University of Luxembourg

DISSERTATION

Graph-Based Algorithms for Smart Mobility Planning and Large-Scale Network Discovery

Presented on 3rd December, 2019 in Luxembourg to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG EN INFORMATIQUE

by

Boonyarit CHANGAIVAL

Dissertation Defence Committee

Prof. Ulrich Sorger , Chairman
University of Luxembourg, Luxembourg

Assoc. Prof. Kittichai Lavangnananda, Vice-chairman
King Mongkut's University of Technology Thonburi, Thailand

Prof. Pascal Bouvry, Supervisor
University of Luxembourg, Luxembourg

Prof. Frédéric Guinand, Member
University of Le Havre Normandy, France

Dr. Grégoire Danoy, Member
University of Luxembourg, Luxembourg

Abstract

Graph theory has become a hot topic in the past two decades as evidenced by the increasing number of citations in research. Its applications are found in many fields, e.g. database, clustering, routing, etc. In this thesis, two novel graph-based algorithms are presented. The first algorithm finds itself in the thriving carsharing service, while the second algorithm is about large graph discovery to unearth the unknown graph before any analyses can be performed.

In the first scenario, the automatisation of the fleet planning process in carsharing is proposed. The proposed work enhances the accuracy of the planning to the next level by taking an advantage of the open data movement such as street networks, building footprints, and demographic data. By using the street network (based on graph), it solves the questionable aspect in many previous works, feasibility as they tended to use rasterisation to simplify the map, but that comes with the price of accuracy and feasibility. A benchmark suite for further research in this problem is also provided. Along with it, two optimisation models with different sets of objectives and contexts are proposed. Through a series of experiment, a novel hybrid metaheuristic algorithm is proposed. The algorithm is called NGAP, which is based on Reference Point based Non-dominated Sorting genetic Algorithm (NSGA-III) and Pareto Local Search (PLS) and a novel problem specific local search operator designed for the fleet placement problem in carsharing called Extensible Neighbourhood Search (ENS). The designed local search operator exploits the graph structure of the street network and utilises the local knowledge to improve the exploration capability. The results show that the proposed hybrid algorithm outperforms the original NSGA-III in convergence under the same execution time.

The work in smart mobility is done on city scale graphs which are considered to be medium size. However, the scale of the graphs in other fields in the real-world can be much larger than that which is why the large graph discovery algorithm is proposed as the second algorithm. To elaborate on the definition of large, some examples are required. The internet graph has over 30 billion nodes. Another one is a human brain network contains around 1011 nodes. Apart of the size, there is another aspect in real-world graph and that is the unknown. With the dynamic nature of the real-world graphs, it is almost impossible to have a complete knowledge of the graph to perform an analysis that is why graph traversal is crucial as the preparation process. I propose a novel memoryless chaos-based graph traversal algorithm called Chaotic Traversal (CHAT). CHAT is the first graph traversal algorithm that utilises the chaotic attractor directly. An experiment with two well-known chaotic attractors, Lozi map and Rössler system is conducted. The proposed algorithm is compared against the memoryless state-of-the-art algorithm, Random Walk. The results demonstrate the superior performance in coverage rate over Random Walk on five tested topologies; ring, small world, random, grid and power-law.

In summary, the contribution of this research is twofold. Firstly, it contributes to the research society by introducing new study problems and novel approaches to propel the advance of the current state-of-the-art. And Secondly, it demonstrates a strong case for the conversion of research to the industrial sector to solve a real-world problem.

Acknowledgements

I remember the time when the Ph.D. study came to my mind. It was when I was a summer student at CERN where I had seen many inspirations from other students and scientists there. And here I am at the end of the Ph.D. study and ready to start a new journey. This Ph.D. study would not have been possible if I have not met Assoc. Prof. Tiranee Achalakul who introduced me to Assoc. Prof. Kittichai Lavangnananda who then introduced me to Prof. Pascal Bouvry. I will always feel grateful for your insights, perspectives and guidances. They are invaluable to me. I will make it my life mission to be able to give the others this kind of opportunities.

I would like to first thank everyone in PCOG team for supporting me during this time, especially Dr. Grégoire Danoy and Dr. Martin Rosalie for his patience listening to my ideas and reports and Abdallah Ibrahim, my officemate who listened to my problems and rants. I would not have made it through the whole study without the support of these amazing people in the wonderful PCOG team. Many thanks also go to Prof. Pierre Kelsen and Dr. Jedrzej Musial for their help in NP-hard proof in the thesis.

Now let's move the location to my home country, Thailand. To all my friends over there, you are sources of laughters and good memories. Even though, for the past years, we did not spend much time together (once a year), I cherish all those times. They lifted me up when I was down. Your encouraging words and supports mean so much to me. I am grateful for thousands of exchanged messages in our group chats and gaming sessions.

The acknowledgement section would not be complete without the mentioning of my family. Thanks mom and dad who always be so understanding and supportive in everything I do. I appreciate all the things you have done for me. I would not be where I am right now without your caring. Thank you for being my emotional anchors. I always think of you two whenever I thought of giving up. I would like to thank my siblings as well since with them near mom and dad, I do not have to worry that much. Your messages in our family group chat are precious to me as they push me to do my best and assure me that you all have my back.

Lastly, I have myself to be thankful of. On this journey where I was the sole traveller, if it was not for my grit and perseverance, I would not have reached the goal. So, I thank myself for being so persevered and having grit to go through all those difficult moments. I want to say to myself "Good job" and I definitely deserve a pat on a shoulder from myself. And to all the people in my life, although you are not mentioned due to the limiting space, I am forever grateful for you all. Thank you.

Contents

Abstract ii

Acknowledgements iii

Contents iv

List of Figures viii

List of Tables xi

Abbreviations xiv

1 Introduction 1
1.1 Research Motivation . . . 1
1.1.1 Graphs and Carsharing . . . 2
1.1.2 Graph Discovery and Chaos Theory . . . 2
1.2 Objectives . . . 2
1.3 Thesis Contributions . . . 3
Smart Mobility . . . 3
Large Graph Discovery . . . 4
1.4 Thesis Organisation . . . 4

2 Smart Mobility Planning 5
2.1 Location Problems and Carsharing . . . 6
2.1.1 Location Problems . . . 6
2.1.2 Shared Fleet Placement . . . 7
2.2 State-of-the-Art Algorithms . . . 8
2.2.1 Exact Methods . . . 8
2.2.2 Heuristic Algorithms . . . 9
2.2.3 Metaheuristic Algorithms . . . 10
Strength Pareto Evolutionary Algorithm-II (SPEA-II) . . . 11
Non-dominated Sorting Genetic Algorithm II (NSGA-II) . . . 12
Reference-Point-Based Non-dominated Sorting Genetic Algorithm (NSGA-III) . . . 13
2.3 Multiobjective Performance Indicators . . . 13

3 Carsharing Fleet Placement Optimisation and Metaheuristic Algorithms 16
3.1 Graph Definition . . . 16
3.2 Benchmark Suite . . . 17
3.2.1 Synthetic Instances . . . 17
3.2.2 Real City Instances . . . 18
3.3 First Phase: Initial Study and Algorithms Screening . . . 19
3.3.1 Optimisation Model . . . 19
3.3.2 Input Parameters . . . 19

3.3.3 Outputs . . . 20
3.3.3.1 Optimisation Objectives and Description . . . 20
3.3.4 NP-hardness Proof . . . 21
3.3.4.1 Objectives Correlation . . . 22
3.3.5 Methodology . . . 22
3.3.5.1 Script for PolySCIP . . . 23
3.3.5.2 Weighted Sum for Heuristic Algorithms . . . 23
3.3.5.3 Solution Encoding and Operators in NSGA-II . . . 23
Solution Encoding . . . 23
Population Initialisation: . . . 23
Selection . . . 25
Crossover . . . 25
Mutation . . . 25
3.3.6 Experimental Setup . . . 25
3.3.7 Experimental Results . . . 27
3.3.7.1 LU1 Scenario: Validation . . . 27
3.3.7.2 LU2 Scenario: Limitation Analysis . . . 28
3.3.7.3 MU1: City-wide Scenario . . . 29
3.3.8 Discussion . . . 31
3.3.9 First Phase Summary: Algorithms and Optimization Objectives . . . 32
3.4 Second Phase: Refinement and Metaheuristic Algorithms Evaluation . . . 32
3.4.1 Optimization Model . . . 33
3.4.1.1 Parameters . . . 33
3.4.1.2 Outputs . . . 34
3.4.1.3 Optimization Objectives and Description . . . 34
3.4.2 NP-Hardness Proof . . . 35
3.4.2.1 Objective Correlation . . . 36
3.4.3 Methodology . . . 36
Solution Encoding . . . 37
Variable length chromosome . . . 37
Sorted don't care symbol . . . 37
Scrambled don't care symbol . . . 38
3.4.4 Experimental Setup . . . 41
3.4.5 Experimental Results . . . 42
3.4.5.1 Statistical Analysis . . . 42
3.4.5.2 Convergence Analysis . . . 43
3.4.6 Second Phase Summary: Observation on Algorithms . . . 45
3.5 Third Phase: Hybridisation . . . 45
3.5.1 Methodology . . . 45
3.5.1.1 Pareto Local Search (PLS) . . . 46
3.5.1.2 Extensible Neighbourhood Search (ENS) . . . 47
Termination conditions . . . 48
Add . . . 49
Cut . . . 49
Improvement . . . 50
3.5.1.3 Hybrid Algorithm: NGAP . . . 50
3.5.2 Hop Parameter Analysis . . . 50
3.5.2.1 Experimental Setup . . . 51
3.5.2.2 Experimental Results . . . 52
3.5.3 NGAP Performance Analysis . . . 54
3.5.3.1 Experimental Setup . . . 56
3.5.3.2 Experimental Results . . . 57
Statistical Analysis . . . 57
Convergence Analysis . . . 58
3.5.3.3 Results Discussion . . . 59

NSGA-III and NGAPs . . . 60
Spread Similarity . . . 60
Performance of Strategy 1 and Strategy 2 . . . 61
3.5.4 Third Phase Summary: Hybridisation and Local Search . . . 61

