.

Dhurandher, S. K., Singhal, S., Aggarwal, S., Pruthi, P., Misra, S., and Woungang, I. (2009). A Swarm Intelligence-based P2P file sharing protocol using Bee Algorithm. In *2009 IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2009*, pages 690–696.

Diwold, K., Beekman, M., and Middendorf, M. (2010). Bee nest site selection as an optimization process. In *ALIFE*, pages 626–633.

Dorigo, M. and Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical computer science*, 344(2-3):243–278.

Dorigo, M. and Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1470–1477. IEEE.

Dorigo, M. and Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1):53–66.

Dorigo, M., Maniezzo, V., and Colorni, A. (1991). The ant system: An autocatalytic optimizing process. *Academia*.

Dorigo, M. and Stützle, T. (2019). Ant colony optimization: overview and recent advances. In *Handbook of metaheuristics*, pages 311–351. Springer.

Düenci, M., Aydemir, A., Esen, I., and Aydin, M. E. (2015). Creep modelling of polypropylenes using artificial neural networks trained with Bee algorithms. *Engineering Applications of Artificial Intelligence*, 45:71–79.

Ebrahimpoor, S., Kiarostami, V., Khosravi, M., Davallo, M., and Ghaedi, A. (2019). Bees metaheuristic algorithm with the aid of artificial neural networks for optimization of acid red 27 dye adsorption onto novel polypyrrole/srfe 12 o 19/graphene oxide nanocomposite. *Polymer Bulletin*, 76(12):6529–6553.

Ebrahimzadeh, A., Addeh, J., and Ranaee, V. (2013). Recognition of control chart patterns using an intelligent technique. *Applied Soft Computing Journal*, 13(5):2970–2980.

Ebrahimzadeh, A. and Mavaddati, S. (2014). A novel technique for blind source separation using bees colony algorithm and efficient cost functions. *Swarm and Evolutionary Computation*, 14:15–20.

Eesa, A. S., Brifcani, A. M. A., and Orman, Z. (2015a). A New DIDS Design Based on a Combination Feature Selection Approach. *International Journal of Computer and Information Engineering*, 9(8):1921–1925.

Eesa, A. S., Orman, Z., and Brifcani, A. M. A. (2015b). A new feature selection model based on ID3 and bees algorithm for intrusion detection system. *Turkish Journal of Electrical Engineering and Computer Sciences*, 23(2):615–622.

Eldukhri, E. E. and Kamil, H. G. (2013). Optimisation of swing-up control parameters for a robot gymnast using the Bees Algorithm. *Journal of Intelligent Manufacturing*, 26(5):1039–1047.

Engelbrecht, A. P. (2013). Particle swarm optimization: Global best or local best? In *2013 BRICS congress on computational intelligence and 11th Brazilian congress on computational intelligence*, pages 124–135. IEEE.

Ercin, O. and Coban, R. (2011). Comparison of the artificial bee colony and the bees algorithm for PID controller tuning. In *2011 International Symposium on INnovations in Intelligent SysTems and Applications*, pages 595–598.

Eusuff, M. M. and Lansey, K. E. (2003). Optimization of water distribution network design using the shuffled frog leaping algorithm. *Journal of Water Resources planning and management*, 129(3):210–225.

Fahmy, A. (2012). Using the Bees Algorithm to select the optimal speed parameters for wind turbine generators. *Journal of King Saud University - Computer and Information Sciences*, 24(1):17–26.

Fahmy, A. A., Kalyoncu, M., and Castellani, M. (2012). Automatic design of control systems for robot manipulators using the bees algorithm. *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, 226(4):497–508.

Falehi, A. D. (2019). Analysis and application of inventive selective harmonic elimination strategy to eliminate high order harmonic from asymmetrical multi-level inverter. *Multidimensional Systems and Signal Processing*, pages 1–19.

Falehi, A. D. and Rafiee, M. (2018). Lvrt/hvrt capability enhancement of dfig wind turbine using optimal design and control of novel piλdμ-amli based dvr. *Sustainable Energy, Grids and Networks*, 16:111–125.

Farajvand, M., Kiarostami, V., Davallo, M., and Ghaedi, A. (2018). Optimization of solvent terminated dispersive liquid–liquid microextraction of copper ions in water and food samples using artificial neural networks coupled bees algorithm. *Bulletin of environmental contamination and toxicology*, 100(3):402–408.

Farhan, A. A. and Bilal, S. (2011). A novel fast and robust digital image watermarking using Bee Algorithm. In *Proceedings of the 14th IEEE International Multitopic Conference 2011, INMIC 2011*, pages 82–86.

References

Farhang, P. and Mazlumi, K. (2014). Low-frequency power system oscillation damping using HBA-based coordinated design of IPFC and PSS output feedback controllers. *Transactions of the Institute of Measurement and Control*, 36(2):184–195.

Fellows, M. R., Fomin, F. V., Lokshtanov, D., Rosamond, F., Saurabh, S., and Villanger, Y. (2012). Local search: Is brute-force avoidable? *Journal of Computer and System Sciences*, 78(3):707–719.

Fenton, A. (2011). *The Bees Algorithm for the Vehicle Routing Problem*. PhD thesis, ResearchSpace@ Auckland.

Firdhous, M., Ghazali, O., and Hassan, S. (2011a). Applying bees algorithm for trust management in cloud computing. In *International Conference on Bio-Inspired Models of Network, Information, and Computing Systems*, volume 103 LNICST, pages 224–229.

Firdhous, M., Ghazali, O., Hassan, S., Harun, N. Z., and Abas, A. (2011b). Honey Bee Based Trust Management System for Cloud Computing. In *Proceedings of the 3rd International Conference on Computing and Informatics, ICOCI*, pages 327–332.

Firouzjah, K. G. (2018). Distribution network expansion based on the optimized protective distance of surge arresters. *IEEE Transactions on Power Delivery*, 33(4):1735–1743.

Fogel, D. B. (2006). *Evolutionary computation: toward a new philosophy of machine intelligence*, volume 1. John Wiley & Sons.

Fon, C. W. and Wong, K. Y. (2010). Investigating the performance of bees algorithm in solving quadratic assignment problems. *International Journal of Operational Research*, 9(3):241–257.

Fries, I. and Camazine, S. (2001). Implications of horizontal and vertical pathogen transmission for honey bee epidemiology. *Apidologie*, 32(3):199–214.

Frisch, K. v. (1993). *The dance language and orientation of bees*. Harvard University Press.

Furlan, M. M. and Santos, M. O. (2017). Bfo: a hybrid bees algorithm for the multi-level capacitated lot-sizing problem. *Journal of Intelligent Manufacturing*, 28(4):929–944.

Gao, X. Z., Wang, X., Zenger, K., and Wang, X. (2012). A bee foraging-based memetic Harmony Search method. In *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, pages 184–189.

Geem, Z. W., Kim, J. H., and Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. *simulation*, 76(2):60–68.

Ghaedi, A. M., Ghaedi, M., Vafaei, A., Iravani, N., Keshavarz, M., Rad, M., Tyagi, I., Agarwal, S., and Gupta, V. K. (2015a). Adsorption of copper (II) using modified activated carbon prepared from Pomegranate wood: Optimization by bee algorithm and response surface methodology. *Journal of Molecular Liquids*, 206:195–206.

Ghaedi, M., Ansari, A., Nejad, A. P., Ghaedi, A., Vafaei, A., and Habibi, M. H. (2015b). Artificial neural network and Bees Algorithm for removal of Eosin B using Cobalt Oxide Nanoparticle-activated carbon: Isotherm and Kinetics study. *Environmental Progress & Sustainable Energy*, 34(1):809–814.

Ghanbarzadeh, A. (2010). Neural network weight optimisation using the Bees Algorithm. In *ASME 2010 10th Biennial Conference on Engineering Systems Design and Analysis, ESDA2010*, pages 849–854.

Gharaei, A. and Jolai, F. (2018). A multi-agent approach to the integrated production scheduling and distribution problem in multi-factory supply chain. *Applied Soft Computing*, 65:577–589.

Gharaei, A. and Jolai, F. (2019). A branch and price approach to the two-agent integrated production and distribution scheduling. *Computers & Industrial Engineering*, 136:504–515.

Gharaei, A. and Jolai, F. (2021). An ernsga-iii algorithm for the production and distribution planning problem in the multiagent supply chain. *International Transactions in Operational Research*, 28(4):2139–2168.

Ghasemi, B., Sadeghi, A., Roghani, M., and Branch, S. T. (2015). The solution of multi-objective multimode resource-constrained project scheduling problem (rcpsp) with partial precedence relations by multi-objective bees algorithm. *SILVAE GENETICA (ISSN: 0037-5349)*, 57(1).

Ghiasi, M., Irani Jam, M., Teimourian, M., Zarrabi, H., and Yousefi, N. (2019). A new prediction model of electricity load based on hybrid forecast engine. *International Journal of Ambient Energy*, 40(2):179–186.

Ghodousi, M., Alesheikh, A. A., Saeidian, B., Pradhan, B., and Lee, C.-W. (2019). Evaluating citizen satisfaction and prioritizing their needs based on citizens' complaint data. *Sustainability*, 11(17):4595.

Gholami, J. and Mohammadi, S. (2018). A novel combination of bees and firefly algorithm to optimize continuous problems. In *2018 8th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 40–46. IEEE.

Gholipour, R., Khosravi, A., and Mojallali, H. (2012). Bees Algorithm Based Intelligent Backstepping Controller Tuning For Gyro System. *Journal of Mathematics and Computer Science*, 05(03):205–211.

Gholipour, R., Khosravi, A., and Mojallali, H. (2013). Parameter Estimation of Loranz Chaotic Dynamic System Using Bees Algorithm. *International Journal of Engineering*, 26(3(C)):257–262.

References

Gholipour, R., Khosravi, A., and Mojallali, H. (2015). Multi-objective optimal backstepping controller design for chaos control in a rod-type plasma torch system using Bees algorithm. *Applied Mathematical Modelling*, 39(15):4432–4444.

Girdenis, A., Vizgirda, I., Bukantas, E., and Wong, P. (2019). An optimal controller for facts devices based on fuzzy rules and bees algorithm. *Annals of Electrical and Electronic Engineering*, 2(4):14–20.

González-Islas, J. C., Bolaños-Rodríguez, E., Lezama-León, E., Solis-Galindo, A., and Muñiz-Molina, H. (2011). Failures Prediction of Pipelines Carrying Natural Gas Using Bees Algorithm. *Meeting Abstracts*, MA2011-02(21):1684.

Guney, K. and Onay, M. (2007a). Amplitude-only pattern nulling of linear antenna arrays with the use of bees algorithm. *Progress In Electromagnetics Research*, 70:21–36.

Guney, K. and Onay, M. (2007b). Bees algorithm for null synthesizing of linear antenna arrays by controlling only the element positions. *Neural Network World*, 17(2):153.

Guney, K. and Onay, M. (2008). Bees algorithm for design of dual-beam linear antenna arrays with digital attenuators and digital phase shifters. *International journal of RF and microwave computer-aided engineering*, 18(4):337–347.

Guney, K. and Onay, M. (2010). Bees algorithm for interference suppression of linear antenna arrays by controlling the phase-only and both the amplitude and phase. *Expert systems with Applications*, 37(4):3129–3135.

Guney, K. and Onay, M. (2011). Synthesis of thinned linear antenna arrays using bees algorithm. *Microwave And Optical Technology Letters*, 53(4):795–799.

Guney, K. and Onay, M. (2013). Bees algorithm for interference suppression of linear antenna arrays by controlling the positions of selected elements. *Journal of Communications Technology and Electronics*, 58(12):1147–1156.

Haddad, O. B., Afshar, A., and Marino, M. A. (2006). Honey-bees mating optimization (hbmo) algorithm: a new heuristic approach for water resources optimization. *water resources management*, 20(5):661–680.

Hadi, B., Khosravi, A., Ranjbar, A., and Sarhadi, P. (2015). A Novel Control Design for Rigid-Link Electrically-Driven Robot Manipulator Using RISE Feedback and Bees Algorithm. *Journal of Knowledge–based Engineering and Innovation*, 1:22–30.