4 Chaos Theory and Large Scale Graph Discovery 63
4.1 Graph Discovery . . . 64
4.1.1 Graph Topologies . . . 64
4.1.2 Graph Traversal Algorithms . . . 65
4.2 Chaos Theory . . . 67
4.2.1 Solution of a Dynamical System and its Properties . . . 67
4.2.1.1 Lozi Map . . . 68
4.2.1.2 Rössler System . . . 68
4.2.2 Bifurcation Diagram for Optimisation . . . 70
4.2.2.1 Rössler System Bifurcation Diagram . . . 70
4.2.2.2 Lozi Map Bifurcation Diagram . . . 72
4.3 Chaotic Traversal (CHAT) . . . 75
4.3.1 Vanilla version . . . 76
4.3.2 Circular version . . . 76
4.3.3 Performance Metric . . . 77
4.3.3.1 Coverage Percentage . . . 77
4.3.3.2 Mean Time Coverage . . . 77
4.4 Experimental Results: Rössler CHAT . . . 78
4.4.1 Experimental Setup . . . 78
4.4.2 Results on Small Graph Instances . . . 79
4.4.2.1 Ring Topology . . . 79
4.4.2.2 Small World Topology . . . 80
4.4.2.3 Random Topology . . . 81
4.4.2.4 Grid Topology . . . 81
4.4.2.5 Power-Law Topology . . . 81
4.4.3 Results on Large Graph Instances . . . 82
4.4.4 Study of Chaotic Dynamics Impact . . . 84
4.4.4.1 Effect of Periodic Solutions on CHAT . . . 85
4.4.4.2 Effect of Chaotic Solutions on CHAT . . . 86
4.5 Experimental Results: Lozi CHAT . . . 88
4.5.1 Experimental Setup . . . 88
4.5.2 Results on Small Graphs Instances . . . 89
4.5.2.1 Ring Topology . . . 89
4.5.2.2 Small World Topology . . . 89
4.5.2.3 Random Topology . . . 90
4.5.2.4 Grid Topology . . . 90
4.5.2.5 Power-Law Topology . . . 91
4.5.3 Results on Large Graphs Instances . . . 91
4.5.4 Study of Chaotic Dynamics Impact . . . 92
4.6 Experimental Results: Mean Time Coverage . . . 93
4.6.1 Test Platform . . . 94
4.6.2 Results on Algorithm Computational Performance . . . 94
4.7 Experimental Results: Summary . . . 95

5 Conclusions 96
5.1 Summary . . . 96
5.2 Future Work . . . 98

Appendix A Preliminary Results for State of the Art Algorithms 100
A.1 Statistical Results . . . 100

A.1.1 Results for 5000-node Instances . . . 100
A.1.2 Results for 10000-node Instances . . . 102
A.1.3 Results for 15000-node Instances . . . 103
A.1.4 Results for 20000-node Instances . . . 104
A.1.5 Results for 25000-node Instances . . . 106
A.1.6 Results for 30000-node Instances . . . 107
A.1.7 Results for 35000-node Instances . . . 108
A.1.8 Results for 40000-node Instances . . . 110
A.1.9 Results for 45000-node Instances . . . 111
A.1.10 Results for Berlin and Dusseldorf Instances . . . 112
A.2 Convergence Results for Real World Instances . . . 114
A.2.1 Convergence Plots of Berlin . . . 114
A.2.2 Convergence Plots of Dusseldorf . . . 114

Appendix B Hybrid Algorithm Results **116**
B.1 Statistical Results . . . 116
B.1.1 Statistical Results for 10000-node Instances . . . 116
B.1.2 Statistical Results for 15000-node Instances . . . 117
B.1.3 Statistical Results for 20000-node Instances . . . 119
B.1.4 Statistical Results for 25000-node Instances . . . 120
B.1.5 Statistical Results for 30000-node Instances . . . 121
B.1.6 Statistical Results for 35000-node Instances . . . 123
B.1.7 Statistical Results for 40000-node Instances . . . 124
B.1.8 Statistical Results for 45000-node Instances . . . 125

Publications **126**

References **130**

List of Figures

3.1 The example graph instance for VPP. 17
3.2 Illustration for virtual area of a station on streets where the diamond shape is a center point and squares are buildings. Green coloured squares are covered buildings, yellow coloured one, otherwise. Station radius is denoted by r and maximum walking distance for coverage is denoted by w. 19
3.3 A two-point crossover process. 25
3.4 A uniform mutation process. 25
3.5 Pareto fronts and single objective solutions from all presented algorithms mentioned in Section **??** in the validation phase. The higher the value on x axis (moving toward the right), the better the distance objective is. The higher the value on y axis, the better the user coverage objective is. . . . 27
3.6 A map showing a solution from NSGA-II that yields the highest user coverage and the highest maximum walking distance. Covered buildings are depicted as nodes inside polygons. 28
3.7 Computation time for various instances (from Luxembourg city) of PolySCIP. The following numbers are in "street nodes x buildings" format. (a) 63x47 (b) 222x307 (c) 366x584 (d) 727x876 (e) 1390x1063. For instance (a) and (b), the number of stations is four. Instance (c), (d) and (e), the number of stations is 10. 29
3.8 NSGA-II's Pareto front and heuristic algorithms' solutions for LU2. The higher the value on x axis (moving toward the right), the better the distance objective is. The higher the value on y axis, the better the user coverage objective is. 29
3.9 NSGA-II Pareto front and solutions of heuristic algorithms in MU1 scenario. 30
3.10 NSGA-II fleet placement solution which maximises user coverage in the city of Munich. Red pins are locations of the carsharing station. Green zones indicate the inner area of Munich City . . . 30
3.11 NSGA-II approximated Pareto front and solutions of heuristic algorithms in MU1 scenario compared to the manual allocation (72 stations). 31
3.12 A solution that yields a low global walking distance, but also yielded low user coverage. 32
3.13 Chromosome representation for variable length chromosomes. 38
3.14 Chromosome representation for sorted don't care chromosomes. 38
3.15 Chromosome representation for scramble don't care chromosomes. 38
3.16 A generational plot of IGD for each encoding scheme on City_1 instance. 40
3.17 A generational plot of spread for each encoding scheme on City_1 instance. 40
3.18 A generational plot of hypervolume for each encoding scheme on City_1 instance. 41
3.19 Generational plot of IGD of each algorithm on Munich instance. 44
3.20 Generational plot of spread of each algorithm on Munich instance. 44
3.21 Generational plot of hypervolume of each algorithm on Munich instance. 44
3.22 Neighbourhood in a graph. 47
3.23 Adding process in ENS. 49
3.24 Cutting process in ENS. 50
3.25 Improvement process in ENS. 50
3.26 Surface plot of average IGD around the *addHop* and *cutHop* of (6,4). 53
3.27 Surface plot of average spread around the *addHop* and *cutHop* of (6,4). 54
3.28 Surface plot of Average IGD around the *addHop* and *cutHop* of (6,4). 55
3.29 Generational plot of IGD of each algorithm on Hamburg instance. 58
3.30 Generational plot of spread of each algorithm on Hamburg instance. 59
3.31 Generational plot of hypervolume of each algorithm on Hamburg instance. 60
3.32 Building blocks in a solution. 60

4.1 Lozi attractor with the parameters $a = 1.7$ and $b = 0.5$ solution to Eq. (4.1). 68
4.2 Attractor solution to the Rössler system (Eq. (4.2)) for $\alpha = 0$ (Eq. (4.3)) with Poincaré section. The arrow indicates the orientation of the normalised value ρ_n. 69
4.3 First Return Map to the Poincaré section for $\alpha = -0.25$. 70
4.4 The bifurcation diagram where α is varied in Eq. (4.3) for the Rössler system Eq. (4.2). 71
4.5 Partitions in the bifurcation diagram according to Eq. (4.6) and Eq. (4.7) 72
4.6 A plot of ρ_n against iteration for growing pattern. 73
4.7 A plot of ρ_n against iteration for oscillation pattern. 73
4.8 A plot of ρ_n against iteration for plateau pattern. 73
4.9 Partitions in the bifurcation diagram according to Eq. (4.8) and Eq. (4.9) for the Lozi map with $a = 1.5$ and $b = [-0.5, 0.47]$. 74
4.10 Partitions in the bifurcation diagram according to Eq. (4.10) and Eq. (4.11) for Lozi map with $a = [1.1, 1.8]$ and $b = 0.1$. 74
4.11 Agent comes to the node 1 from node 4. 76
4.12 The difference between Vanilla, Circular First and Circular Last neighbour list in the situation of Fig. 4.11. 77
4.13 Performance of Rössler CHATs in comparison to the Random Walk on the ring topology. The black solid line is the result of the Circular CHAT and the greosslery line is the result of Vanilla CHAT. The dashed line represents the result of the Random Walk. The shaded areas highlight periodic solutions of the Rössler system. 80
4.14 Performance of Rössler CHATs in comparison to the Random Walk on the small world topology (see Fig. 4.13 definitions). 80
4.15 Performance of Rössler CHATs in comparison to the Random Walk on the random topology (see Fig. 4.13 definitions). 81
4.16 Performance of Rössler CHATs in comparison to the Random Walk on the grid topology (see Fig. 4.13 definitions). 82
4.17 Performance of Rössler CHATs in comparison to the Random Walk on power-law topology (see Fig. 4.13 definitions). 82
4.18 First Return Map to the Poincaré section for $\alpha = 0.33$. 84
4.19 First Return Map to the Poincaré section for $\alpha = -0.26$. Periodic points are indicated to underline the structure of the first return map. These periodic points corresponds to orbits in the attractor. Only periodic points associated to orbits with a period lower than five are indicated. 85
4.20 The density of iterates at $\alpha = -0.26$. 85
4.21 Periodic solution and traversal of an agent in a grid from the starting point (S). 86
4.22 First Return Map to the Poincaré section for $\alpha = 1.124$. Periodic points represent periodic orbits in the three-dimensional space. Period 2 orbit is thus represented by the points $A = (x_a, y_a)$ and $B = (y_a, x_a)$ in the first return map. Only periodic points associated to orbits with a period lower than four are indicated. 87
4.23 The density of iterates at $\alpha = 1.124$. 87
4.24 A plot of ρ against iteration. 88
4.25 Performance of Lozi CHAT: both are compared to the Random Walk on the ring topology. The black solid line represents the results of the Circular (Last) CHAT and the grey line the result of Vanilla CHAT. The dashed line represents the result of the Random Walk. The Circular First method is not shown due to its inferior results compared to the Circular Last method. 90
4.26 Performance of Lozi CHAT in comparison to the Random Walk on small world topology (see Fig. 4.25 definitions). 90
4.27 Performance of Lozi CHATs in comparison to the Random Walk on random topology (see Fig. 4.25 definitions). 91
4.28 Performance of Lozi CHATs in comparison to the Random Walk on grid topology (see Fig. 4.25 definitions). 91
4.29 Performance of Lozi CHATs in comparison to the Random Walk on power-law topology (see Fig. 4.25 definitions). 92
4.30 The density of iterates of Lozi map for parameter values $a = 1.5$ and $b = -0.44$. 93
4.31 The density of iterates of Lozi map for parameters values $a = 1.78$ and $b = 0.1$. 93

A.1 Generational plots. (a) A generational plot of IGD (b) A generational plot of spread (c) A generational plot of hypervolume. 114

A.2 Generational plots. (a) A generational plot of IGD (b) A generational plot of spread (c) A generational plot of hypervolume. 115

List of Tables

3.1 Number of street nodes and public transport locations in observed major cities in Europe. 18
3.2 Spearman correlation between two objectives. Objective 1 is user coverage maximisation. Objective 2 is global walking distance minimisation. 22
3.3 Evaluation scenarios. 26
3.4 NSGA-II configuration parameters. Population size is increased as a graph becoming larger. The mutation rates are also modified to accommodate the number of stations in a solution. 27
3.5 Numerical results in LU1 scenario. Only extreme solutions from two Pareto fronts are mentioned. 28
3.6 Comparing Pareto fronts for PolySCIP and NSGA-II. 28
3.7 Execution time for NSGA-II and heuristic algorithms on MU1. The measured time depicts an execution time each algorithm takes to locate 100 stations. 30
3.8 Spearman correlation coefficient between each pair of objectives. Objective 1 is user coverage maximisation. Objective 2 is vehicle number minimisation. Objective 3 is public transport coverage maximisation. 36
3.9 Parameter configuration for each encoding scheme. 39
3.10 IGD for each encoding scheme. 39
3.11 Spread for each encoding scheme. 39
3.12 Hypervolume for each encoding scheme. 40
3.13 Parameters of SPEA-II, NSGA-II and NSGA-III. 42
3.14 Average results obtained for the IGD metric. 42
3.15 Average results obtained for the spread metric. 43
3.16 Average results obtained for the hypervolume metric. 43
3.17 *addHop* and *cutHop* parameters 52
3.18 Average IGD for each set of *addHop* and *cutHop* (part 1) 52
3.19 Average IGD for each set of *addHop* and *cutHop* (part 2) 52
3.20 Average IGD for each set of *addHop* and *cutHop* (part 3) 53
3.21 Average spread for each set of *addHop* and *cutHop* (part 1) 53
3.22 Average spread for each set of *addHop* and *cutHop* (part 2) 54
3.23 Average spread for each set of *addHop* and *cutHop* (part 3) 54
3.24 Average hypervolume for each set of *addHop* and *cutHop* (part 1) 55
3.25 Average hypervolume for each set of *addHop* and *cutHop* (part 2) 55
3.26 Average hypervolume for each set of *addHop* and *cutHop* (part 3) 55
3.27 Termination criteria for each instance. 56
3.28 Parameter configuration for experimental algorithms. 56
3.29 Average results of evaluated algorithm for the IGD metric. 57
3.30 Average results of evaluated algorithm for the spread metric. 58
3.31 Average results of evaluated algorithm for the hypervolume metric. 59

4.1 Experiments formulation. The best values of α obtained from the small graph experiments are used to explore large graphs. Every α is tested for 300 runs. 78
4.2 Coverage Percentage of CHAT and Random Walk. The first number is the Coverage Percentage. The second number behind $\pm$ is the standard deviation. 83
4.3 Wilcoxon test between each algorithm where ▲, ▼ mean that the result is better, respectively worse, significantly The five symbols in a column represent a performance for each topology: Ring, Small World, Random, Grid and Power-Law respectively 83

4.4 Experiments formulation. The best values of a and b obtained from the small graph experiments are used to explore large graphs. Selected combinations of a, b, and method are tested for 30 runs on each graph. 89
4.5 Best performance of Rössler CHAT and Lozi CHAT for each graph topology. The first number is the Coverage Percentage and the number behind ± is the standard deviation. For Rössler and Lozi CHAT configurations, please refer to Tab. 4.1 and Tab. 4.4 respectively. 92
4.6 Mean Time Coverage for each Algorithm in node/second. The CPU time and number of discovered nodes are calculated from 30 runs. The best results are in bold. 94
4.7 CPU time for generating the value sequence for Rössler system ($\alpha = 1.124$), Lozi map ($a = 1.5$, b = -0.44), and Marsenne Twister algorithm (in second). These CPU times are averaged from 30 runs. 94
4.8 Best parameters for Lozi CHAT for each tested graph topology 95

A.1 Average results obtained for the IGD metric. 101
A.2 Average results obtained for the spread metric. 101
A.3 Average results obtained for the hypervolume metric. 101
A.4 Average results obtained for the IGD metric. 102
A.5 Average results obtained for the spread metric. 102
A.6 Average results obtained for the hypervolume metric. 103
A.7 Average results obtained for the IGD metric. 103
A.8 Average results obtained for the spread metric. 104
A.9 Average results obtained for the hypervolume metric. 104
A.10 Average results obtained for the IGD metric. 105
A.11 Average results obtained for the spread metric. 105
A.12 Average results obtained for the hypervolume metric. 105
A.13 Average results obtained for the IGD metric. 106
A.14 Average results obtained for the spread metric. 106
A.15 Average results obtained for the hypervolume metric. 107
A.16 Average results obtained for the IGD metric. 107
A.17 Average results obtained for the spread metric. 108
A.18 Average results obtained for the hypervolume metric. 108
A.19 Average results obtained for the IGD metric. 109
A.20 Average results obtained for the spread metric. 109
A.21 Average results obtained for the hypervolume metric. 109
A.22 Average results obtained for the IGD metric. 110
A.23 Average results obtained for the spread metric. 110
A.24 Average results obtained for the hypervolume metric. 111
A.25 Average results obtained for the IGD metric. 111
A.26 Average results obtained for the spread metric. 112
A.27 Average results obtained for the hypervolume metric. 112
A.28 Average results obtained for the IGD metric. 112
A.29 Average results obtained for the spread metric. 113
A.30 Average results obtained for the hypervolume metric. 113

B.1 Average results obtained for the IGD metric. 116
B.2 Average results obtained for the spread metric. 117
B.3 Average results obtained for the hypervolume metric. 117
B.4 Average results obtained for the IGD metric. 118
B.5 Average results obtained for the spread metric. 118
B.6 Average results obtained for the hypervolume metric. 118
B.7 Average results obtained for the IGD metric. 119
B.8 Average results obtained for the spread metric. 119
B.9 Average results obtained for the hypervolume metric. 120
B.10 Average results obtained for the IGD metric. 120
B.11 Average results obtained for the spread metric. 121
B.12 Average results obtained for the hypervolume metric. 121

B.13 Average results obtained for the IGD metric. 122
B.14 Average results obtained for the spread metric. 122
B.15 Average results obtained for the hypervolume metric. 122
B.16 Average results obtained for the IGD metric. 123
B.17 Average results obtained for the spread metric. 123
B.18 Average results obtained for the hypervolume metric. 124
B.19 Average results obtained for the IGD metric. 124
B.20 Average results obtained for the spread metric. 125
B.21 Average results obtained for the hypervolume metric. 125
B.22 Average results obtained for the IGD metric. 126
B.23 Average results obtained for the spread metric. 126
B.24 Average results obtained for the hypervolume metric. 126

Abbreviations

LSCP	Location Set Covering Problem
MCLP	Maximal Covering Location Problem
FLP	Facility Location Problem
FPP	Fleet Placement Problem
MOEA	Multiobjectives Evolutionary Algorithm
MOGA	Multiobjectives Genetic Algorithm
SPEA-II	Strength ParetoEvolutionary Algorithm II
VEGA	Vector Evaluated Genetic Algorithm
NSGA	Non-dominated Sorting Genetic Algorithm
NSGA-II	Non-dominated Sorting Genetic Algorithm II
NSGA-III	Reference Point based Non-dominated Sorting Genetic Algorithm
IGD	Inverted Generational Distance
UB	Upper Bound
LB	Lower Bound
SEAMO	Simple Evolution Algorithm for Multiobjective Optimisation
PESA	Pareto Envelope-based Selection Algorithm
X3C	Exact Cover By 3-Sets
ENS	Extensible Neighbourhood Search
PLS	Pareto Local Search
TSP	Traveling Salesman Problem
NGAP	Reference Point based Non-dominated Sorting Genetic Algorithm with Pareto Local Search
MAN	Metropolitan Area Network
WWW	World Wide Web
PPI	Protein-Protein Interaction
DFS	Depth First Search
BFS	Breadth First Search
CPU	Central Processing Unit
GPU	Graphic Processing Unit
LRTA*	Learning Real-Time A*
CPRNG	Chaotic Pseudo Random Number Generator

PSO	Particle Swarm Optimization
ODE	Ordinary Differential Equations
RK4	Runge-Kutta method (fourth order)
CHAT	Chaotic Traversal
PRNG	Pseudo Random Number Generator

I dedicate this thesis to my loving family, the Changaivals

Chapter 1

Introduction

It is undeniable that this era is the age of information. The amount of 2.5 quintillion bytes of data are generated each day mostly from social media, communication and open data movement [1]. Following the trend of Big Data, graph representation emerges as one of the main focuses of several data representations. The main strength of graph representation is the ability to record the relationships between data entities which, if utilised efficiently, can yield valuable insights. Applications can be seen in various fields where graphs are used for analysis such as biology (proteins, brain, and disease maps), transportation, telecommunication and social network [2]. With a large amount of these data, it calls for efficient algorithms to process them in an acceptable amount of time. As commonly known, time is the most critical resource known to mankind. Therefore, with this realisation, two novel algorithms are proposed. The first algorithm which is the novel hybrid metaheuristic algorithm with application in carsharing fleet planning to determine the optimal fleet location in a city of operation. It does not require any pre-analysis of the city to pick out potential locations, hence promises fully automated planning process. The second algorithm is a novel graph traversal algorithm that is memoryless to accomodate with the Big Data demand that can go beyond any capability of any resources. It not only find its application in graph discovery, but is also one of pioneering works that utilises chaos theory in a graph traversal domain. By proposing these two novel graph algorithms, the usage of graph research in the real-world scenarios is illustrated and hopefully, this research inspires others to pioneer into various potential directions and real applications.