Hadi, B., Khosravi, A., Ranjbar N., A., and Sarhadi, P. (2014). RISE Feedback Control Design for RLED Robot Manipulator Using Bees Algorithm. *Journal of Advances in Computer Research*, 5(4):53–65.

Hakimi, S. and Ebrahimzadeh, A. (2015). Digital modulation classification using the bees algorithm and probabilistic neural network based on higher order statistics. *International Journal of Information and Communication Technology Research*, 7(4):1–15.

Hameed, N., Othman, W., Wahab, A., and Alhady, S. (2019). Optimising pid controller using bees algorithm and firefly algorithm. *ROBOTIKA*, 1(1):22–27.

Hasanvand, M. (2019). Optimization of laminate's single lap joints by fsdt theory. *Welding in the World*, 63(5):1265–1273.

Hashem, S. H., Habeeb, S., and Khalil, B. M. (2013). Enhance Network Intrusion Detection System Using Bee Algorithm. *Journal of Madenat Alelem University College*, 5(1):108–116.

Heinrich, B. (1979). Resource heterogeneity and patterns of movement in foraging bumblebees. *Oecologia*, 40(3):235–245.

Held, M. and Karp, R. M. (1962). A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied mathematics*, 10(1):196–210.

Holland, J. H. (1975). Adaptation in natural and artificial systems, university of michigan press. *Ann arbor, MI*, 1(97):5.

Hoos, H. H. and Stützle, T. (2004). *Stochastic local search: Foundations and applications*. Elsevier.

Hussain, D. M. and Surendran, D. (2020). Content based image retrieval using bees algorithm and simulated annealing approach in medical big data applications. *Multimedia Tools and Applications*, 79(5):3683–3698.

Hussein, W. A., Abdullah, S. N. H. S., and Sahran, S. (2017a). The patch-levy-based bees algorithm applied to dynamic optimization problems. *Discrete Dynamics in Nature and Society*, 2017.

Hussein, W. A., Sahran, S., and Abdullah, S. N. H. S. (2013). A New Initialization Algorithm for Bees Algorithm. In *International Multi-Conference on Artificial Intelligence Technology (Communications in Computer and Information Science)*, volume 378 CCIS, pages 39–52.

Hussein, W. A., Sahran, S., and Abdullah, S. N. H. S. (2015). An Improved Bees Algorithm for Real Parameter Optimization. *International Journal of Advanced Computer Science and Applications*, 6(10):23–39.

Hussein, W. A., Sahran, S., and Abdullah, S. N. H. S. (2016). A fast scheme for multilevel thresholding based on a modified bees algorithm. *Knowledge-Based Systems*, 101:114–134.

References

Hussein, W. A., Sahran, S., and Sheikh Abdullah, S. N. H. (2014). Patch-Levy-based initialization algorithm for Bees Algorithm. *Applied Soft Computing Journal*, 23:104–121.

Hussein, W. A., Sahran, S., and Sheikh Abdullah, S. N. H. (2017b). The variants of the Bees Algorithm (BA): a survey. *Artificial Intelligence Review*, 47(1):67–121.

Idris, R. M., Khairuddin, A., and Mustafa, M. W. (2009a). Optimal Allocation of FACTS Devices for ATC Enhancement Using Bees Algorithm. *International Journal of Electrical and Computer Engineering*, 3(6):1295–1302.

Idris, R. M., Khairuddin, A., and Mustafa, M. W. (2010a). A Parallel Bees Algorithm for ATC enhancement in modern electrical network. In *AMS2010: Asia Modelling Symposium 2010 - 4th International Conference on Mathematical Modelling and Computer Simulation*, pages 450–455.

Idris, R. M., Khairuddin, A., and Mustafa, M. W. (2010b). Optimal allocation of FACTS devices in deregulated electricity market using bees algorithm. *WSEAS Transactions on Power Systems*, 5(2):108–119.

Idris, R. M., Khairuddin, A., and Mustafa, M. W. (2010c). The placement of FACTS devices in modern electrical network using Bees Algorithm. In *Proceedings of the 9th WSEAS International Conference on Applications of Electrical Engineering, AEE '10*, pages 40–45.

Idris, R. M., Kharuddin, A., and Mustafa, M. (2009b). Optimal choice of facts devices for atc enhancement using bees algorithm. In *2009 Australasian Universities Power Engineering Conference*, pages 1–6. IEEE.

Ilka, R., Alinejad-Beromi, Y., and Yaghobi, H. (2015). Geometry optimization of five-phase permanent magnet synchronous motors using Bees Algorithm. *Iranian Journal of Electrical and Electronic Engineering*, 11(4):345–353.

Imannezhad, R. and Avakh Darestani, S. (2018). Project scheduling problem with resource constraints and activities interruption using bees algorithm. *International Journal of Industrial Engineering & Production Research*, 29(3):277–291.

Ismail, A. (2019). Domino algorithm: A novel constructive heuristics for traveling salesman problem. In *IOP Conference Series: Materials Science and Engineering*, volume 528.

Ismail, A., Hartono, N., Zeybek, S., and Pham, D. (2020). Using the bees algorithm to solve combinatorial optimisation problems for tsplib. In *IOP Conference Series: Materials Science and Engineering*, volume 847, page 012027. IOP Publishing.

Ismail, A. H., Hartono, N., Zeybek, S., Caterino, M., and Jiang, K. (2021). Combinatorial bees algorithm for vehicle routing problem. In *Macromolecular Symposia*, volume 396, page 2000284. Wiley Online Library.

Jamhuri, J., Norizah, K., Hasmadi, M., and Siti, A. (2020). Timber transportation planning using bees algorithm. In *IOP Conference Series: Earth and Environmental Science*, volume 463, page 012171. IOP Publishing.

Jamil, M. and Yang, X.-S. (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194.

Jana, N. D., Sil, J., and Das, S. (2015). Improved Bees Algorithm for protein structure prediction using AB off-lattice model. In *International Conference on Soft Computing-MENDEL, Advances in Intelligent Systems and Computing*, volume 378, pages 39–52.

Jevtic, A., Gazi, P., Andina, D., and Jamshidi, M. (2010). Building a swarm of robotic bees. In *2010 World Automation Congress*, pages 1–6.

Jevtic, A., Gutiérrez, A., Andina, D., and Jamshidi, M. (2011). Distributed bees algorithm for task allocation in swarm of robots. *IEEE Systems Journal*, 6(2):296–304.

Johnson, D. S. and McGeoch, L. A. (1997). The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization*, 1(1):215–310.

Jones, K. O. and Bouffet, A. (2008). Comparison of Bees algorithm, ant Colony Optimisation and Particle Swarm Optimisation for PID controller tuning. In *Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing, CompSysTech'08*, pages 9–1.

Jung, S. H. (2003). Queen-bee evolution for genetic algorithms. *Electronics letters*, 39(6):575–576.

Kalami, M.-S. (2014). Electric Power Cable Fault Recognition via combination of wavelet transform and optimized artificial neural network by using bees algorithm. *International Journal of Mechatronics, Electrical and Computer Technology*, 4(10):1112–1132.

Kamaruddin, S. and Abd Latif, M. A. H. (2019). Application of the bees algorithm for constrained mechanical design optimisation problem. *International Journal of Engineering Materials and Manufacture*, 4(1):27–32.

Kamaruddin, S., Bahari, M., Pham, D., Hamzas, M., and Zakaria, S. (2019). Bees algorithm enhanced with nelder and mead method for numerical function optimisation. In *AIP Conference Proceedings*, volume 2129, page 020166. AIP Publishing LLC.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical report, Citeseer.

References

Karaboga, D. and Akay, B. (2009). Artificial bee colony (abc), harmony search and bees algorithms on numerical optimization. In *Innovative production machines and systems virtual conference*.

Karaboga, D. and Gorkemli, B. (2011). A combinatorial artificial bee colony algorithm for traveling salesman problem. In *2011 International Symposium on Innovations in Intelligent Systems and Applications*, pages 50–53. IEEE.

Karaboga, D. and Gorkemli, B. (2019). Solving traveling salesman problem by using combinatorial artificial bee colony algorithms. *International Journal on Artificial Intelligence Tools*, 28(01):1950004.

Karimi, S., Salamat, A., and Javadpour, S. (2016). Designing and optimizing of composite and hybrid drive shafts based on the bees algorithm. *Journal of Mechanical Science and Technology*, 30(4):1755–1761.

Karunakaran, V., Joseph, S. I., Teja, R., Suganthi, M., and Rajasekar, V. (2019). A wrapper based feature selection approach using bees algorithm for extreme rainfall prediction via weather pattern recognition through svm classifier. *International Journal of Civil Engineering and Technology (IJCIET)*, 10(1).

Kashkooli, F. M. and Nasir, K. (2016). Economic design of shell-and-tube heat exchangers using bees algorithm. In *2nd International Conference on Research in Engineering Science and Technology*, volume May.

Kataria, P. and Rupal, N. (2012). Mining spatial data & enhancing classification using bio-inspired approaches. *International Journal of Science and Research (IJSR)*, 3(2):1473–1479.

Kavousi, A., Vahidi, B., Salehi, R., Bakhshizadeh, M. K., Farokhnia, N., and Fathi, S. H. (2012). Application of the bee algorithm for selective harmonic elimination strategy in multilevel inverters. *IEEE Transactions on Power Electronics*, 27(4):1689–1696.

Kazemi, M., Heydari Shirazi, K., and Ghanbarzadeh, A. (2012). Optimization of semi-trailing arm suspension for improving handling and stability of passenger car. *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics*, 226(2):108–121.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE.

Keshavarznejad, M., Rezvani, M. H., and Adabi, S. (2021). Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms. *Cluster Computing*, pages 1–29.

Khademi-Zahedi, R. and Alimouri, P. (2019). Finite element model updating of a large structure using multi-setup stochastic subspace identification method and bees optimization algorithm. *Frontiers of Structural and Civil Engineering*, 13(4):965–980.

Khalaf, O. I., Abdulsahib, G. M., and Sabbar, B. M. (2020). Optimization of wireless sensor network coverage using the bee algorithm. *J. Inf. Sci. Eng.*, 36(2):377–386.

Khalid, N. S. A., Mustafa, M. W., and Idris, R. M. (2015). Optimal Parameters Tuning of Power System Stabilizer via Bees Algorithm. *Applied Mechanics and Materials*, 781:397–401.

Khang, N. T. T. M., Phuc, N. B., and Nuong, T. T. H. (2011). The Bees Algorithm for A Practical University Timetabling Problem in Vietnam. In *2011 IEEE International Conference on Computer Science and Automation Engineering*, pages 42–47.

Khanmirzaei, Z. (2010). Training recurrent neuro-fuzzy system using two novel population-based algorithms for temperature forecasting. In *Proceedings - 10th IEEE International Conference on Computer and Information Technology, CIT-2010, 7th IEEE International Conference on Embedded Software and Systems, ICESS-2010, ScalCom-2010*, pages 438–445.

Khanmirzaei, Z. and Teshnehlab, M. (2010). Prediction Using Recurrent Neural Network Based Fuzzy Inference system by the Modified Bees Algorithm. *International Journal of Advancements in Computing Technology*, 2(2):42–55.

Khazaei, S., Tahani, A., Yazdani-Asrami, M., and Gholamian, S. A. (2015). Optimal design of three phase surface mounted permanent magnet synchronous motor by particle swarm optimization and bees algorithm for minimum volume and maximum torque. *Journal of Advances in Computer Research*, 6(2):83–98.

Khodsuz, M. and Seyyedbarzegar, S. (2019a). The investigation of different pattern recognition performance as surge arrester fault classifier. *Journal of Intelligent & Fuzzy Systems*, 36(3):2845–2855.

Khodsuz, M. and Seyyedbarzegar, S. (2019b). Surge arrester monitoring under different operating conditions using bees-anfis. *Iranian Journal of Electrical and Electronic Engineering*, 15(1):151–160.

Khosravi, A., Addeh, J., and Ganjipour, J. (2011). Breast cancer detection using BA-BP based neural networks and efficient features. In *2011 7th Iranian Conference on Machine Vision and Image Processing, MVIP 2011 - Proceedings*, volume 1.

Khosravy Far, H. and Aghazadeh, F. (2015). Dynamic Cellular Learning Bees Algorithm For Optimization in Dynamic Environments. *International Journal of Research in Management & Technology-IRACST*, 5(1):2249–9563.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.

References

Klein, S., Pasquaretta, C., Barron, A. B., Devaud, J.-M., and Lihoreau, M. (2017). Inter-individual variability in the foraging behaviour of traplining bumblebees. *Scientific reports*, 7(1):1–12.

Koc, E. (2010). *Bees Algorithm: theory, improvements and applications*. Cardiff University.

Konstantinov, S., Diveev, A., Balandina, G., and Baryshnikov, A. (2019a). Comparative research of random search algorithms and evolutionary algorithms for the optimal control problem of the mobile robot. *Procedia Computer Science*, 150:462–470.

Konstantinov, S., Khamidova, U., and Sofronova, E. (2019b). A novel hybrid method of global optimization based on the grey wolf optimizer and the bees algorithm. *Procedia Computer Science*, 150:471–477.

Koo, C. L., Mohamad, M. S., Ornatu, S., Salleh, A. H. M., Deris, S., and Yoshioka, M. (2014). A gene knockout strategy for succinate production using a hybrid algorithm of bees algorithm and minimization of metabolic adjustment. In *2014 IEEE International Conference on Granular Computing (GrC)*, pages 131–136. IEEE.

Koza, J. R. (1992). Evolution of subsumption using genetic programming. In *Proceedings of the First European Conference on Artificial Life*, pages 110–119.

Krainyukov, A., Kutev, V., and Andreeva, E. (2014). Using bees algorithms for solution of radar pavement monitoring inverse problem. *Transport and Telecommunication*, 15(1):53–66.

Kyritsis, M., Gulliver, S. R., Feredoes, E., and Din, S. U. (2018). Human behaviour in the euclidean travelling salesperson problem: Computational modelling of heuristics and figural effects. *Cognitive Systems Research*, 52:387–399.

Laili, Y., Tao, F., Pham, D. T., Wang, Y., and Zhang, L. (2019). Robotic disassembly re-planning using a two-pointer detection strategy and a super-fast bees algorithm. *Robotics and Computer-Integrated Manufacturing*, 59:130–142.

Lambiase, A., Iannone, R., Miranda, S., Lambiase, A., and Pham, D. T. (2016). Bees algorithm for effective supply chains configuration. *International Journal of Engineering Business Management*, 8:1–9.

Lara, C., Flores, J. J., and Calderón, F. (2008). Solving a school timetabling problem using a bee algorithm. In *Mexican International Conference on Artificial Intelligence*, pages 664–674. Springer.

Law, A. M. (2013). A tutorial on how to select simulation input probability distributions. In *2013 Winter Simulations Conference (WSC)*, pages 306–320. IEEE.

Lee, J. S., Wang, J. W., and Giang, K. Y. (2014). A new image watermarking scheme using multi-objective bees algorithm. *Applied Mathematics and Information Sciences*, 8(6):2945–2953.

Lee, J. Y. and Darwish, A. H. (2008). Multi-objective environmental/economic dispatch using the bees algorithm with weighted sum. In *EKC2008 Proceedings of the EU-Korea Conference on Science and Technology*, pages 267–274.

Lee, J. Y. and Kim, J. M. (2010). Optimization methodology of low carbon mixed energy systems using the bees algorithm. In *EKC 2009 Proceedings of the EU-Korea Conference on Science and Technology, Springer Proceedings in Physics*, volume 135, pages 31–39.

Lee, J.-Y. and Oh, J.-S. (2009). The Bees Algorithm with Weighted Sum Using Memorized Zones for Multi-objective Problem. *Journal of the Korean Society of Marine Engineering*, 33(3):395–402.

Leeprechanon, N. and Phonrattanasak, P. (2013). Bees two-hive algorithm for optimal power flow. *Applied Mechanics and Materials*, 313-314(March):870–875.

Leeprechanon, N. and Polratanasak, P. (2010). Multiobjective bees algorithm with clustering technique for environmental/economic dispatch. In *ECTI-CON 2010 - The 2010 ECTI International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, pages 621–625.

Leitao, J., Gil, P., Ribeiro, B., and Cardoso, A. (2018). Application of bees algorithm to reduce household's energy costs via load deferment. In *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 100–105. IEEE.

Lenin, K., Reddy, B. R., and Surya Kalavathi, M. (2014). Dwindling of real power loss by using Improved Bees Algorithm. *International Journal of Recent Research in Electrical and Electronics Engineering (IJRREEE)*, 1(1):34–42.

Leong, K. H., Wang, C., Abdul-Rahman, H., Shavarebi, K., Boursier, P., and Loo, S.-C. (2020). Application of decision theory and bee-inspired method to railway system route optimization. *International Journal of Management Science and Engineering Management*, 15(1):59–69.

Leong, Y. Y., Chong, C. K., Choon, Y. W., Chai, L. E., Deris, S., Illias, R. M., Omatu, S., and Mohamad, M. S. (2012). Simulation of Fermentation Pathway Using Bees Algorithm. *ADCAIJ : Advances in Distributed Computing and Artificial Intelligence Journal*, 1(2):13–19.

Li, H., Liu, K., and Li, N. (2010a). Improved bees algorithm for the large-scale layout optimization without performance constraints. In *Proceedings 2010 IEEE 5th International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2010*, pages 459–463.

References

Li, H., Liu, K., and Li, X. (2010b). A comparative study of artificial bee colony, bees algorithms and differential evolution on numerical benchmark problems. In *Communications in Computer and Information Science*, volume 107 CCIS, pages 198–207.

Li, K., Liu, Q., Xu, W., Liu, J., Zhou, Z., and Feng, H. (2019). Sequence planning considering human fatigue for human-robot collaboration in disassembly. *Procedia CIRP*, 83:95–104.

Lien, L. C. and Cheng, M. Y. (2012). A hybrid swarm intelligence based particle-bee algorithm for construction site layout optimization. *Expert Systems with Applications*, 39(10):9642–9650.

Lien, L. C. and Cheng, M. Y. (2014). Particle bee algorithm for tower crane layout with material quantity supply and demand optimization. *Automation in Construction*, 45:25–32.

Lihoreau, M., Chittka, L., Le Comber, S. C., and Raine, N. E. (2012a). Bees do not use nearest-neighbour rules for optimization of multi-location routes. *Biology Letters*, 8(1):13–16.

Lihoreau, M., Raine, N. E., Reynolds, A. M., Stelzer, R. J., Lim, K. S., Smith, A. D., Osborne, J. L., and Chittka, L. (2012b). Radar tracking and motion-sensitive cameras on flowers reveal the development of pollinator multi-destination routes over large spatial scales. *PLoS biology*, 10(9):e1001392.

Lin, S. and Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516.

Liu, J., Xu, W., Zhou, Z., and Pham, D. T. (2020a). Scheduling of robotic disassembly in remanufacturing using bees algorithms. *Evolutionary Computation in Scheduling*, pages 257–298.

Liu, J., Zhou, Z., Pham, D. T., Xu, W., Cui, J., and Yang, C. (2020b). Service platform for robotic disassembly planning in remanufacturing. *Journal of Manufacturing Systems*, 57:338–356.

Liu, J., Zhou, Z., Pham, D. T., Xu, W., Ji, C., and Liu, Q. (2018a). Robotic disassembly sequence planning using enhanced discrete bees algorithm in remanufacturing. *International Journal of Production Research*, 56(9):3134–3151.

Liu, J., Zhou, Z., Pham, D. T., Xu, W., Ji, C., and Liu, Q. (2020c). Collaborative optimization of robotic disassembly sequence planning and robotic disassembly line balancing problem using improved discrete bees algorithm in remanufacturing. *Robotics and Computer-Integrated Manufacturing*, 61:101829.

Liu, J., Zhou, Z., Pham, D. T., Xu, W., Yan, J., Liu, A., Ji, C., and Liu, Q. (2018b). An improved multi-objective discrete bees algorithm for robotic disassembly line balancing problem in remanufacturing. *The International Journal of Advanced Manufacturing Technology*, 97(9-12):3937–3962.

Liu, Q., Liu, Z., Xu, W., Tang, Q., Zhou, Z., and Pham, D. T. (2019). Human-robot collaboration in disassembly for sustainable manufacturing. *International Journal of Production Research*, 57(12):4027–4044.

Lobo, F. G., Bazargani, M., and Burke, E. K. (2020). A cutoff time strategy based on the coupon collector's problem. *European Journal of Operational Research*.

Lones, M. A. (2020). Mitigating metaphors: A comprehensible guide to recent nature-inspired algorithms. *SN Computer Science*, 1(1):49.

Long, V. (2015). Application of a Pheromone-Based Bees Algorithm for Simultaneous Optimisation of Key Component Sizes and Control Strategy for Hybrid Electric Vehicles. *International Journal of Swarm Intelligence and Evolutionary Computation*, 04(01):1–7.

LONG, V. and NHAN, N. (2012). Bees-algorithm-based optimization of component size and control strategy parameters for parallel hybrid electric vehicles. *International journal of automotive technology*, 13(7):1177–1183.

Lu, W., Quan, Z., Liu, Q., Zhang, D., and Xu, W. (2015). QoE based spectrum allocation optimization using bees algorithm in cognitive radio networks. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 327–338.

Luangpaiboon, P. (2011). Bee parameter determination via weighted centriod modified simplex and constrained response surface optimisation methods. *International Journal of Industrial and Manufacturing Engineering*, 5(8):1578–1584.

Luantangsrisuk, V., Songmuang, P., and Kongkachandra, R. (2017). Automated Test Assembly with Minimum Redundant Questions Based on Bee Algorithm. *Proceedings - 12th International Conference on Signal Image Technology and Internet-Based Systems, SITIS 2016*, pages 652–656.

Luo, G. H., Huang, S. K., Chang, Y. S., and Yuan, S. M. (2014). A parallel Bees Algorithm implementation on GPU. *Journal of Systems Architecture*, 60(3):271–279.

MacRae, C. A., Ozlen, M., and Ernst, A. T. (2019). A hybrid benders decomposition and bees algorithm matheuristic approach to transmission expansion planning considering energy storage. *arXiv preprint arXiv:1903.01236*.

Mahmuddin, M. and Al-dawoodi, A. G. M. (2017). Experimental study of variation local search mechanism for bee algorithm feature selection. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 9(2-2):103–107.

References

Mahmuddin, M. and Yusof, Y. (2009). A Hybrid Simplex Search and Bio-Inspired Algorithm for Faster Convergence. In *2009 International Conference on Machine Learning and Computing*, pages 203–207.

Mahmuddin, M. and Yusof, Y. (2010). Automatic estimation total number of cluster using a hybrid test-and-generate and K-means algorithm. In *2010 International Conference on Computer Applications and Industrial Electronics*, pages 593–596.

Maleki, A. (2018). Design and optimization of autonomous solar-wind-reverse osmosis desalination systems coupling battery and hydrogen energy storage by an improved bee algorithm. *Desalination*, 435:221–234.

Malekzadeh, M., Khosravi, A., Alighale, S., and Azami, H. (2012). Optimization of orthogonal poly phase coding waveform based on bees algorithm and artificial bee colony for MIMO radar. In *International Conference on Intelligent Computing*, volume 7389 LNCS, pages 95–102.

Malik, M. (2012). Bees algorithm for degree-constrained minimum spanning tree problem. In *2012 National Conference on Computing and Communication Systems, NCCCS 2012*, pages 97–104. IEEE.

Manusov, V., Matrenin, P., and Khasanzoda, N. (2019). Swarm algorithms in dynamic optimization problem of reactive power compensation units control. *International Journal of Electrical & Computer Engineering (2088-8708)*, 9(5).

Martínez-Cagigal, V. and Hornero, R. (2017). A binary bees algorithm for p300-based brain-computer interfaces channel selection. In *International Work-Conference on Artificial Neural Networks*, pages 453–463. Springer.

Martino, G., Yuce, B., Iannone, R., and Packianather, M. S. (2016). Optimisation of the replenishment problem in the Fashion Retail Industry using Tabu-Bees algorithm. *IFAC-PapersOnLine*, 49(12):1685–1690.

Marzi, H., Haj Darwish, A., and Helfawi, H. (2017). Training anfis using the enhanced bees algorithm and least squares estimation. *Intelligent Automation & Soft Computing*, 23(2):227–234.