1.1 Research Motivation

There are numbers of existing work using graphs for data analysis such as recommendation system and social analysis. In fact, most publications in the past two decades contributed to those areas of analysis which also reflects the popularity that graph (or network) theory has gained [3]. There are several advantages in using a graph. Firstly, it preserves the relationship between entities in its natural form which is not offered by other data structures [4]. Second, a graph is a suitable mean of visualisation [5]. Third, graph theory has been studied for several decades, there is a plethora of algorithms that can aid in any tasks, e.g., shortest path calculation and graph generators [6]. Finally, more and more graph data are now available on the internet, e.g., street network and social network data [7, 8]. These advantages lead to a simple, yet vast question, "How can graph be beneficial in other fields of research and practical for real-world applications?". It is already proven that graphs can find their applications in many other fields. Therefore, in this research, I propose two novel algorithms for two different problems (but still, are graph-based) than traditional use cases.

1.1.1 Graphs and Carsharing

Carsharing services are now being offered in several major cities across the continents. Despite their popularity, existing research can only be of use to some extent due to the lack of accurate real-world street network (i.e., rasterisation versus street network). This leads to models that cannot be applied on the problem directly and will require a lot of modifications every time the problem instance is changed. Furthermore, this planning process normally follows the manual procedure starting from analysing the demographic data per district to many field observations, and then initial planning with expert knowledge. These tasks are tedious as described, but can be improved through the help of automation. The benefit is not to only relieve the burdens of staff, so that they can focus more on other tasks, but also speeding up and enhancing the accuracy of the planning process in this competitive era.

Due to the aforementioned, the aim of this research is to facilitate the planning and decision making process via automation. The proposed work takes full advantage of the fact that the street networks, in general, are available online and in a graph representation, e.g. from voluntary platform such as OpenStreetMap. In addition, government are more and more supportive to the open data initiative resulting in a flooding amount of demographic and topological data of many countries available for free. It would be wasteful not to make a good use of these available data which can help enhancing the accuracy and feasibility of the automated planning greatly. In the first part of this thesis, through a series of research, I propose an novel optimisation model that encompass three real objectives from the carsharing company; (1) maximising user (2) maximising public transport coverage and (3) minimising the number of vehicles in the fleet. The optimisation method is aided by the utilisation of a graph to enhance the feasibility and accuracy of the model and solving algorithm.

1.1.2 Graph Discovery and Chaos Theory

Nowadays, data have exceedingly high potential value. This can be realised from the real examples in the world such as Google and Facebook who govern large amount of data each day. Despite that, data in itself, are useless, unless information or insight are extracted out of them through data analysis.

As stated before, graphs are currently one of the most popular data representation as well as the fastest-growing [3]. To put it into perspective, a single human connects to about 130 other individuals [9]. If we scale it to 10,000 people, this could be as many as 1,300,000 individuals and this is just a size of a town population. These graphs can grow infinitely. These graphs are normally marked as unknown as it is difficult to track changes such as edge modification, node metadata, or new nodes. However, in order to perform an analysis on those graphs, the caveat is that some information about them (e.g., size, topology, etc.) must be known. Therefore, there is a need for a graph discovery process before any analysis. In the second half of this research, the progression from the known graph territory (street network) to that of a very large unknown graph is a necessary step and, hence, a novel memoryless graph traversal algorithm based on chaotic systems is proposed.

1.2 Objectives

There are two applications illustrated in this research dealing with different scenarios and sizes of graph as a solid proof how novelty in graph algorithms can be beneficial in solving real-world problem. The first application is in smart mobility where we collaborate with a carsharing company to bring automation to the planning process. The second application moves from a medium-size known graph to a very large unknown graph. It focuses on

large graph discovery to accomodate the Big Data era, and also demonstrates the interdisciplinary research between computer science and applied mathematic. Therefore, the objectives of this research are;

1. To study state-of-the-art algorithms for fleet placement in carsharing business and related businesses (such as bikesharing) and identify the strengths and weaknesses of each techniques. This is an initial state in disseminate each technique and aid in developing a new algorithm to solve the fleet placement problem.
2. To create a realistic benchmark suite consisting of real-city-based synthetic instances and real city instances for performance measurement. This benchmark suite ought to propel the research in this field even further.
3. To model the optimisation problem with the usage of graph, in order to enhance the accuracy and feasibility of the problem which can contribute to the development of the research field.
4. To design and develop a hybrid metaheuristic algorithm that is more efficient in solving the problem than the current state-of-the-art algorithms.
5. To create a strong collaboration with the industries and encourage the utilisation of the research work into the real-world application.
6. To study the nature of the unknown graphs and state-of-the-art techniques for graph traversal and identify their strengths and weaknesses and constraints of their operations.
7. To apply the chaos theory and their essences through two well-known chaotic system, namely, Lozi and Rössler attractors.
8. To design and develop a novel memoryless large unknown graph traversal algorithm which incorporates the chaos theory.

1.3 Thesis Contributions

The contribution of this thesis is twofold. The first contribution is the two novel algorithms that are extended from the state-of-the-art knowledge by combining existing theories. Second, it brings the research to the real-world application through the collaboration with an industrial sector. These points contribute to the advancement in the academia. Since there are two algorithms in this research, we mention the contribution of each algorithm for comprehensiveness.

Smart Mobility

For the smart mobility, the focuses are on using the available open data and automatisation of the planning process.

1. The automatisation to the carsharing field is introduced through the utilisation of the street network (represented as a graph) and demographic data that are freely available on the internet and the proposed optimisation models that promise higher accuracy and feasibility of the yielded solutions.
2. A novel hybrid metaheuristic algorithm based on NSGA-III and Pareto Local Search (PLS) with problem specific local search operator to fully utilise the graph neighbourhood concept in order to improve the performance of the state-of-the-art algorithm is proposed.

3. The extensive study on the classical location problems, e.g., facility location problem (FLP) and maximal covering location problem (MCLP) and their variants, are conducted. The study also provide the explanations of state-of-the-art algorithms and solvers used in solving those problems. The links between these location problems and the carsharing fleet placement problem are also mentioned along with the current trend in the study of sharing economy field i.e. electric shared vehicles and bikesharing.

4. The results from the research are validated by the field experts and aided the company in the planning process of their car fleets for new cities.

Large Graph Discovery

Graph discovery is a crucial step before any analysis. A novel graph traversal method that is still efficient in the face of very large graph is proposed.

1. A novel memoryless graph traversal algorithm based on two chaotic attractors; Lozi and Rössler map, is proposed.

2. The proposed graph traversal algorithm is one of a few pioneering works that introduce chaos theory to the field of graph traversal and the proposed algorithm perform better than the state-of-the-art algorithm, Random Walk.

3. Five graph topologies, e.g. ring, small world, random, grid and power-law, are studied and their applications in the real-world scenarios are given. Furthermore, the current state-of-the-art graph traversal algorithms are reviewed to observe strengths and weaknesses. Lastly, general concept of chaos they and two chaotic systems along with their bifurcation diagram and analysis tools are studied to provide the foundation for the proposed novel large graph traversal algorithm.

1.4 Thesis Organisation

This thesis proposes two novel graph algorithms. For the sake of better comprehension, they are explained orderly based on the scale of the problem, hence the first part of this research provides full details of fleet placement in carsharing. Afterward, a very large unknown graph traversal algorithm based on the chaos theory is explained. The thesis is organised as follows.

In Chapter 2, the related work in classical optimisation problems and the links between them and the fleet placement problem in carsharing are presented. The current trend in carsharing is also mentioned as to concretise the benefits that our approach and graph can bring to the field. Then, Chapter 3 presents the definition of graph and benchmark suite. The three phases of in the experiments from the algorithm selection to the hybrid metaheuristic algorithm proposal are also highlighted. Finally, we conclude Chapter 3 with the findings of the conducted experiments.

Chapter 4 is a transition from medium-sized graph used in the carsharing problem to very large unknown graphs. Therefore, this chapter presents the current state-of-the-art on graph traversal algorithms along with the foundations of chaos theory. Furthermore, the performance indicators for unknown graph traversal are provided along with the conclusion of the very large graphs experiment findings.

Finally, the conclusion which we draw from these two algorithms are presented in Chapter 5. Besides that, I discuss the future directions of the research and fields that can benefit from the usage of graph to encourage more application-based research which both contribute to the industrial sector and academia.

Chapter 2

Smart Mobility Planning

Smart Mobility refers the usage of modes of available transportation which can be public transportation, personal driving service (i.e. taxi) and ride-sharing (i.e. carsharing, bikesharing, and car pooling) in such a way that, according to [10], enables these specific six features in the transportation.

1. Reducing pollution
2. Reducing traffic congestion
3. Reducing noise pollution
4. Improving transfer speed
5. Reducing transfer costs

In this research, carsharing planning, a new mode of transportation in a city, is studied. Carsharing is an important component of the shared mobility ecosystem for big cities already in the now and will be come even bigger in the future. It can reduce the number of household cars down by half by increasing the efficiency and utilization of vehicles [11]. In addition, carsharing helps reducing traffic congestion and pollution in cities [12]. Supported by municipalities and end users, carsharing is currently experiencing an unprecedented growth and is projected to reach 36 million users with a fleet of 430,000 vehicles globally in 2025 [13].

Depending on whether the vehicles need to be returned to their initial location at the end of the trip, carsharing can be categorised as either free float or round trip. Free-float trips can start and end in any part of the city covered by the service. Flee-float mobility is highly flexible, available on demand, and serves all kinds of customer trips (both one-way and round-trip). However, flexibility comes at a high operational cost for the company which needs to maintain a high density of vehicles even in low-demand areas of the city to cope with low levels of utilisation [14]. Also, vehicles that end up in low demand areas need to be relocated. An example of well-known free-floating carsharing companies are SHARE NOW (which is the merge of Car2Go and DriveNow) [15].

While in the round-trip or station-based scenario, shared vehicles must be returned to the area where the vehicle was picked up. In other words, each vehicle not in use resides in a well-designated area, which is typically only few kilometers wide. As a result, round-trip services are less flexible, but they are easier to implement and manage. Station-based services, hence, require fewer cars, as carsharing operators can place vehicles in densely

populated high-demand zones of the city only, and no fleet relocation is typically required. In such a scenario, choosing correct locations for fleet vehicles and their resulting coverage are of the utmost importance.

In the scope of this research, several objectives that come directly from the carsharing expert and their effects on the vehicle locations are studied. These objectives inherently make our work solely focuses on multiobjective optimisation. Hence, this chapter presents the foundation knowledge of this work which consists the previous work in location problem and state-of-the-art multiobjective optimisation algorithms. This chapter is organized as follows. The first part provides the review of the scientific approaches applied on similar generic location problems and then related work on fleet management in carsharing to provide the background of fleet management and optimisation in carsharing. The second part features state-of-the-art algorithms in the previous work. Finally, the third part presents the performance indicators for multiobjective algorithms.