Masajedi, P., Shirazi, K. H., and Ghanbarzadeh, A. (2013). Verification of bee algorithm based path planning for a 6DOF manipulator using ADAMS. *Journal of Vibroengineering*, 15(2):805–815.

Masmoudi, M. A., Hosny, M., Braekers, K., and Dammak, A. (2016). Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. *Transportation Research Part E: Logistics and Transportation Review*, 96:60–80.

Massah, A., Zamani, A., Salehinia, Y., Sh, M. A., and Teshnehlab, M. (2013). A hybrid controller based on cpg and zmp for biped locomotion. *Journal of Mechanical science and technology*, 27(11):3473–3486.

Mastrocinque, E., Yuce, B., Lambiase, A., and Packianather, M. S. (2013). A multi-objective optimization for supply chain network using the bees algorithm. *International Journal of Engineering Business Management*, 5:38.

Mayteekrieangkrai, N. and Wongthatsanekorn, W. (2015). Optimized ready mixed concrete truck scheduling for uncertain factors using bee algorithm. *Songklanakarin Journal of Science and Technology*, 37(2):221–230.

Mayteekriengkrai, N. and Wongthatsanekorn, W. (2017). Ready mixed concrete trucks scheduling for minimising fuel cost of ready mixed concrete trucks from multiple plants to multiple sites using bee algorithm. *International Journal of Operational Research*, 28(3):327–347.

Mazitov, T., Božek, P., Abramov, A., Nikitin, Y., and Abramov, I. (2016). Using bee algorithm in the problem of mapping. *Procedia Engineering*, 149(June):305–312.

Mazraeh, H. D., Manesh, F. H., and Tabasi, S. H. (2013). Bee algorithm for solving an inverse parabolic system. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(4):410–424.

Mehdinejadiani, B. (2017). Estimating the solute transport parameters of the spatial fractional advection-dispersion equation using bees algorithm. *Journal of contaminant hydrology*, 203:51–61.

Mehdinejadiani, B., Naseri, A. A., Jafari, H., Ghanbarzadeh, A., and Baleanu, D. (2013). A mathematical model for simulation of a water table profile between two parallel subsurface drains using fractional derivatives. *Computers & Mathematics with Applications*, 66(5):785–794.

Metni, N. and Lahoud, J. (2013). Neuro-control robustness analysis of an inverted pendulum using the bee algorithm. *Applied Mechanics and Materials*, 339:3–9.

Mirjalili, S., Mirjalili, S. M., and Lewis, A. (2014). Grey wolf optimizer. *Advances in engineering software*, 69:46–61.

Mirshekari, E., Ghanbarzadeh, A., and Shirazi, K. H. (2016). Structure comparison and optimal design of 6-rus parallel manipulator based on kinematic and dynamic performances. *Latin American Journal of Solids and Structures*, 13(13):2414–2438.

Mizanadl, S. M. M. and Ardakani, M. D. (2012). Using the Bees Algorithm with the Boundary Elements Method to Solve the Inverse Problem of Transient Heat Conduction. *Int J Advanced Design and Manufacturing Technology*, 5(2):51–58.

Mohamad Idris, R., Khairuddin, A., and Mustafa, M. W. (2009). A multi-objective bees algorithm for optimum allocation of FACTS devices for restructured power system. In *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, pages 1–6. IEEE.

References

Mohammed, M. A. and Al-Khafaji, H. (2017). Maximal itemsets mining algorithm based on bees' algorithm. In *2017 Annual Conference on New Trends in Information & Communications Technology Applications (NTICT)*, pages 1–6. IEEE.

Mollabakhshi, N. and Eshghi, M. (2013). Combinational circuit design using bees algorithm. In *2013 IEEE Conference Anthology, ANTHOLOGY 2013*.

Mongkolkosol, P. and Luangpaiboon, P. (2011). Steepest descent algorithm to determine the proper levels of bees parameters on dynamic multi-zone dispatching problems. In *IMECS 2011 - International MultiConference of Engineers and Computer Scientists 2011*, number 3 in 2, pages 1417–1422.

Moradi, A., Nafchi, A. M., and Ghanbarzadeh, A. (2015). Multi-objective optimization of truss structures using the bee algorithm. *Scientia Iranica*, 22(5):1789–1800.

Moradi, A., Nafchi, A. M., Ghanbarzadeh, A., and Soodmand, E. (2011a). Optimization of linear and nonlinear full vehicle model for improving ride comfort vs. road holding with the Bees Algorithm. In *2011 IEEE Colloquium on Humanities, Science and Engineering, CHUSER 2011*, pages 17–22.

Moradi, A., Shirazi, K. h., Keshavarz, M., Falehi, A. d., and Moradi, M. (2014). Smart piezoelectric patch in non-linear beam: Design, vibration control and optimal location. *Transactions of the Institute of Measurement and Control*, 36(1):131–144.

Moradi, S., Fatahi, L., and Razi, P. (2010). Finite element model updating using bees algorithm. *Structural and Multidisciplinary Optimization*, 42(2):283–291.

Moradi, S. and Kargozarfard, M. H. (2013). On multiple crack detection in beam structures. *Journal of Mechanical Science and Technology*, 27(1):47–55.

Moradi, S., Razi, P., and Fatahi, L. (2011b). On the application of bees algorithm to the problem of crack detection of beam-type structures. *Computers and Structures*, 89(23-24):2169–2175.

Moussa, A. and El-Sheimy, N. (2010). Localization of wireless sensor network using Bees Optimization Algorithm. In *2010 IEEE International Symposium on Signal Processing and Information Technology, ISSPIT 2010*, pages 478–481.

Mugiira, E. and Ndolo, A. (2016). Optimal Placement of FACTS Devices to Improve Power Flow Using Bees Algorithm. *International Journal of Advanced Research in Computer Science*, 7(3):1–6.

Muhamad, Z., Mahmuddin, M., Nasrudin, M. F., and Sahran, S. (2011). Local Search Manoeuvres Recruitment in The Bees Algorithm. In *Proceeding of The 3rd International Conference on Computing and Informatics*, pages 43–48.

Naderian, M. (2014). Environmental Pollution Assessment using different intelligent techniques: A case study for Iran. *Bull. Env. Pharmacol. Life Sci*, 3(10):126–132.

Nafchi, A. M. and Moradi, A. (2011). Constrained multi-objective optimization problems in mechanical engineering design using Bees Algorithm. *Journal of Solid Mechanics*, 3(4):353–364.

Nafchi, A. M., Moradi, A., Ghanbarzadeh, A., Rezazadeh, A., and Soodmand, E. (2011). Solving engineering optimization problems using the bees algorithm. In *2011 IEEE Colloquium on Humanities, Science and Engineering, CHUSER 2011*, pages 162–166.

Nafchi, A. M., Moradi, A., Ghanbarzadeh, A., Yaghoubi, S., and Moradi, M. (2012). An Improved Bees Algorithm For Solving Optimization Mechanical Problems. In *20th Annual International Conference on Mechanical Engineering*, pages 1–5.

Nagy, D., Mihálydeák, T., and Aszalós, L. (2019). Different types of search algorithms for rough sets. *Acta Cybernetica*, 24(1):105–120.

Najm, I. A. and Hammash, N. M. (2015). Analytic Behavior of Parallel Bees Algorithm. *Research Journal of Applied Sciences, Engineering and Technology*, 11(6):661–665.

Nasrinpour, H. R., Bavani, A. M., and Teshnehlab, M. (2017). Grouped bees algorithm: a grouped version of the bees algorithm. *Computers*, 6(1):5.

Nebti, S. (2013). Bio-Inspired Algorithms for Color Image Segmentation. *International Journal of Computer Applications*, 73(18):11–16.

Nebti, S. and Boukerram, A. (2010). Handwritten digits recognition based on swarm optimization methods. In *2010 International conference on networked digital technologies, Communications in Computer and Information Science*, number PART 1 in 87 CCIS, pages 45–54.

Nebti, S. and Boukerram, A. (2013). Handwritten characters recognition based on nature-inspired computing and neuro-evolution. *Applied Intelligence*, 38(2):146–159.

Nelson, K. and Wille, L. (1995). Comparative study of heuristics for optimal printed circuit board assembly. In *Proceedings of Southcon'95*, pages 322–327. IEEE.

Nemmich, M. A., Debbat, F., and Slimane, M. (2018a). A data clustering approach using bees algorithm with a memory scheme. In *International Conference on Computer Science and its Applications*, pages 261–270. Springer.

References

Nemmich, M. A., Debbat, F., and Slimane, M. (2018b). A novel hybrid firefly bee algorithm for optimization problems. *International Journal of Organizational and Collective Intelligence (IJOCI)*, 8(4):21–46.

Nemmich, M. A., Debbat, F., and Slimane, M. (2019). A permutation-based bees algorithm for solving resource-constrained project scheduling problem. *International Journal of Swarm Intelligence Research (IJSIR)*, 10(4):1–24.

Nemmich, M. A., Debbat, F., and Slimane, M. (2020a). Hybrid bees approach based on improved search sites selection by firefly algorithm for solving complex continuous functions. *Informatica*, 44(2).

Nemmich, M. A., Debbat, F., and Slimane, M. (2020b). Hybridizing bees algorithm with firefly algorithm for solving complex continuous functions. *International Journal of Applied Metaheuristic Computing (IJAMC)*, 11(2):27–55.

Nguyen, D. H. (2015). A Hybrid SFL-Bees Algorithm. *International Journal of Computer Applications*, 128(5):13–18.

Nguyen, K., Nguyen, P., and Tran, N. (2012). A hybrid algorithm of harmony search and bees algorithm for a university course timetabling problem. *International Journal of Computer Science Issues (IJCSI)*, 9(1):12.

Ohashi, K., Leslie, A., and Thomson, J. D. (2008). Trapline foraging by bumble bees: V. effects of experience and priority on competitive performance. *Behavioral Ecology*, 19(5):936–948.

Ohashi, K. and Thomson, J. D. (2012). Trapline foraging by bumble bees: Vi. behavioral alterations under speed–accuracy trade-offs. *Behavioral ecology*, 24(1):182–189.

Ohashi, K. and Thomson, J. D. (2013). Trapline foraging by bumble bees: Vi. behavioral alterations under speed–accuracy trade-offs. *Behavioral ecology*, 24(1):182–189.

Ohashi, K., Thomson, J. D., and D'souza, D. (2007). Trapline foraging by bumble bees: Iv. optimization of route geometry in the absence of competition. *Behavioral Ecology*, 18(1):1–11.

Osaba, E., Carballedo, R., Diaz, F., Onieva, E., Masegosa, A. D., and Perallos, A. (2018). Good practice proposal for the implementation, presentation, and comparison of metaheuristics for solving routing problems. *Neurocomputing*, 271:2–8.

Osamy, W., Khedr, A. M., Aziz, A., and El-Sawy, A. A. (2018). Cluster-tree routing based entropy scheme for data gathering in wireless sensor networks. *IEEE Access*, 6:77372–77387.

Osman, A., KALYONCU, M., and HASSAN, A. (2018). The bees'algorithm for design optimization of a gripper mechanism. *Selçuk-Teknik Dergisi*, pages 69–86.

Osman, I. H. and Laporte, G. (1996). Metaheuristics: A bibliography.

Othman, N., Musirin, I., Rahim, M. A., and Othman, Z. (2010). Bees algorithm technique for loss minimization in power transmission network using static var compensator. In *PEOCO 2010 - 4th International Power Engineering and Optimization Conference, Program and Abstracts*, pages 164–169. IEEE.

Otri, S. (2011). *Improving the bees algorithm for complex optimisation problems*. PhD thesis, Cardiff University.

Özbakir, L., Baykasoğlu, A., and Tapkan, P. (2010). Bees algorithm for generalized assignment problem. *Applied Mathematics and Computation*, 215(11):3782–3795.

Ozbakir, L. and Tapkan, P. (2010). Balancing fuzzy multi-objective two-sided assembly lines via Bees Algorithm. *Journal of Intelligent and Fuzzy Systems*, 21(5):317–329.

Ozbakir, L. and Tapkan, P. (2011). Bee colony intelligence in zone constrained two-sided assembly line balancing problem. *Expert Systems with Applications*, 38(9):11947–11957.