2.1 Location Problems and Carsharing

Proper fleet placement is of the high importance in station-based round-trip carsharing. In this section, scientific approaches relevant to the fleet placement optimisation are reviewed.

2.1.1 Location Problems

There are several similarities of fleet location with classic optimisation problems. Two of the classic problems are selected which are maximal covering location problem and facility location problem since they are highly relevant to the fleet placement problem.

Before describing the maximal covering problem, it is essential to introduce first the location set covering problem (LSCP). In LSCP, the objective is to identify the minimal number and the location of the facilities such that all demand points are ensured to be within the maximal service distance from a facility. If the problem is solved over a range of maximal distance, a cost-effective curve demonstrating the relationship between maximal distance and minimum number of location can be constructed. However, as resources are often of limited, it is not always possible to formulate the LCSP in the real world scenario [16]. Hence, the focus is shifted to cover as many as possible within the desired distance and available resources leading to the introduction of the maximal covering location problem (MCLP).

Church and ReVelle proposed maximal covering location problem (MCLP) in 1974 [17] for facility and emergency siting. MCLP concept also appears in the location set covering problem. The objective is to maximise the partial coverage with a number of facilities, where each facility has a fixed coverage distance. MCLP is shown to be NP-hard, which means it becomes intractable and cannot be solved in an acceptable time by exact methods when the size of an instance is large [18]. MCLP has been applied in many real-world problems. Seargeant used MCLP as a base model to place health care facility location based on the demographic data in the regions [19]. Schmid and Doerner formulate their ambulance siting problem on MCLP with the integration of patients data and traces of taxis in Vienna to estimate the traveling time to reach the patient [20]. Another example in telecommunication is from Ghaffarinasab and Motallebzadehb, the authors proposed a bi-level version of hub interdiction problem (also another variant of MCLP) [21]. MCLP was also extended to be multi-objective. Xiao et al. proposed a MCLP with two objectives which were facility cost and proximity minimisation [22]. Kim and Murray solved another bi-objective version of the MCLP where it aimed to maximise primary and backup coverage (overlapping coverage for reliability) [23]. There is also a work from Malekpoor et al. which

the authors formulated the electrification in disaster relief camp which aim to find a set of locations to reduce the project cost and increase the share of systems between sites [24].

The second problem is the facility location problem (FLP). The objective is to find locations to place all the facilities to supply the stores, while minimising the maximum cost (p-center) or the average cost (p-median) [25]. In this problem, one constraint is to have all the stores covered and one store can be covered by only one facility [16]. Other variants to FLP are GSM antennas location problem and patrolling allocation problem. The GSM antenna location problem is very close to the fleet placement problem. Its goal is to find sites to set up GSM antennas to maximise the number of covered mobile phone users. [26, 27]. In the same sense, the patrolling allocation aims to effectively allocate police patrol cars in the neighbourhoods based on the demands and priority of the situation. However, these problem considers a coarse map (e.g. raster map) and normally does not pinpoint exact locations. Furthermore, both mentioned problems do not consider the minimum distance between the site and nearby users as it does not affect the quality of service. One of many interesting applications of the facility location problem is shown in [28] where they utilised the spatial information and studied the difference between the optimal facilities locations and the current ones. Another application is in siting rescue boat locations. The problem was proposed as a multi-objective problem to consider not only the response time to the incidents, but also operating cost and working hours [29]. From these recent research, the distance is one crucial quality as it reflects response time and cost saving. This can also be the case in carsharing scene since the walking distance between users and vehicles might encourage the user to use the service. Unfortunately, this important aspect was not considered in the past work such as GSM antenna and patrolling allocation problem or even typical FLP and MCLP.

These two problems are highly related to our fleet placement problem (FPP). The similarity between FPP and MCLP is that they both try to maximise the partial coverage, with a constraint of fixed coverage distance and fixed number of facilities. Meanwhile, FLP objective is to minimise the maximum operating cost which is well aligned with FPP second objective to minimize the maximum walking distance. Therefore, FPP can be seen as a combination of these two problems.

2.1.2 Shared Fleet Placement

Shared fleet placement can be formulated into MCLP or FLP. However, there are other factors to be considered. In previous works, they already have a list of preferred locations. These possible sites are evaluated by considering convenience factors such as parking cost, the proximity to essential facilities and accessing time as presented in [30, 31]. The solution was then a combination of selected sites to maximise user coverage. In fact, they are very similar to the facility location problem. Kumar and Bierlaire evaluated potential stations by the distance between the station and other facilities like hospitals and train stations. They also had access to historical data to make a decision on where to place the station which is not available in our case [30]. Another popular approach is to locate the fleet by user demands [32, 33, 34]. Boyací et al. proposed an optimisation model to maximise the user coverage based on the demand and predicted destinations [32]. On the contrary, Lage et al. studied method to identify the potential of city districts in station-based one-way trip scenario where the demands were estimated from the taxi trips and customer profiles in Sao Paulo, Brazil [33]. Lastly, Schwer and Timpf proposed an idea to locating the fleet in round-trip carsharing by combining both user demands and proximity to other mean of transportation and other facilities into a model and utilised the open source data available from the government [34].

There are also works which focus on electric carsharing fleet as well from the increasing awareness of the environmental movement and support from the government. The electric carsharing fleet is more complicate

than its fossil-fuel counterpart in that there are constraints for batteries and traveling distance of the car. Çalik and Fortz proposed a model for one-way electric carsharing service which considered previously mentioned factors [35]. Since charging is very important in electric carsharing service, Jiao et al. formulated their model to consider a situation where the user change the drop off station [36]. The trend even extended to charging station location optimisation which was presented in [37] where the authors based the location on the source and destination of trips in both simulation and Vienna. In addition, the shared fleet placement is also studied in the bikesharing community where station locations are highly important as well [38, 39, 40, 41]. Another aspect in the carsharing business which can be considered a further step after fleet allocation is the relocation strategy. This can happen to both station-based and free-float models due to the fluctuation in demand. The relocation incurs cost and this cost is also needed to be minimized. Weikl and Bogenberger proposed a relocation strategy based on demand clustering and demands prediction [42]. They also came up with an optimization framework to find the optimal strategies given a situation. Barth and Tood also proposed a model to simulate the demands which can be used to analyze the performance of the carsharing system [43]. Some of these works employed only a raster approach and Euclidean distance to calculate walking distance between fleet vehicles and share mobility users and due to the resterisation, the feasibility and accuracy of the obtained results are in question.

2.2 State-of-the-Art Algorithms

The presented previous work mostly dealt with a single objective optimisation problem or the weighted sum formulation of the multiobjective problem. However, by using the weighted sum approach, the outcome is only limited to one result per configuration (i.e. a set of weights for each objective), not to mention that it is hard to determine the right weights for studied objectives. Therefore, multiobjective optimisation algorithms are getting more attention over the past decades since in real-world scenarios, there are usually more than one objective to be optimised concurrently. The multiobjective algorithms also present the outcome as Pareto front which enables decision makers to choose suitable solutions from diverse set of solutions. In this section, some of the state-of-the-art algorithms are studied and categorised into three categories, (a) exact methods (b) heuristic algorithms and (c) metaheuristic algorithms.

2.2.1 Exact Methods

Exact methods (or algorithms) perform the search over the whole interesting search space, and a problem is solved by dividing it into subproblems which can be solved efficiently. The strength of the exact algorithms is the guarantee of global optimum, but the computational cost is so expensive to the point that it becomes intractable in NP-hard problems. Examples of classical algorithms in the exact methods class are branch and bound, branch and cut or A*. There are also commercial exact solvers such as CPLEX [44] and AMPL [45] which were mentioned in the literature, but to our knowledge, these algorithms and solvers are only able to solve a single objective optimisation problem. For generic location problems such as MCLP and FLP, exact methods are usually used to solve the problem as found in [19, 20, 46, 47, 48, 29, 24]. As for the shared fleet placement, the most common approach is also utilising the exact solvers such as CPLEX or MATLAB [32, 30, 36, 49, 50].It is important to note that the size of problem in those articles were not large. In addition, the multiobjective optimisation problems in the aforementioned works were transformed into weighted sum equations or have ϵ-constraint transformation before being able to be solved with CPLEX and other solvers. This limitation led to research for multiobjective exact solvers and in 2016, PolySCIP was proposed [51].

PolySCIP employs a "Lifted Weight Space Approach" [51]. This approach first optimises the objectives lexicographically. Then weighted (single objective) optimisation problem from the first phase is optimised by using positive weight vectors. This helps the algorithm in exploring the Pareto front in the problem space. If the new non-dominated solution is found, the old solution (the one that has been dominated) is discarded and the process continues until all non-dominated solutions are found. As a result, the outcome is a Pareto front instead of just one solution. The method was proven mathematically to find all global optimum by the authors.

2.2.2 Heuristic Algorithms

Heuristic algorithms can be simply described as set of rules to follow and are normally designed or tailored to solve specific problems and/or instances. The rule can be as simple as taking whatever that is the best at hand (see Algorithm 1). Numbers of variation of heuristic algorithms were used in solving both MCLP and carsharing fleet management. Church and ReVelle first proposed a heuristic algorithm which add one facility location one at a time. The next attempt in solving MCLP was using Lagrangian heuristic algorithm which is a combination of the Lagrangian Relaxation approach and a greedy method [52]. Heuristic algorithms are still being used nowadays as shown in [53] to solve the FLP problem.

Algorithm 1 Greedy search algorithm

```
Data: Number of locations (N), Potential locations(L)
Result: List of selected locations(S)
S ← ∅
 for l ∈ L do
 |  evaluate location l using a fitness function
end
sort locations according to fitness score
 S ← N best fitness location (l)
 Return S
```

The earliest heuristic algorithm was introduced by Church and ReVelle [17] to solve MCLP. It also appeared several times in carsharing management even in the recent research [30, 32]. The concept of this heuristic algorithm in [17] is the iterative search. The algorithm starts with an empty list of locations. Then, in each iteration, each location is evaluated according to the fitness function. The algorithm then add the location with the highest fitness score in the list and that selected location is removed from the location pool, not to be evaluated and selected again in the consequent iteration. This algorithm has a complexity of $O(sn)$ where s is the number of (desired) stations and n is the number of street nodes. The pseudocode of the algorithm is shown in Algorithm 2.

Algorithm 2 Iterative search algorithm

```
Data: Number of locations (N), Potential locations(L)
Result: List of selected locations(S)
S ← ∅
 for n ∈ {1...N} do
 |  for location(l) in L do
 |  |  evaluate location l using a fitness function
 |  end
 |  sort location according to fitness score
 |   S ← best fitness location
 |   remove facilities that are covered by location l
end
Return S
```

Even though, the algorithm was first introduced to solved the single objective optimisation problems, there are several ways to adapt it to solve multiobjective problems, i.e. weighted sum, and ϵ-constraint. These variations can be extended further to yield an approximated front as a result. In weighted sum approach, each objective fitness is normalised as shown in Equation 2.1.

$$F_n = \frac{F_a - LB}{UB - LB}, \tag{2.1}$$

where F_n is the normalised fitness and F_a is the actual fitness (e.g. coverage, walking distance, or bi-objective) before normalisation. UB is the upper bound (the highest fitness) and LB is the lower bound (the lowest fitness). With the weighted sum approach, the priority of the each objectives can be adjusted as shown in Equation 2.2. The iterative search can be launched multiple times with different weight ratios for each optimisation objective. To elaborate on this matter, supposedly there is a list of weight; [(0.1,0.8,0.1), (0.33, 0.33, 0.33), (0.1, 0.1, 0.8)]. With these weights, up to three solutions can be achieved.

$$F_t = \sum_{i=1}^{m} w_i F_i \tag{2.2}$$

Where F_t is the total fitness score from weighted sum and w_i is the weight associated to objective i. Finally, F_i is the normalised fitness score of objective i. The granularity of weights can also be adjusted at the cost of computation complexity since more combinations of weights requires more executions of the algorithm. Once, a certain amount of solutions (decided by a decision makers) are collected, an approximated front can be constructed.

2.2.3 Metaheuristic Algorithms

According to [54], a metaheuristic algorithm are described as a general-purpose optimisation algorithm that can be used to solve almost any optimisation problem. The name, metaheuristic, comes from the fact that this type of algorithm imitates the living organisms such as human genes, and flocks of animals. Metaheuristic algorithms are known for their efficiency in solving optimisation problems. The result is not always optimal, in contrast to the exact methods, but in most cases, close to the optimum. A true benefit is their execution time, which for middle to large size instances is several orders of magnitude smaller than for the exact methods and can also be controlled to some extent. In fact, Zarandi et al. reported that CPLEX (a famous exact solver) cannot handle a problem with a large size of input. Hence, once the problem size is too large, heuristic and metaheuristic algorithms are usually employed [47]. There are a lot of works reporting their efficiency in solving single-objective FLP and MCLP. Tabu Search, (TS), Simulated Annealing (SA), Variable Neighbourhood Search (VNS) and Genetic Algorithm (GA) were considered to solve MCLP [55, 56, 47, 57, 58].

Not only metaheuristic algorithms were used to solve the single-objective version of these problems, several works showed their efficiency in solving multi-objective version (which are more complicate) as well. Multi-objective optimisation can be performed with two methods. The first one is to use a weighted sum approach and the second method is dominance-based approach [59]. In a weighted sum approach, weights are given to each objective based on their priorities. This can be problematic since normally, these weights are not exactly known and required to go through several adjustment to eventually yield satisfying results.

The dominance-based approach handles the multiobjective problem differently. Instead of transforming the multiobjective problem into a single objective one, this approach utilises Pareto-dominance defined as;

$$z \succ z' \Leftrightarrow \forall i \in \{1, 2, \ldots, n\}, z_i \leq z'_i \cup \exists j \in \{1, 2, \ldots, n\}, z_j < z'_j \ ,$$

where z is a solution and indices i and j are indices for the fitness of objective i and j. As a result, in a single run of the dominance-based metaheuristic algorithm, it can produce an approximated front in contrast to the weighted sum approach mentioned in the heuristic category.

Example of multiobjective metaheuristics in solving MCLP are as follows. Xiao et al. employed a multi-objective Evolutionary Algorithm (MOEA) to solve the bi-objective MCLP which focused on facility cost and proximity minimisation using a specific encoding scheme and dedicated operators [22]. Kim and Murray solved the bi-objective reliability-focused MCLP where it aimed to maximise primary and backup coverage with a heuristic algorithm and a multi-objective Genetic Algorithm (MOGA) [23]. Karasakal and Silav utilised the crowding distance function of NSGA-II [60] in SPEA-II [61] and reported that the new algorithm outperformed the original NSGA-II and SPEA-II. [62]. RanjbarTezenji et al. proposed their modified version of NSGA-II and used it to solve bi-objective MCLP [63].

With the further development of metaheuristics community, hybridisation was introduced to improve the performance of metaheuristics in several real-world scenarios through several strategies such as a combination of metaheuristics and (complementary) metaheuristics and a combination of metaheuristics and exact methods. To the extent of our knowledge, most of the existed works in MCLP and its variants were single-objective such as hybrid artificial bee colony [64, 65], hybrid simulated annealing [66] and hybrid variable neighbourhood search [67].

From the survey, metaheuristic algorithms were not used in the shared fleet placement problem and previous works mainly employed exact methods and heuristic algorithms. It was possible due to small sized test problems (at most 800 potential locations) and thus not able to represent the scenario where service providers do not have any potential locations in mind.

In this section, three state-of-the-art multiobjective metaheuristic algorithms are explained, namely, Strength Pareto Evolutionary Algorithm-II (SPEA-II), Non-dominated Sorting Genetic Algorithm II (NSGA-II), and recently introduced algorithm, Reference-Point-Based Non-dominated Sorting Genetic Algorithm (NSGA-III).

Strength Pareto Evolutionary Algorithm-II (SPEA-II)

Strength Pareto Evolutionary Algorithm was first introduced by Zitzler and Thiele and was shown to achieve better solution quality than other state-of-the-art algorithm at the time such as Non-dominated Sorting Genetic Algorithm (NSGA) [68] and Vector Evaluated Genetic Algorithm (VEGA) [69] in Knapsack 0/1 problem [61]. SPEA-II is an improved version of SPEA [70]. SPEA-II addressed three weaknesses in SPEA and greatly improve the performance over its predecessor. These three weaknesses are fitness assignment, density estimation and archive truncation.

In SPEA-II, each individual's fitness value is the sum of its strength raw fitness and a density estimation which is different from SPEA where all individuals that are dominated by the same solution in the archive have the same rank. This turns SPEA into just a random search algorithm. However, with the SPEA-II method, it has been prevented and the solution guiding process (by fitness) is enhanced. Then selection, crossover, and mutation operators are applied to fill an archive of individuals. The non-dominated individuals of the original population and the archive are copied into a new population through the new environmental selection process. The selection process was updated by employing two concepts. The first one is keeping the size of archive at

constant. The second concept is the boundary solutions preservation truncation. Following these two concepts, there are three cases that can occur.

1. If the number of solutions from archive and non-dominated solutions from the population is exactly the archive size, all of them are kept in the archive.
2. If the number of solutions from archive and non-dominated solutions from the population is less than that of an archive size, the dominated solutions in the population are sorted. Then only n-top dominated solutions are selected to fill the archive.
3. If the number of solutions from archive and non-dominated solutions from the population is more than that of an archive size, the truncation will be applied. A truncation operator based on the distances to the k-th nearest neighbour is used which removes a solution which has the minimum distance to another solution in the Pareto front. With this truncation process, the boundary solutions can be preserved.

Algorithm 3. shows the process of SPEA-II.

Algorithm 3 Strength Pareto Evolutionary Algorithm II

```
population ← Initialize (size)
archive ← ∅
while True do
    Evaluate (population)
    Evaluate (archive)
    archive ← EnvironmentSelection (population + archive)
    if termination criteria are False then
        parents ← Tournament (archive)
        population ← Recombination + Mutation (parents)
    end
    else
        Return archive
    end
end
```

Non-dominated Sorting Genetic Algorithm II (NSGA-II)

NSGA-II is another well-known multi-objective metaheuristic. It is largely based on the Genetic Algorithm (GA) starting from population initialisation, selection of parents, crossover, mutation to obtain a new population of solutions [71]. Individuals in both the parent and offspring populations are sorted according to their rank through the fast non-dominated sorting algorithm introduced in [60], and the best solutions are chosen to create a new population.

Apart from the non-dominated sorting algorithm, another important concept in NSGA-II is the crowding distance which is the euclidean distance between neighbour solutions in n-th dimension (where n is the number of objectives) and the crowding distance of boundary solutions are infinity. This crowding distance is used to guide the the algorithm toward a uniformly spread out Pareto front. This concept is also used in tournament selection (tournamentDCD) and a special operator introduced in NSGA-II, crowded comparison operator. In the parent selection, tournamentDCD is applied where the crowding distance is used to cut the tie in case both solution are non-dominated. In crowded comparison operator, there are two cases in choosing a solution with crowding distance.

1. If it is between solutions with different ranks (after non-dominated sorting), the lower rank solution is prefer.

2. If it is between solutions of the same rank, the solution which is located in a less crowded region (only a few neighbour solutions) is preferred.

With this strategy, NSGA-II is able to preserve the boundary solutions and at the same time, it adds diversity in the Pareto front. NSGA-II algorithm is presented in Algorithm 4.

Algorithm 4 Nondominated Sorting Genetic Algorithm (NSGA-II)

```
population ← InitialisePopulation (size)
Evaluate (population)
NondominatedSort (population)
CrowdingDistance (population)
while termination criteria are False do
    parents ← TournamentDCD (population)
    offspring ← recombination+mutation(parents)
    Evaluate (offspring)
    NondominatedSort (population + offspring)
    CrowdingDistance (population + offspring)
    population ← CrowdedComparison (population + offspring)
end
Return population
```

Reference-Point-Based Non-dominated Sorting Genetic Algorithm (NSGA-III)
NSGA-III is also based on GA. It was also proposed by Deb et al [72]. Sharing the similar name to NSGA-II, NSGA-III employs a completely different approach. The algorithm revolves around the concept of reference points. The algorithm was also reported to be highly efficient (in comparison to two variations of MOEA/D [73]) in benchmark solving multiobjective problem, e.g., DTLZ and WFG. (where there are three or more objectives). The recent study also reported a superior hypervolume of NSGA-III over NSGA-II in various benchmark problem such as DTLZ, knapsack and distance minimisation problems [74]. The original NSGA-III is restricted to three or more objectives optimisation problem only and this limitation was unlocked in U-NSGA-III [75]. In this research, NSGA-III is referred to the original version as presented in [72].