Oztemel, E. and Selam, A. A. (2017). Bees algorithm for multi-mode, resource-constrained project scheduling in molding industry. *Computers & Industrial Engineering*, 112:187–196.

Packianather, M., Landy, M., and Pham, D. (2009). Enhancing the speed of the bees algorithm using pheromone-based recruitment. In *2009 7th IEEE International Conference on Industrial Informatics*, pages 789–794. IEEE.

Packianather, M. S., Al-Musawi, A. K., and Anayi, F. (2019). Bee for mining (b4m)–a novel rule discovery method using the bees algorithm with quality-weight and coverage-weight. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 233(14):5101–5112.

Packianather, M. S., Eaton, M., Papadopoulos, I., and Alexopoulos, T. (2018). Optimizing the number of acoustic emission sensors using the bees algorithm for detecting surface fractures. *Procedia CIRP*, 67:362–367.

Packianather, M. S. and Kapoor, B. (2015). A wrapper-based feature selection approach using Bees Algorithm for a wood defect classification system. In *10th System of Systems Engineering Conference, SoSE 2015*, pages 498–503. IEEE.

Packianather, M. S., Yuce, B., Mastrocinque, E., Fruggiero, F., Pham, D. T., and Lambiase, A. (2014). Novel Genetic Bees Algorithm applied to single machine scheduling problem. In *2014 World Automation Congress Proceedings*, pages 906–911.

Parsa, H. R., AsgharGholamian, S., and Abbasi, M. (2013). Design and Optimization of Eddy Current Testing Probe Using Bees Algorithm and Finite Element Analysis. *International Journal of Modern Education and Computer Science*, 5(12):40–46.

References

Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE control systems magazine*, 22(3):52–67.

Paul, S., Müller, H., Preiser, R., de Lima Neto, F. B., Marwala, T., and De Wilde, P. (2014). Developing a management decision-making model based upon a complexity perspective with reference to the Bee Algorithm. *Emergence: Complexity and Organization*, 16(4).

Pham, D. and Darwish, A. H. (2008a). Fuzzy selection of local search sites in the bees algorithm. In *Proceedings of the 4th Virtual International Conference on Intelligent Production Machines and Systems*, pages 1–14.

Pham, D. and Ghanbarzadeh, A. (2007). Multi-objective optimisation using the bees algorithm. In *3rd International Virtual Conference on Intelligent Production Machines and Systems*, page 6.

Pham, D., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., and Zaidi, M. (2005). The bees algorithm. In *Technical Note, Manufacturing Engineering Centre, Cardiff University, UK*.

Pham, D. and Karaboga, D. (2012). *Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks*. Springer Science & Business Media.

Pham, D., Koc, E., Lee, J., and Phrueksanant, J. (2007a). Using the bees algorithm to schedule jobs for a machine. In *Proceedings of Eighth International Conference on Laser Metrology, CMM and Machine Tool Performance*, pages 430–439.

Pham, D., Muhamad, Z., Mahmuddin, M., Ghanbarzadeh, A., Koc, E., and Otri, S. (2007b). Using the bees algorithm to optimise a support vector machine for wood defect classification. In *Memorias del Innovative Production Machines and Systems Virtual Conference*.

Pham, D., Otri, S., Ghanbarzadeh, A., and Koc, E. (2006a). Application of the Bees Algorithm to the Training of Learning Vector Quantisation Networks for Control Chart Pattern Recognition. In *IEEE 2th International Conference Information and Communication Technologies: From Theory to Applications*, pages 1624–1629.

Pham, D., Pham, Q., Ghanbarzadeh, A., and Castellani, M. (2008a). Dynamic optimisation of chemical engineering processes using the bees algorithm. *IFAC Proceedings Volumes*, 41(2):6100–6105.

Pham, D. T., Afify, A., and Koc, E. (2007c). Manufacturing cell formation using the bees algorithm. In *Innovative Production Machines and Systems Virtual Conference, Cardiff, UK*.

Pham, D. T., Al-Jabbouli, H., Mahmuddin, M., Otri, S., and Darwish, A. H. (2008b). Application of the bees algorithm to fuzzy clustering. In *Proc. of the 4th International Conference on Evolutionary Computation and Global Optimization*.

Pham, D. T., Ang, M. C., Ng, K. W., Otri, S., and Darwish, H. a. (2008c). Generating branded product concepts: comparing the Bees Algorithm and an evolutionary algorithm. In *4th International Conference on Intelligent Production Machines and Systems*, pages 398–403.

Pham, D. T., Baronti, L., Zhang, B., and Castellani, M. (2018). Optimisation of engineering systems with the bees algorithm. *International Journal of Artificial Life Research (IJALR)*, 8(1):1–15.

Pham, D. T. and Castellani, M. (2009). The bees algorithm modelling foraging behaviour to solve continuous optimization problems. *Proceedings of the Institution of Mechanical Engineers, Part C Journal of Mechanical Engineering Science*, 223(12):2919–2938.

Pham, D. T. and Castellani, M. (2014). Benchmarking and comparison of nature-inspired population-based continuous optimisation algorithms. *Soft Computing*, 18(5):871–903.

Pham, D. T. and Castellani, M. (2015). A comparative study of the bees algorithm as a tool for function optimisation. *Cogent Engineering*, 2(1):1091540.

Pham, D. T., Castellani, M., and Fahmy, A. A. (2008d). Learning the inverse kinematics of a robot manipulator using the Bees Algorithm. In *IEEE International Conference on Industrial Informatics (INDIN)*, pages 493–498.

Pham, D. T., Castellani, M., and Ghanbarzadeh, A. (2007d). Preliminary design using the bees algorithm. In *Laser Metrology and Machine Performance VIII - 8th International Conference and Exhibition on Laser Metrology, Machine Tool, CMM and Robotic Performance, LAMDAMAP 2007*, pages 420–429.

Pham, D. T., Castellani, M., and Le-Thi, H. A. (2013). The Bees Algorithm: Modelling nature to solve complex optimisation problems. In *International Conference on Manufacturing Research (ICMR2013)*, number 1 in September, pages 481–488.

Pham, D. T., Castellani, M., and Le Thi, H. A. (2014). Nature-inspired intelligent optimisation using the bees algorithm. In *Transactions on Computational Intelligence XIII*, pages 38–69. Springer.

Pham, D. T., Castellani, M., Sholedolu, M., and Ghanbarzadeh, A. (2008e). The bees algorithm and mechanical design optimisation. In *ICINCO 2008 - Proceedings of the 5th International Conference on Informatics in Control, Automation and Robotics*, volume ICSO, pages 250–255.

Pham, D. T. and Darwish, A. H. (2008b). Fuzzy Selection of Local Search Sites in the Bees Algorithm. In *4th International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2008)*, pages 1–14.

References

Pham, D. T. and Darwish, A. H. (2009). Optimising fuzzy membership functions using the Bees Algorithm with Kalman filtering. In *Proceedings 5th International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2009)*, pages 328–333.

Pham, D. T. and Darwish, A. H. (2010). Using the bees algorithm with Kalman filtering to train an artificial neural network for pattern classification. *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, 224(7):885–892.

Pham, D. T., Darwish, A. H., and Eldukhri, E. E. (2009a). Optimisation of a fuzzy logic controller using the Bees Algorithm. *International Journal of Computer Aided Engineering and Technology*, 1(2):250–264.

Pham, D. T., Darwish, A. H., Eldukhri, E. E., and Otri, S. (2007e). Using the Bees Algorithm to tune a fuzzy logic controller for a robot gymnast. In *Proceedings of IPROMS 2007 conference, Cardiff, UK*.

Pham, D. T., Ghanbarzadeh, A., Koc, E., Ghanbarzadeh, A., and Otri, S. (2006b). Application of the Bees Algorithm to the Training of Radial Basis Function Networks for Control Chart Pattern Recognition. In *5th CIRP international seminar on intelligent computation in manufacturing engineering (CIRP ICME'06)*, volume January.

Pham, D. T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S., and Zaidi, M. (2006c). The bees algorithm—a novel tool for complex optimisation problems. In *Intelligent Production Machines and Systems*, pages 454–459. Elsevier.

Pham, D. T., Ghanbarzadeh, A., and Otri, S. (2006d). Optimisation of the weights of multi-layered perceptrons using the bees algorithm. In *Proceeding of 5th International Symposium on Intelligent Manufacturing Systems*, number 1 in May, pages 38–46.

Pham, D. T., Ghanbarzadeh, A., Otri, S., and Koç, E. (2009b). Optimal design of mechanical components using the bees algorithm. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 223(5):1051–1056.

Pham, D. T. and Kalyoncu, M. (2009). Optimisation of a fuzzy logic controller for a flexible single-link robot arm using the bees algorithm. In *IEEE International Conference on Industrial Informatics (INDIN)*, pages 475–480.

Pham, D. T. and Koç, E. (2010). Design of a Two-dimensional Recursive Filter Using the Bees Algorithm. *International Journal of Automation and Computing*, 7(3):399–402.

Pham, D. T., Koç, E., Kalyoncu, M., and Tınkır, M. (2008f). Hierarchical PID Controller Design for a Flexible Link Robot Manipulator Using the Bees Algorithm. In *Proceedings of 6th International Symposium on Intelligent Manufacturing Systems, October 14-16, 2008*, page 32.

Pham, D. T., Koç, E., Lee, J. Y., and Phrueksanant, J. (2007f). Using the bees algorithm to schedule jobs for a machine. In *Laser Metrology and Machine Performance VIII - 8th International Conference and Exhibition on Laser Metrology, Machine Tool, CMM and Robotic Performance, LAMDAMAP 2007*, pages 430–439.

Pham, D. T., Lee, J. Y., Darwish, A. H., and Soroka, A. J. (2008g). Multi-objective Environmental/Economic Power Dispatch using the Bees Algorithm with Pareto optimality. In *4th International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2008a)*.

Pham, D. T., Mahmuddin, M., Otri, S., and Al-Jabbouli, H. (2007g). Application of the Bees Algorithm to the Selection Features for Manufacturing Data. In *Innovative Production Machines and Systems: Proceedings of the 3rd International Virtual Conference on Intelligent Production Machines and Systems*, pages 517–522.

Pham, D. T., Marzi, H., Marzi, A., Marzi, E., Darwish, A. H., and Lee, J. Y. (2010). Using grid computing to accelerate optimization solution: A system of systems approach. In *2010 5th International Conference on System of Systems Engineering, SoSE 2010*, pages 1–6.

Pham, D. T., Muhamad, Z., Mahmuddin, M., Ghanbarzadeh, A., Koc, E., and Otri, S. (2007h). Using the Bees Algorithm to Optimise a Support Vector Machine for Wood Defect Classification. In *3rd International Virtual Conference on Intelligent Production Machines and Systems*.

Pham, D. T., Negm, M. A., and Otri, S. (2008h). Using the Bees Algorithm to solve a stochastic optimisation problem. In *4th International Virtual Conference on Intelligent Production Machines and Systems (IPROMS)*.

Pham, D. T., Otri, S., Afify, A. A., Mahmuddin, M., and Al-Jabbouli, H. (2007i). Data Clustering Using the Bees Algorithm. In *40th CIRP international manufacturing systems seminar*.

Pham, D. T., Otri, S., and Darwish, A. H. (2007j). Application of the Bees Algorithm to PCB assembly optimisation. In *3rd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007)*, pages 511–516.

Pham, D. T., Soroka, A. J., Ghanbarzadeh, A., Koc, E., Otri, S., and Packianather, M. (2006e). Optimising neural networks for identification of wood defects using the bees algorithm. In *2006 IEEE International Conference on Industrial Informatics, INDIN'06*, pages 1346–1351.

Pham, D. T., Suarez-Alvarez, M. M., and Prostov, Y. I. (2011). Random search with k-prototypes algorithm for clustering mixed datasets. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 467(2132):2387–2403.

Phan, H. D., Ellis, K., Barca, J. C., and Dorin, A. (2020). A survey of dynamic parameter setting methods for nature-inspired swarm intelligence algorithms. *Neural Computing and Applications*, 32(2):567–588.

References

Phonrattanasak, P. (2011). Optimal placement of wind farm on the power system using multiobjective Bees algorithm. In *Proceedings of the World Congress on Engineering 2011, WCE 2011*, volume 2, pages 1414–1418.

Phonrattanasak, P. and Leeprechanon, N. (2016). Multiobjective optimal placement of public fast charging station on power distribution network using hybrid ant colony optimization and bees algorithm. *International Journal of Engineering and Technology*, 8(6):2431–2442.

Phonrattanasak, P., Miyatake, M., and Sakamoto, O. (2013). Optimal location and sizing of solar farm on Japan east power system using multiobjective Bees algorithm. In *2013 IEEE Energytech*. IEEE.

Phrueksanant, J. (2013). *Machine scheduling using the Bees Algorithm*. PhD thesis, Cardiff University.

Phuc, N. B., Khang, N. T. T. M., and Nuong, T. T. H. (2011a). A new hybrid GA-Bees algorithm for a real-world university timetabling problem. In *Proceedings - 2011 International Conference on Intelligent Computation and Bio-Medical Instrumentation, ICBMI 2011*, pages 321–326.

Phuc, N. B., Khang, N. T. T. M., and Nuong, T. T. H. (2011b). A new hybrid ga-bees algorithm for a real-world university timetabling problem. In *2011 International Conference on Intelligent Computation and Bio-Medical Instrumentation*, pages 321–326. IEEE.

Polratanasuk, P., Mesacharoenwong, P., Anantasate, S., and Leeprechanon, N. (2010). Solving optimal power flow problem using parallel Bee algorithm. *Proceeding of the IEEE Selected Topics in Power Systems and Remote Sensing*, pages 60–64.

Pourkamalianaraki, M. and Sadeghi, M. (2016). Honey bee-inspired algorithms for SNP haplotype reconstruction problem. *Journal of Experimental and Theoretical Artificial Intelligence*, 28(1-2):201–214.

Rabbani, M., Navazi, F., Eskandari, N., and Farrokhi-Asl, H. (2020). A green transportation location-inventory-routing problem by dynamic regional pricing. *Journal of Industrial Engineering and Management Studies*, 7(1):35–58.

Rajagopalan, S., Thippana, V. C., and Parimi, A. M. (2018). Optimal placement of the interline power flow controller using the bees algorithm to minimize power loss. In *2018 4th International Conference on Electrical Energy Systems (ICEES)*, pages 212–217. IEEE.

Rajasekhar, A., Lynn, N., Das, S., and Suganthan, P. N. (2017). Computing with the collective intelligence of honey bees–a survey. *Swarm and Evolutionary Computation*, 32:25–48.

Ramesh, S. (2018). Protected reliable routing for manet using bees algorithm. *Taga Journal*.

Ramirez, F. J., Lee, J. Y., Packianather, M. S., and Pham, D. T. (2010). Enhancing multi-stage deep-drawing processes through the novel use of the Bees-Algorithm. In *6th IPROMS Virtual Conference*, pages 1–6.

Ramlie, F., Jamaludin, K. R., Dolah, R., and Muhamad, W. Z. A. W. (2016). Optimal feature selection of Taguchi character recognition in the Mahalanobis-Taguchi system using bees algorithm. *Global Journal of Pure and Applied Mathematics*, 12(3):2651–2671.

Ramlie, F., Muhamad, W. Z. A. W., Jamaludin, K. R., Cudney, E., and Dollah, R. (2020). A significant feature selection in the mahalanobis taguchi system using modified-bees algorithm. *International Journal of Engineering Research and Technology*.

Rao, R. V., Savsani, V. J., and Vakharia, D. (2011). Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, 43(3):303–315.

Rashedi, E., Nezamabadi-Pour, H., and Saryazdi, S. (2009). Gsa: a gravitational search algorithm. *Information sciences*, 179(13):2232–2248.

Rashtchi, V., Gholinezhad, J., and Farhang, P. (2010). Optimal coordination of overcurrent relays using Honey Bee Algorithm. In *2010 International Congress on Ultra Modern Telecommunications and Control Systems and Workshops, ICUMT 2010*, pages 401–405. IEEE.

Reinelt, G. (1991). Tsplib—a traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384.

Reinelt, G. (1994). *The traveling salesman: computational solutions for TSP applications*. Springer-Verlag.

Riff, M.-C. and Montero, E. (2013). A new algorithm for reducing metaheuristic design effort. In *2013 IEEE Congress on Evolutionary Computation*, pages 3283–3290. IEEE.

Rodríguez, S., Sánchez, N., and Gómez, D. (2019). Optimization of geometric parameters of power transformer using bee's algorithm. *Annals of Electrical and Electronic Engineering*, 2(7):7–10.

Rufai, K. I., Muniyandi, R. C., and Othman, Z. A. (2014). Improving bee algorithm based feature selection in intrusion detection system using membrane computing. *Journal of Networks*, 9(3):523–529.

Ruz, G. A. and Goles, E. (2012). Reconstruction and update robustness of the mammalian cell cycle network. In *2012 IEEE Symposium on Computational Intelligence and Computational Biology, CIBCB 2012*, pages 397–403.

Ruz, G. A. and Goles, E. (2013). Learning gene regulatory networks using the bees algorithm. *Neural Computing and Applications*, 22(1):63–70.

References

Ruz, G. A., Goles, E., Montalva, M., and Fogel, G. B. (2014). Dynamical and topological robustness of the mammalian cell cycle network: A reverse engineering approach. *BioSystems*, 115(1):23–32.

Saad, E., Awadalla, M., and Darwish, R. (2008). A data gathering algorithm for a mobile sink in large-scale sensor networks. In *2008 The Fourth International Conference on Wireless and Mobile Communications*, pages 207–213. IEEE.

Sabah Al-Araji, A. (2016). Development of a Swing-Tracking Sliding Mode Controller Design for Nonlinear Inverted Pendulum System via Bees-Slice Genetic Algorithm. *Eng. &Tech.Journal*, 34(15):7982.

Sabri, A. N., Radzi, N. H. M., and Samah, A. A. (2018). A study on bee algorithm and a algorithm for pathfinding in games. In *2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, pages 224–229. IEEE.

Sabzi, S., Pourdarbani, R., Kalantari, D., and Panagopoulos, T. (2020). Designing a fruit identification algorithm in orchard conditions to develop robots using video processing and majority voting based on hybrid artificial neural network. *Applied Sciences*, 10(1):383.

Sadeghi, A. and Alahyari, A. (2013). Bi-Objective MRCPCP problem with Bees Metaheuristic Algorithm. *International Research Journal of Applied and Basic Sciences*, 4(8):2165–2170.

Sadeghi, A., Kalanaki, A., Noktehdan, A., Samghabadi, A. S., and Barzinpour, F. (2011). Using bees algorithm to solve the resource constrained project scheduling problem in PSPLIB. In *International Conference on Theoretical and Mathematical Foundations of Computer Science*, pages 486–494.

Sadiq, A. T., Duaimi, M. G., and Shaker, S. A. (2012). Data missing solution using rough set theory and swarm intelligence. In *Proceedings - 2012 International Conference on Advanced Computer Science Applications and Technologies, ACSAT 2012*, pages 173–180.

Sadiq, A. T. and Hamad, A. G. (2010). Bsa: a hybrid bees' simulated annealing algorithm to solve optimization & np-complete problems. *Engineering & Technology Journal*, 28(2):271–281.

Sadiq AlObaidi, A. T. and Hamad, A. G. (2012). Exploration-Balanced Bees Algorithms to Solve Optimization and NP-Complete Problems. *International Journal of Research and Reviews in Soft and Intelligent Computing (IJRRSIC)*, 2(1).

Saeidian, B., Mesgari, M. S., and Ghodousi, M. (2016). Evaluation and comparison of Genetic Algorithm and Bees Algorithm for location-allocation of earthquake relief centers. *International Journal of Disaster Risk Reduction*, 15:94–107.

Sagayam, R. and Akilandeswari, M. K. (2012). Comparison of Ant Colony and Bee Colony Optimization for Spam Host Detection. *International Journal of Engineering Research and Development*, 4(8):26–32.

Sagheer, A. M., Sadiq, A. T., and Ibrahim, M. S. (2012). Improvement of scatter search using Bees Algorithm. In *6th International Conference on Signal Processing and Communication Systems*.

Saif, S., Das, P., and Biswas, S. (2021). A hybrid model based on mba-anfis for covid-19 confirmed cases prediction and forecast. *Journal of The Institution of Engineers (India): Series B*, pages 1–14.

Salamat, A. R. and Ghanbarzadeh, A. (2012). Multi Objective Optimization of Antisymmetric Angle-Ply Laminate under Transverse Loads with Bees Algorithm. *American Journal of Advanced Scientific Research*, 1(3):93–98.

Samadzadegan, F. and Ferdosi, E. (2012). Classification of Polarimetric SAR Images Based on Optimum SVMs Classifier Using Bees Algorithm. In *Proceedings of International Conference on intelligent Computational Systems (ICICS'2012)*, pages 106–111.

Samadzadegan, F. and Hasani, H. (2015). Determination of optimum classification system for hyperspectral imagery and LiDAR data based on Bees Algorithm. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences; International Conference on Sensors & Models in Remote Sensing & Photogrammetry,*, 40(1/W5):651–656.

Sarailoo, M., Rahmani, Z., and Rezaie, B. (2015). A novel model predictive control scheme based on bees algorithm in a class of nonlinear systems: Application to a three tank system. *Neurocomputing*, 152:294–304.

Satheesh, A. and Manigandan, T. (2013). Maintaining power system stability with FACTS controller using bees algorithm and NN. *Journal of Theoretical and Applied Information Technology*, 49(1):38–47.

Sayadi, F., Ismail, M., Misran, N., and Jumari, K. (2009). Multi-Objective optimization using the Bees algorithm in time-varying channel for MIMO MC-CDMA systems. *European Journal of Scientific Research*, 33(3):411–428.

Sayarshad, H. R. (2010). Using bees algorithm for material handling equipment planning in manufacturing systems. *International Journal of Advanced Manufacturing Technology*, 48(9-12):1009–1018.

Schmickl, T., Blaschon, B., Gurmann, B., and Crailsheim, K. (2003). Collective and individual nursing investment in the queen and in young and old honeybee larvae during foraging and non-foraging periods. *Insectes Sociaux*, 50(2):174–184.

Scionti, A., D'Amico, C., Ciccia, S., Li, Y., and Terzo, O. (2019). Smart scheduling strategy for lightweight virtualized resources towards green computing. In *Conference on Complex, Intelligent, and Software Intensive Systems*, pages 305–316. Springer.

References

Seeley, T. D. (1986). Social foraging by honeybees: how colonies allocate foragers among patches of flowers. *Behavioral Ecology and Sociobiology*, 19(5):343–354.

Seeley, T. D., Kühnholz, S., and Weidenmüller, A. (1996). The honey bee's tremble dance stimulates additional bees to function as nectar receivers. *Behavioral ecology and sociobiology*, 39(6):419–427.

Seeley, T. D., Visscher, P. K., and Passino, K. M. (2006). Group decision making in honey bee swarms: When 10,000 bees go house hunting, how do they cooperatively choose their new nesting site? *American scientist*, 94(3):220–229.

Sen, M. A. and Kalyoncu, M. (2015). Optimisation of a PID Controller for an Inverted Pendulum Using the Bees Algorithm. *Applied Mechanics and Materials*, 789-790:1039–1044.

Sen, M. A. and Kalyoncu, M. (2016). Optimal Tuning of a LQR Controller for an Inverted Pendulum Using the Bees Algorithm. *Journal of Automation and Control Engineering*, December(1):384–387.

Şenyiğgit, E., Düğenci, M., Aydin, M. E., and Zeydan, M. (2013). Heuristic-based neural networks for stochastic dynamic lot sizing problem. *Applied Soft Computing Journal*, 13(3):1332–1339.

Shafia, M. A., Moghaddam, M. R., and Tavakolian, R. (2011). A Hybrid Algorithm for Data Clustering Using Honey Bee Algorithm, Genetic Algorithm and K-Means Method. *Journal of Advanced Computer Science and Technology Research*, 1:110–125.

Sharma, M. K., Leeprechanon, N., and Phonrattanasak, P. (2017). Hybrid ant colony optimization and bees algorithm for planning of public fast charging stations on a residential power distribution system. *Science & Technology Asia*, pages 112–125.