Unlike NSGA-II which utilises crowding distance to preserve diversity in the solutions, NSGA-III is more radical in that it is designed to be able to receive the Pareto front or preferred reference points from the user. This enables NGSA-III to keep explore the search space. In the absence of the supplied reference points, predefined structured placement of any reference points can be used in place. In [72], Das and Dennis' method was used [76]. The hyperplane and referenced are constructed from the sorted Pareto fronts from the same non-dominated sorting algorithm in NSGA-II. There are also two more special processes in NSGA-III. The first process is Association. The Association process is used to map solutions in the sorted fronts with the reference points which the result is used in the second process, Niche-Preservation. The preservation keeps adding reference point associated solutions to the next generation until the whole population is filled to the declared size. Algorithm 5 demonstrate the overall procedure of NSGA-III. Readers are referred to [72] for more details on the processes.

2.3 Multiobjective Performance Indicators

In the case of single objective optimisation problem, it is simple to directly compare the quality of the solutions yielded by various algorithms to the optimum. However, in multiobjective optimisation, the solving algorithms yield a set of solutions, hence the direct comparison cannot be applied. With the vigorous research community,

Algorithm 5 Reference-Point-Based Non-dominated Sorting Genetic Algorithm (NSGA-III)

```
Population ← InitialisePopulation (size)
Evaluate (population) while termination criteria are False do
    parents ← TournamentSelection (population)
    offspring ← recombination+mutation(Parents)
    (F_1, F_2, , ...) ← NondominatedSort (population,offspring)
    while |nextGeneration| ≤ size do
        nextGeneration = nextGeneration ∪ F_i
        i = i + 1
    end
    if |nextGeneration| == size then
        population ← nextGeneration
    end
    else
        F_l ← F_i
        nextGeneration ← (F_1 ∪ F_2 ∪ ... ∪ F_{l-1})
        referencePoints ← createHyperplane(nextGeneration)*
        Associate(nextGeneration, referencePoints)*
        K = size − (|tmpPopulation|)
        population ← Niching (K, referencePoints, F_l, nextGeneration)

    end
end
Return population
```

several indicators were proposed in the past decades. These indicators can be categorised based on the quality aspect that they assess, i.e. the convergence (distance to the optima), diversity aspects and both of convergence and diversity altogether [54].

In this work, three indicators are chosen. These selected indicators measure different aspects of the yielded solutions, namely, inverted generational distance (IGD) [77], spread [60] and hypervolume [54]. They are only applied on approximated Pareto front (solutions from heuristic and metaheuristic algorithms) since true Pareto front (from exact methods) is normally used as a reference in these indicators.

Inverted generational distance (IGD) [77] measures the distance between the obtained Pareto solutions and the optimal ones. IGD is defined in Eq. 2.3, where d_i is the Euclidean distance from point i in the Pareto front approximation to the closest one in the optimal Pareto front, and n is the number of solutions in the Pareto front. *IGD=0* indicates that the evaluated Pareto front consists only of solutions from the optimal Pareto front.

$$IGD = \frac{\sqrt{\sum_{i=1}^{k} d_i^2}}{n} \quad . \tag{2.3}$$

Spread [60] measures the diversity of the obtained front and is defined as:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (N-1)\bar{d}} \quad , \tag{2.4}$$

where d_i is the Euclidean distance between consecutive solutions, $\bar{d}$ is the mean of these distances, and d_f and d_l are the Euclidean distances to the *extreme* solutions of the optimal Pareto front. A zero value indicates an

ideal distribution, i.e. pointing out a perfect spread of the solutions in the Pareto front.

$$HV = \text{volume}\left(\bigcup_{i=1}^{|Q|} v_i\right) \quad . \tag{2.5}$$

Hypervolume [78] assesses both convergence and diversity of a Pareto front. It calculates the m-dimensional volume (in the objective space) covered by the solutions in the evaluated Pareto front Q and a dominated reference point W. For each solution $i \in Q$, a hypercube v_i is constructed with the reference point W and the solution i as the diagonal corners of the hypercube. The hypervolume is calculated as the union of all hypercubes, as shown in Eq. (2.5). The higher the hypervolume the better the algorithm performed.

The previous work and the progression of the classical problems to the various real-world applications are discussed. Several state-of-the-art algorithms are also introduced to establish the foundation for the experiments in this research. In the next chapter, the formulation of the fleet placement problem (FPP) is proposed with the experiments and analyses. The results are discussed to dissect and study the behaviours of the algorithms and how they react to different instances.

Chapter 3

Carsharing Fleet Placement Optimisation and Metaheuristic Algorithms

This section explains the methodology in fleet placement problem (FPP) and vehicle placement problem (VPP). First, the definition of the graph used in this work and a benchmark suite for further analysis is given along with the explanation on using demographic and other open data to enhance the accuracy of our work. Then, the study is separated into three phases. Diifferent optimisation objectives are studied and also different algorithms are used during the evaluation process in these phases. Finally, a hybrid metaheuristic algorithm and a novel problem specific operator to efficiently solve the optimization problem is proposed.

3.1 Graph Definition

Our problem instance is defined as an undirected weighted graph to adhere to the reality where users can walk in both directions to access a vehicle in contrast to the direction of roads or streets. The formulation of this graph is:

$$G = (V, E, P, D, W).$$

V is a set of vertices in a graph composed of three subsets: $V = S \cup B \cup T$, where S is a a subset of potential vehicle locations on streets, B is a subset of residential buildings and T is a subset contain public transport locations. The members in these subsets are modeled as vertices in a graph, hence a member of V. In this research, only bus stops, train stations and bus stations are considered because they are more relevant to the placement problem scenario.

The set of edges E represents roads and streets. Each edge $(u, v) \in E$ $(u, v \in V)$ has a weight of $d \in D$ which signifies a walking distance from u to v where u and v are street nodes or potential vehicle locations in S.

A building in the subset B accommodates users since it was shown that users normally walk to vehicles and public transportation from their residence in [30, 79, 80]. The number of users in each building is represented by a weight $p_j \in P$ $(p_j \geq 0)$ which associates with a building $b_j \in B$.

FIGURE 3.1: The example graph instance for VPP.

The last subset is W that contains the importance factor $w_k \in W$ of the public transport locations $t_k \in T$. Willing et al. formulated the model to predict the free-float carsharing demands based on the past vehicle rental data and its proximity to points of interest (POIs)[81], e.g. the proximity to the bus stop encourage a higher usage of the shared vehicle. Although, it is not of the same scenario, This is a good starting point to incorporate the effect of establishment in the proximity into our model. The coefficients of relevant transportation types are taken from [81] to be used in our work. The importance factor for each type of public transport is defined as $\{bus\_stop : 1, bus\_station : 1, train\_station : -0.54\}$.

The example of the graph instance is presented in Figure 3.1. There are two types of nodes in the figure. The yellow nodes represent the buildings and bus stops. The blue nodes represent the potential vehicle locations and lastly, the grey lines represents roads and street.

3.2 Benchmark Suite

This section provides details on the construction of the two types of problem instances in our benchmark suite. The first one consists of synthetic instances, while the other ones are large-scale real-world instances.

3.2.1 Synthetic Instances

There are three types of nodes in our graph instances, i.e. street, building and public transport nodes. In order to generate the most realistic problem instances, the CityEngine [82] is used to generate the base city graph. The distances between street nodes are assigned uniformly at random between 10 and 500 meters. The ratio of buildings and public transport locations over the number of street nodes has been observed from 30 major cities in Europe, e.g. Paris, Athens and Madrid, using data from Openstreetmap [7]. The outcome ratios are shown in Table 3.1.

For each instance, the number of residential buildings and public transport location nodes are then generated uniformly at random using the ratios observed in 30 cities. Finally, these additional nodes are randomly attached to edges in the graph. The population of users in the generated instances are estimated between 1-2% of the total building number and distributed uniformly among all buildings in the instance to make the instance sparser. Through this method, a total of 100 synthetic instances are created. There are 10 different sizes, i.e.

Table 3.1: Number of street nodes and public transport locations in observed major cities in Europe.

City	Street	Building	Bus Stop	Bus Station	Train Station
Amsterdam	43113	228551	942	9	41
Athens	25755	107549	1276	7	31
Barcelona	273956	178772	4947	33	370
Berlin	186813	380156	6015	7	275
Birmingham	55557	243103	4094	2	34
Brussel	50451	250602	1838	4	104
Bucharest	40369	70222	1274	33	67
Budapest	87926	124251	4444	17	79
Cologne	80688	284052	1331	3	18
Copenhagen	27845	72942	614	13	36
Dortmund	74224	225422	1556	8	16
Frankfurt	51699	113894	1376	5	69
Greater London	299779	390721	20084	21	459
Hamburg	131338	336143	4150	6	128
Helsinki	124472	64946	2443	9	34
Kiev	108163	74599	2061	18	72
Lisbon	126355	268361	1808	56	119
Lyon	111096	477717	3804	6	81
Madrid	104943	111236	4986	13	244
Milan	166968	310803	4630	20	159
Munich	118732	164782	2060	3	112
Oslo	69866	84698	626	12	61
Paris	36106	102325	2458	4	283
Prague	113138	168513	2564	12	87
Rome	88658	224311	8373	69	148
Rotterdam	43755	214176	836	4	46
Stockholm	232343	359400	10420	18	214
Turin	21045	25439	2342	3	27
Vienna	101964	236142	3043	8	130
Warsaw	142305	116085	3860	25	48
Ratio		[0.964, 3.748]	[0.012, 0.061]	[0.0001, 0.0003]	[0.0002, 0.0026]

5,000, 10,000, 15,000, 20,000, 25,000, 30,000, 35,000, 40,000, 45,000, and 50,000-node. Each size contains 10 different instances with different city layout from CityEngine software.

3.2.2 Real City Instances

The benchmark suite includes four large-scale instances; Munich, Hamburg, Dusseldorf and Berlin as these cities are targets for the collaborated carsharing company. These instances are obtained by combining four elements: footprints of the buildings, public transport locations, land use data and street network. All these data are extracted from OpenStreetMap [7]. Other existing works utilised some of these data such as land use data [83] and street-level map [84], but none of them had used all four data sources together. The distinguish features of this research are; (1) locating residential buildings using land use data (2) extracting the public transport locations (3) importing street-level maps as graphs using OSMnx [85] and (4) linking buildings and public transport locations to the base graph using OSMnx and OSRM [86] which permit a realistic walking distance calculation. The number of shared mobility customers is estimated at 2% of the city population and customers are distributed uniformly on a district-basis to residential buildings.