Sharma, M. K., Phonrattanasak, P., and Leeprechanon, N. (2015). Improved bees algorithm for dynamic economic dispatch considering prohibited operating zones. In *Proceedings of the 2015 IEEE Innovative Smart Grid Technologies - Asia, ISGT ASIA 2015*, number 1 in 1.

Shatnawi, N., Faidzul, M., and Sahran, S. (2013a). Optimization of multilevel image thresholding using the bees algorithm. *Journal of Applied Sciences*, 13(3):458–464.

Shatnawi, N., Sahran, S., and Faidzul, M. (2013b). A memory-based bees algorithm: An enhancement. *Journal of Applied Sciences*, 13(3):497–502.

Shatnawi, N., Sahran, S., and Faidzul, M. (2013c). Bees algorithm using lévy-flights for start configuration. In *2nd International Conference on Computer Science and Computational Mathematics (ICCSCM 2013)*, pages 12–16.

Shatnawi, N. M. (2017). Data clustering using lévy flight and local memory bees algorithm. *International Journal of Business Intelligence and Data Mining*, 12(1):14–24.

Shatnawi, N. M. (2018). Bees algorithm and support vector machine for malaysian license plate recognition. *International Journal of Business Information Systems*, 28(3):284–298.

Sherme, A. E. (2012). A novel method for automatic modulation recognition. *Applied Soft Computing Journal*, 12(1):453–461.

Shirasaki, Y., Shimomura, S., Sugimoto, M., Uwate, Y., and Nishio, Y. (2010). Optimization technique of pheromone-based bees algorithm. In *2010 IEEE Workshop on Nonlinear Circuit Networks*, volume 1.

Shirasaki, Y., Shimomura, S., Sugimoto, M., Uwate, Y., and Nishio, Y. (2011). Effectiveness of guidepost pheromone for honeybee colony optimization. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 1793–1798. IEEE.

Shrme, A. E. (2011). Hybrid intelligent technique for automatic communication signals recognition using Bees Algorithm and MLP neural networks based on the efficient features. *Expert Systems with Applications*, 38(5):6000–6006.

Siavashi, M., Garusi, H., and Derakhshan, S. (2017). Numerical simulation and optimization of steam-assisted gravity drainage with temperature, rate, and well distance control using an efficient hybrid optimization technique. *Numerical Heat Transfer, Part A: Applications*, 72(9):721–744.

Simon, D. (2008). Biogeography-based optimization. *IEEE transactions on evolutionary computation*, 12(6):702–713.

Socha, K. (2009). *Ant colony optimisation for continuous and mixed-variable domains*. Citeseer.

Songmuang, P. and Luantangsrisuk, V. (2016). Comparison of document clustering methods based on bees algorithm and firefly algorithm using thai documents. In *International Symposium on Natural Language Processing*, pages 88–96. Springer.

Songmuang, P. and Ueno, M. (2011). Bees algorithm for construction of multiple test forms in E-testing. *IEEE Transactions on Learning Technologies*, 4(3):209–221.

Songmuang, P., Ueno, M., and Nagaoka, K. (2012). Development of automated e-testing construction system with redundant item detection. In *The 8th International Conference on eLearning for Knowledge-Based Society, Keynote Address*, volume 104.

References

Srikun, I. and Sawetsakulanond, B. (2017). A solution for stochastic optimal power flow with integrated wind power generation using a modified cultural-based bee algorithm. *19th International Conference on Electrical Machines and Systems, ICEMS 2016*, 2(1).

Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.

Stützle, T. (1999). Local search algorithms for combinatorial problems: analysis, improvements, and new applications. *Darmstadt University of Technology PhD Thesis*.

Stutzle, T. and Hoos, H. (1997). Max-min ant system and local search for the traveling salesman problem. In *Proceedings of 1997 IEEE international conference on evolutionary computation (ICEC'97)*, pages 309–314. IEEE.

Stützle, T. and Hoos, H. H. (2000). Max–min ant system. *Future generation computer systems*, 16(8):889–914.

Taher, M. T. and Masoudrahmani, A. (2012). Multicast Routing in Computer Networks considering Quality of Service (QoS) based on Honey Bee Algorithm. *International Journal of Computer Applications*, 58(2):20–24.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons.

Tandis, E. and Assareh, E. (2017). Inverse design of airfoils via an intelligent hybrid optimization technique. *Engineering with Computers*, 33(3):361–374.

Tapkan, P., Ozbakir, L., and Baykasoglu, A. (2012a). Modeling and solving constrained two-sided assembly line balancing problem via bee algorithms. *Applied Soft Computing Journal*, 12(11):3343–3355.

Tapkan, P., OZbakiR, L., and BaykasogLu, A. (2013a). Solving fuzzy multiple objective generalized assignment problems directly via bees algorithm and fuzzy ranking. *Expert systems with applications*, 40(3):892–898.

Tapkan, P., Özbakır, L., and Baykasoğlu, A. (2016a). Bee algorithms for parallel two-sided assembly line balancing problem with walking times. *Applied Soft Computing*, 39:275–291.

Tapkan, P., Ozbakir, L., and Baykasoğlu, A. (2012b). Bees Algorithm for constrained fuzzy multi-objective two-sided assembly line balancing problem. *Optimization Letters*, 6(6):1039–1049.

Tapkan, P., Özbakir, L., and Baykasoğlu, A. (2013b). Solving fuzzy multiple objective generalized assignment problems directly via bees algorithm and fuzzy ranking. *Expert Systems with Applications*, 40(3):892–898.

Tapkan, P., Özbakır, L., Kulluk, S., and Baykasoğlu, A. (2016b). A cost-sensitive classification algorithm: Bee-miner. *Knowledge-Based Systems*, 95:99–113.

Teimoury, E. and Haddad, H. (2013a). A bee algorithm for parallel batch production scheduling. *International Journal of Economics, Finance and Management*, 2(2):169–171.

Teimoury, E. and Haddad, H. (2013b). Solving a parallel batch production scheduling problem using a Bee Algorithm. *Applied mathematics in Engineering, Management and Technology*, 1(1):62–65.

Teodorovic, D. and Dell'Orco, M. (2005). Bee colony optimization–a cooperative learning approach to complex transportation problems. *Advanced OR and AI methods in transportation*, 51:60.

Thomson, J. D. and Goodell, K. (2001). Pollen removal and deposition by honeybee and bumblebee visitors to apple and almond flowers. *Journal of Applied ecology*, 38(5):1032–1044.

Thomson, J. D., Slatkin, M., and Thomson, B. A. (1997). Trapline foraging by bumble bees: Ii. definition and detection from sequence data. *Behavioral Ecology*, 8(2):199–210.

Tiacharoen, S. and Chatchanayuenyong, T. (2019). Optimal fuzzy sliding mode controller design using bee algorithm for dynamic voltage restorer system. *ECTI Transactions on Electrical Engineering, Electronics, and Communications*, 17(1):69–77.

Tiacharoen, S., Potiya, S., and Polratanasuk, P. (2010). Solving various types of Economic Dispatch problem using bees algorithm. In *ECTI-CON 2010 - The 2010 ECTI International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, pages 617–620. IEEE.

Tian, S., Liu, Q., Xu, W., and Yan, J. (2013). A discrete hybrid bees algorithm for service aggregation optimal selection in cloud manufacturing. In *International conference on intelligent data engineering and automated learning*, pages 110–117. Springer.

Tien Bui, D., Khosravi, K., Li, S., Shahabi, H., Panahi, M., Singh, V. P., Chapi, K., Shirzadi, A., Panahi, S., Chen, W., *et al.* (2018). New hybrids of anfis with several optimization algorithms for flood susceptibility modeling. *Water*, 10(9):1210.

Tkach, I. and Amador, S. (2021). Towards addressing dynamic multi-agent task allocation in law enforcement. *Autonomous Agents and Multi-Agent Systems*, 35(1):1–18.

Tkach, I., Edan, Y., Jevtic, A., and Nof, S. Y. (2013). Automatic multi-sensor task allocation using modified distributed bees algorithm. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1401–1406. IEEE.

Tkach, I., Jevtic, A., Nof, S. Y., and Edan, Y. (2018). A modified distributed bees algorithm for multi-sensor task allocation. *Sensors*, 18(3):759.

References

Tolabi, H. B., Ali, M. H., Shahrin Bin Md Ayob, and Rizwan, M. (2014a). Novel hybrid fuzzy-Bees algorithm for optimal feeder multi-objective reconfiguration by considering multiple-distributed generation. *Energy*, 71:507–515.

Tolabi, H. B., Ayob, S. B., Moradi, M. H., and Shakarmi, M. (2014b). New technique for estimating the monthly average daily global solar radiation using bees algorithm and empirical equations. *Environmental Progress & Sustainable Energy*, 33(3):1042–1050.

Tolabi, H. B., Moradi, M. H., and Tolabi, F. B. (2013). New technique for global solar radiation forecast using bees algorithm. *International Journal of Engineering, Transactions B: Applications*, 26(11):1385–1392.

Toloei, A. R., Zarchi, M., and Attaran, B. (2014). Application of Active Suspension System to Reduce Aircraft Vibration using PID Technique and Bees Algorithm. *International Journal of Computer Applications*, 98(6):17–24.

Triwate, P. and Luangpaiboon, P. (2010). Bees algorithm for dynamic multi-zone dispatching in truck load trucking. In *IEEM2010 - IEEE International Conference on Industrial Engineering and Engineering Management*, pages 1165–1169.

Trongwanichnam, K., Thitapars, S., and Leeprechanon, N. (2019). Impact of plug-in electric vehicles load planning to load factor and total generation cost in a power system. In *2019 IEEE PES GTD Grand International Conference and Exposition Asia (GTD Asia)*, pages 599–604. IEEE.

Tsai, H.-C. (2014a). Integrating the artificial bee colony and bees algorithm to face constrained optimization problems. *Information Sciences*, 258:80–93.

Tsai, H.-C. (2014b). Novel bees algorithm: stochastic self-adaptive neighborhood. *Applied Mathematics and Computation*, 247:1161–1172.

Tudu, B., Majumder, S., Mandal, K., and Chakraborty, N. (2011). Optimal unit sizing of stand-alone renewable hybrid energy system using bees algorithm. In *2011 International Conference on Energy, Automation and Signal*.

Tudu, B., Mandal, K. K., and Chakraborty, N. (2014). Optimal design and performance evaluation of a grid independent hybrid micro hydro-solar-wind-fuel cell energy system using meta-heuristic techniques. *1st International Conference on Non Conventional Energy, ICONCE 2014*, pages 89–93.

Uysal, F., Kilinc, E., Kurt, H., Celik, E., Dugenci, M., and Sagiroglu, S. (2017). Estimating seebeck coefficient of ap-type high temperature thermoelectric material using bee algorithm multi-layer perception. *Journal of Electronic Materials*, 46(8):4931–4938.

Vejdannik, M. and Sadr, A. (2017). Automatic microstructural characterization and classification using dual tree complex wavelet-based features and Bees Algorithm. *Neural Computing and Applications*, 28(7):1877–1889.

Vejdannik, M. and Sadr, A. (2018). Automatic microstructural characterization and classification using probabilistic neural network on ultrasound signals. *Journal of Intelligent Manufacturing*, 29(8):1923–1940.

Von Frisch, K. (1974). Decoding the language of the bee. *Science*, 185(4152):663–668.

von Frisch, K. and Lindauer, M. (1956). The" language" and orientation of the honey bee. *Annual review of entomology*, 1(1):45–58.

Voß, S., Martello, S., Osman, I. H., and Roucairol, C. (2012). *Meta-heuristics: Advances and trends in local search paradigms for optimization*. Springer Science & Business Media.

Wang, X., Ceberio, M., Virani, S., Garcia, A., and Cummins, J. (2013). A hybrid algorithm to extract fuzzy measures for software quality assessment. *Journal of Uncertain Systems*, 7(3):219–237.

Wang, X., Contreras, A. F. G., Ceberio, M., Del Hoyo, C., Gutierrez, L. C., and Virane, S. (2012). Interval-based algorithms to extract fuzzy measures for software quality assessment. In *2012 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS)*, pages 1–6. IEEE.

Wang, X., Cummins, J., and Ceberio, M. (2011). The Bees algorithm to extract fuzzy measures for sample data. In *2011 Annual Meeting of the North American Fuzzy Information Processing Society. Institute of Electrical and Electronics Engineers.*

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82.

Wong, L.-P., Low, M. Y. H., and Chong, C. S. (2010). Bee colony optimization with local search for traveling salesman problem. *International Journal on Artificial Intelligence Tools*, 19(03):305–334.

Wong, P. and Chua, A. (2019). Control chart pattern identification using a synergy between neural networks and bees algorithm. *Annals of Electrical and Electronic Engineering*, 2(4):8–13.

Wongthatsanekorn, W. and Matheekrieangkrai, N. (2014). A Case Study of Bee Algorithm for Ready Mixed Concrete Problem. *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, 8(7):1253–1258.

Woodgate, J. L., Makinson, J. C., Lim, K. S., Reynolds, A. M., and Chittka, L. (2017). Continuous radar tracking illustrates the development of multi-destination routes of bumblebees. *Scientific reports*, 7(1):17323.

References

Xie, Y., Murphey, Y. L., and Kochhar, D. S. (2018). Svm parameter optimization using swarm intelligence for learning from big data. In *International Conference on Computational Collective Intelligence*, pages 469–478. Springer.

Xie, Y., Zhou, Z., Pham, D. T., Liu, Q., Xu, W., Ji, C., Lou, P., and Tian, S. (2015a). A forager adjustment strategy used by the bees algorithm for solving optimization problems in cloud manufacturing. In *ASME 2015 International Manufacturing Science and Engineering Conference, MSEC 2015*, volume 2, pages 1–7.

Xie, Y., Zhou, Z., Pham, D. T., Xu, W., and Ji, C. (2015b). A multiuser manufacturing resource service composition method based on the bees algorithm. *Computational Intelligence and Neuroscience*, 2015.

Xu, S., Ji, Z., Pham, D. T., and Yu, F. (2010a). Bio-inspired binary bees algorithm for a two-level distribution optimisation problem. *Journal of Bionic Engineering*, 7(2):161–167.

Xu, S., Ji, Z., Pham, D. T., and Yu, F. (2010b). Bio-inspired binary bees algorithm for a two-level distribution optimisation problem. *Journal of Bionic Engineering*, 7(2):161–167.

Xu, S., Ji, Z., Pham, D. T., and Yu, F. (2011a). Binary bees algorithm–bioinspiration from the foraging mechanism of honeybees to optimize a multiobjective multidimensional assignment problem. *Engineering Optimization*, 43(11):1141–1159.

Xu, S., Yu, F., Luo, Z., Ji, Z., Pham, D. T., and Qiu, R. (2011b). Adaptive bees algorithm—bioinspiration from honeybee foraging to optimize fuel economy of a semi-track air-cushion vehicle. *The Computer Journal*, 54(9):1416–1426.

Xu, W., Tang, Q., Liu, J., Liu, Z., Zhou, Z., and Pham, D. T. (2020). Disassembly sequence planning using discrete bees algorithm for human-robot collaboration in remanufacturing. *Robotics and Computer-Integrated Manufacturing*, 62:101860.

Xu, W., Tian, S., Liu, Q., Xie, Y., Zhou, Z., and Pham, D. T. (2016a). An improved discrete bees algorithm for correlation-aware service aggregation optimization in cloud manufacturing. *The International Journal of Advanced Manufacturing Technology*, 84(1-4):17–28.

Xu, W., Zhong, X., Zhao, Y., Zhou, Z., Zhang, L., and Pham, D. T. (2016b). Manufacturing service reconfiguration optimization using hybrid bees algorithm in cloud manufacturing. In *Monterey Workshop*, pages 87–98. Springer.

Xu, W., Zhou, Z., Pham, D. T., Liu, Q., Ji, C., and Meng, W. (2012). Quality of service in manufacturing networks: A service framework and its implementation. *International Journal of Advanced Manufacturing Technology*, 63(9-12):1227–1237.

Yaghoubi, S. and Fereshteh-Saniee, F. (2020). Optimization of the geometrical parameters for elevated temperature hydro-mechanical deep drawing process of 2024 aluminum alloy. *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering*, page 0954408920949364.

Yang, F., Zheng, J., Tan, H., and Fan, Y. (2015). A specific combination scheme for communication modulation recognition based on the bees algorithm and neural network. *Journal of Communications*, 10(10):797–803.

Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. In *International symposium on stochastic algorithms*, pages 169–178. Springer.

Yang, X.-S. (2020). Nature-inspired optimization algorithms: challenges and open problems. *Journal of Computational Science*, page 101104.

Yang, X.-S. and Deb, S. (2010). Engineering optimisation by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation*, 1(4):330–343.

Yang, X.-S., Deb, S., and Fong, S. (2014). Metaheuristic algorithms: optimal balance of intensification and diversification. *Applied Mathematics & Information Sciences*, 8(3):977.

Yazdi, E., Azizi, V., and Haghighat, A. T. (2010). Evolution of Biped Locomotion Using Bees Algorithm, Based On Truncated Fourier Series. In *Proceedings of the world congress on engineering and computer science*.

Yazdi, E., Azizi, V., and Haghighat, A. T. (2011). Biped locomotion with arm swing, based on truncated fourier series and bees algorithm optimizer. In *Intelligent Automation and Systems Engineering*, pages 15–26. Springer.

Yin, L. H., Choon, Y. W., Chai, L. E., Chong, C. K., Deris, S., Illias, R. M., and Mohamad, M. S. (2013). Prediction of Vanillin Production in Yeast Using a Hybrid of Continuous Bees Algorithm and Flux Balance Analysis (CBAFBA). In *Advances in Biomedical infrastructure 2013, Proceedings of International Symposium on Biomedical Data Infrastructure (BDI 2013)*, pages 101–116. Springer Berlin Heidelberg.

Yin, L. H., Choon, Y. W., Mohamad, M. S., En Chai, L., Chong, C. K., Abdullah, A., Deris, S., and Illias, R. M. (2014). Prediction of Vanillin and Glutamate Productions in Yeast Using a Hybrid of Continuous Bees Algorithm and Flux Balance Analysis (CBAFBA). *Current Bioinformatics*, 9(3):284–294.

Yonezawa, Y. and Kikuchi, T. (1996). Ecological algorithm for optimal ordering used by collective honey bee behavior. In *MHS'96 Proceedings of the Seventh International Symposium on Micro Machine and Human Science*, pages 249–256. IEEE.

Yuan, H., Zhou, M., Liu, Q., and Abusorrah, A. (2020). Fine-grained resource provisioning and task scheduling for heterogeneous applications in distributed green clouds. *IEEE/CAA Journal of Automatica Sinica*, 7(5):1380–1393.

References

Yuce, B., Fruggiero, F., Packianather, M. S., Pham, D. T., Mastrocinque, E., Lambiase, A., and Fera, M. (2017). Hybrid genetic bees algorithm applied to single machine scheduling with earliness and tardiness penalties. In *Computers & Industrial Engineering*, volume 113, pages 842–858. Elsevier.

Yuce, B., Mastrocinque, E., Lambiase, A., Packianather, M. S., and Pham, D. T. (2014). A multi-objective supply chain optimisation using enhanced Bees Algorithm with adaptive neighbourhood search and site abandonment strategy. *Swarm and Evolutionary Computation*, 18:71–82.

Yuce, B., Packianather, M. S., Mastrocinque, E., Pham, D. T., and Lambiase, A. (2013). Honey bees inspired optimization method: The bees algorithm. *Insects*, 4(4):646–662.

Yuce, B., Pham, D. T., Packianather, M. S., and Mastrocinque, E. (2015). An enhancement to the Bees Algorithm with slope angle computation and Hill Climbing Algorithm and its applications on scheduling and continuous-type optimisation problem. *Production and Manufacturing Research*, 3(1):3–19.

Zabil, M. H. M. and Zamli, K. Z. (2013). Implementing a t-way test generation strategy using bees algorithm. *International Journal of Advances in Soft Computing and its Applications*, 5(3):116–126.

Zabil, M. H. M., Zamli, K. Z., and Othman, R. R. (2012). Sequence-based interaction testing implementation using Bees Algorithm. In *2012 IEEE Symposium on Computers and Informatics, ISCI 2012*, pages 81–85.

Zabil, M. M., Zamli, K., and Lim, K. (2018). Evaluating bees algorithm for sequence-based t-way testing test data generation. *Indian Journal of Science and Technology*, 11:4.

Zaeri, R., Ghanbarzadeh, A., Attaran, B., and Zaeri, Z. (2011). Fuzzy Logic Controller based pitch control of aircraft tuned with bees algorithm. In *The 2nd International Conference on Control, Instrumentation and Automation*, pages 705–710.

Zahedi, R. K., Alimouri, P., Nguyen-Xuan, H., and Rabczuk, T. (2017). Crack detection in a beam on elastic foundation using differential quadrature method and the bees algorithm optimization. In *International Conference on Advances in Computational Mechanics*, pages 439–460. Springer.

Zarchi, M. and Attaran, B. (2017). Performance improvement of an active vibration absorber subsystem for an aircraft model using a bees algorithm based on multi-objective intelligent optimization. *Engineering Optimization*, 49(11):1905–1921.

Zarchi, M. and Attaran, B. (2019). Improved design of an active landing gear for a passenger aircraft using multi-objective optimization technique. *Structural and Multidisciplinary Optimization*, 59(5):1813–1833.

Zarea, H., Kashkooli, F. M., Soltani, M., and Rezaeian, M. (2018). A novel single and multi-objective optimization approach based on bees algorithm hybrid with particle swarm optimization (bahpso): Application to thermal-economic design of plate fin heat exchangers. *International Journal of Thermal Sciences*, 129:552–564.

Zarea, H., Moradi Kashkooli, F., Mansuri Mehryan, A., Saffarian, M. R., and Namvar Beherghani, E. (2014). Optimal design of plate-fin heat exchangers by a Bees Algorithm. *Applied Thermal Engineering*, 69(1-2):267–277.

Zarei, K., Atabati, M., and Ahmadi, M. (2017). Shuffling cross–validation–bee algorithm as a new descriptor selection method for retention studies of pesticides in biopartitioning micellar chromatography. *Journal of Environmental Science and Health, Part B*, 52(5):346–352.

Zarei, K., Atabati, M., and Kor, K. (2014). Bee algorithm and adaptive neuro-fuzzy inference system as tools for QSAR study toxicity of substituted benzenes to Tetrahymena pyriformis. *Bulletin of Environmental Contamination and Toxicology*, 92(6):642–649.

Zarei, K., Atabati, M., and Moghaddary, S. (2013). Predicting the heats of combustion of polynitro arene, polynitro heteroarene, acyclic and cyclic nitramine, nitrate ester and nitroaliphatic compounds using bee algorithm and adaptive neuro-fuzzy inference system. *Chemometrics and Intelligent Laboratory Systems*, 128:37–48.

Zargartalebi, H., Attaran, B., Noghrehabadi, A., and Ghanbarzadeh, A. (2012). Simulating Flow in Partly Porous Region Using RBF Neural Network and the Bees Algorithm. In *20th Annual International Iranian Mechanical Engineering Conference*, pages 16–19.

Zeybek, S., Ismail, A. H., Hartono, N., Caterino, M., and Jiang, K. (2021). An improved vantage point bees algorithm to solve combinatorial optimization problems from tsplib. In *Macromolecular Symposia*, volume 396, page 2000299. Wiley Online Library.

Zeybek, S. and Koç, E. (2015). The Vantage Point Bees Algorithm. In *2015 7th International Joint Conference on Computational Intelligence (IJCCI)*, pages 340–345. IEEE.

Zhang, X. and Cheng, X. (2016). Parameter tuning for bees algorithm on continuous optimization problems. In *International Conference in Communications, Signal Processing, and Systems*, pages 409–417. Springer.

Zhou, Z. D., Xie, Y. Q., Pham, D. T., Kamsani, S., and Castellani, M. (2016). Bees Algorithm for multimodal function optimisation. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 230(5):867–884.

Ziarati, K., Akbari, R., and Zeighami, V. (2011). On the performance of bee algorithms for resource-constrained project scheduling problem. *Applied Soft Computing*, 11(4):3720–3733.