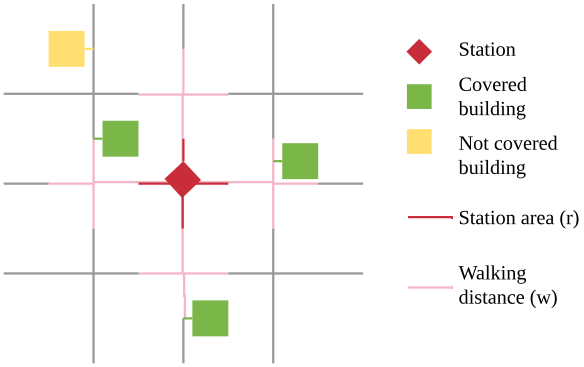


FIGURE 3.2: Illustration for virtual area of a station on streets where the diamond shape is a center point and squares are buildings. Green coloured squares are covered buildings, yellow coloured one, otherwise. Station radius is denoted by r and maximum walking distance for coverage is denoted by w.

3.3 First Phase: Initial Study and Algorithms Screening

In the first phase of this study, the aim is to understand the fleet placement problem (FPP) and study the effectiveness of state-of-the-art algorithms (and solvers) in solving the problem. With these purpose, A bi-objective optimisation model to determine the locations of the fleet in the city is proposed. These locations are called stations where shared vehicles can be picked up and returned. The two devised objectives for this first phase are (1) maximising the user coverage and (2) minimising the maximum global distance from every fleet location to its covered users. The motivation behind the first objective is to maximise the outreach of the service to the users. While, the motivation behind the second objective is to make it convenient to access the service which is one of the important aspect studied in [87].

3.3.1 Optimisation Model

In this model, carsharing stations are placed on the street. A station is a virtual area on the city map defined by two elements: a center point and a radius. The center point is a street node and the radius indicates the size of the station (denoted by r). The coverage of a station is determined by the given maximum walking distance (w). The virtual area is illustrated in Figure 3.2.

Due to the round-trip nature of the service, each car taken from the station needs to be returned to this station after completing the trip. The typical customers are people residing in residential areas covered by the carsharing stations, who walk to pick up the nearest vehicle. Under the aforementioned assumptions and constraints, the fleet placement problem is described as follows.

3.3.2 Input Parameters

In the fleet placement problem (FPP), stations are put on the street node in $S = \{s_1, s_2, \ldots, s_n\}$ and has a common size of r (see Figure 3.2). Users are assumed to start the trip from their residence (referred as building) in $B = \{b_1, b_2, \ldots, b_m\}$ where each residence or building has different number of users defined as $P = \{p_1, p_2, \ldots, p_m\}$. Each street node cover a set of buildings which is in accordance to the defined walking distance w and distance matrix D. A station s_i covers a building b_j *iff* that station locates at most $w - r$ meters from the building. All the parameters in FPP are described in the following.

- n: Number of street nodes.

- m: Number of buildings.
- f: Maximum number of desired fleet stations.
- w: Maximum walking distance allowed.
- r: Station area radius.
- i: Index for street nodes (belongs to $\mathbb{Z}^+$).
- j: Index for buildings (belongs to $\mathbb{Z}^+$).
- S: Set of street nodes, $S = \{s_1, s_2, \ldots, s_n\}$ and indexed by i.
- B: Set of buildings, $B = \{b_1, b_2, \ldots, b_m\}$ and indexed by j.
- P: Population, $P = \{p_1, p_2, \ldots, p_m\}$, such that p_j is the population of building $b_j \in B$.
- $D_{n\times m}$: A distance matrix between street nodes $\in S$ and buildings $\in B$. D_{ij} denotes walking distance between node s_i and building b_j where a distance function $d(s_i, b_j) = D_{ij}$.

3.3.3 Outputs

The outputs of FPP are the state of building node, street node and the maximum global walking distance. These variables reflect the the output from the solution and are explained as follows.

- c_j represents the fact that building b_j is or is not covered by a station ($b_j \in B$). If the building b_j is covered then $c_j = 1$ and $c_j = 0$ otherwise. A building is covered when the center point of one station (at least) is located within walking distance smaller than $w - r$.
- s'_i represents the state of node s'_i ($s_i \in S$). A node is active if it is the center point of a station. If s'_i is active then $s'_i = 1$ and $s'_i = 0$ if s'_i is inactive. It is important to note that a station can be active even if it does not cover any buildings. In this version, the station is able to accommodate only one vehicle at a time.
- z: maximum walking distance from every selected stations to their covered buildings.

3.3.3.1 Optimisation Objectives and Description

In the fleet location optimisation problem, the objectives are to maximise the coverage or number of covered users and to minimise the maximum global walking distance between users and fleet stations. With these two objectives, the optimisation model is formulated as follows:

$$
\begin{aligned}
&\text{Maximise} && \sum_{j=1}^{m}(c_j \times p_j) \\
&\text{Minimise} && z = max_{i,j}(s'_i \times c_j \times D_{ij}) \\
&\text{subject to} && \sum_{i=1}^{n} s'_i \leq f && (1) \\
& && c_j = \begin{cases} 1 & \text{if } D_{ij} \times s'_i \leq w - r \cup s'_i = 1 \\ 0 & \text{otherwise} \end{cases}; && (2) \\
& && z \geq s'_i \times c_j \times D_{ij} && (3) \\
& && i \in \{1, \ldots, n\} && (4) \\
& && j \in \{1, \ldots, m\} && (5)
\end{aligned}
$$

Constraint 1 denotes that the number of stations cannot exceed the provided number f of fleet stations. Constraint 2 restricts the model to consider any covered buildings b_j only once and the building b_j is covered *iff* there is at least one active station in proximity to the building b_j. Constraint 3 finds the maximum walking distance of the active station s_i to all building b_j that it covers. Finally, constraint 4 and 5 denote the domains of indices i and j accordingly.

3.3.4 NP-hardness Proof

According to [88], any decision problem that can be reduced from an NP-complete problem, whether it is a member of NP or not, is not solvable in polynomial time unless P=NP since it is as hard as the NP-complete problem. In order to prove the NP-hardness of FPP, its computational complexity is analysed. Therefore, the decision counterpart of fleet placement problem (FPP) – FPP–D is introduced. The decision counterpart FPP–D inherits all parameters from FPP.

In this section, the NP-hardness of FPP through proving the NP-completeness of FPP–D is demonstrated. For FPP–D, the question is to determine whether there exists a solution with f station(s) such that all buildings are covered.

Proposition 1 The FPP is NP-hard in the strong sense even if there is only one user in each building.

Proof. A polynomial-time transformation to the FPP–D from the strongly NP-complete problem "Set Cover Problem (Minimum Cover Problem)" is introduced [89, 88].

Set Cover Problem or SCP can be defined as follows: given a universe U of R elements, a collection of subsets of U, $G = \{g_1, g_2, g_3, ..., g_L\}$ and a positive integer $K \leq |G|$, the question is "Does G contain a cover for U of size K or less, i.e., a subset $G' \subset G$ with $|G'| \leq K$ such that every element of U belongs to at least one member of G'?"

Given an instance of SCP, the following instance of FPP–D is introduced. Firstly, let all buildings in B be the equivalence of universe U in SCP and $m = |B| = R$. Then, let S be the direct transformation of collection G where $S' = \{s'_1, s'_2, s'_3, ..., s'_L\}$, such that $s'_l = g_l, l = 1, 2, 3, \ldots, L$, hence $n = L$. In addition, assuming $w = 1$ and $r = 0$ so that the building is covered if it is connected to the location (s_l). With the prior assumption, the distances in matrix D are assumed to be one if the location is a 1-hop neighbour of the building and zero, otherwise. Therefore matrix D reflects the membership of S and is used to constitutes the membership of

TABLE 3.2: Spearman correlation between two objectives. Objective 1 is user coverage maximisation. Objective 2 is global walking distance minimisation.

	Correlation Coefficient	p-Value
Objective 1 - Objective 2	0.977	2.2e-16

collection S in FPP–D. Next, let $p_j = 1; j \in \{1, 2, 3, \ldots, |B|\}$ which means there is only user in building j. Finally, the threshold value $f = K$ is set.

Let X be a solution to SCP. A solution for FPP–D is constructed in which the buildings in B (U) are covered by f stations where $s'_l = g_l \in X$, such that $x_l = s'_l$, if $s'_l \in X$ and $x_l = \emptyset$ if $s'_l \notin X$. Since X is a cover of U (in SCP), all buildings in B are covered and the number of stations in the corresponding solution (for FPP–D) is $f = |X|$.

Now assume that there exists a solution Y in FPP–D with $|Y| \leq K$ and $|Y|$ should not exceed K, otherwise, $|Y| > K$ and the condition will not hold. Therefore, there are at most K station(s) with $y_i \neq \emptyset$. Since all buildings forms B and all buildings in B belongs to at least one member of Y, the selected stations with $y_i \neq \emptyset$ represents a solution to SCP, given a polynomial transformation from SCP to FPP-D. Since all input numbers in the FPP–D instance have size at most polynomial in the size of the input, FPP is strongly NP-hard. □

SCP (as an optimisation problem) was proved to be polynomially non-approximable within the ratio $c \cdot \ln |G|$, for some constant $c > 0$ [90]. Therefore, the following statement is proposed.

Statement 1. There exists no polynomial $(c \cdot \ln n)$-approximation algorithm for the FPP where n is the input size, unless $P = NP$.

3.3.4.1 Objectives Correlation

The conflict between the two objectives of FPP, namely, (1) maximising user coverage (Objective 1) and (2) minimising global walking distance (Objective 2), are proven in this section, using Spearman rank correlation coefficient [91] which is a non-parametric rank test with the confidence interval of 95%. P-Value is used to decide whether accept or reject the null hypothesis as it reflects the probability of obtaining the observed results of a test, assuming that the null hypothesis is correct. Hence, by this definition and the confidence interval of 95%, if the observed p-value is less than 0.05, the null-hypothesis is rejected.. The spearman coefficient is in $[-1, 1]$ where -1 indicates the negative correlation between two data, meaning while x increase, y decrease and vice versa (conflicting). 0 means there is no association between two data. Finally, 1 means there is a positive correlation, for instance, x increases with y or x decreases with y. The closer the coefficient to either side, the stronger the correlation is.

Table 3.2 shows that the coefficient is 0.977. This indicates very strong positive correlation between the two objectives. It means that as the user coverage increases, the global walking distance also does. However, in the context of FPP, this indicates the conflict in the two objectives since the aim is to find a set of high user coverage station locations that still yields low global walking distance. Therefore, with this correlation analysis, the two objectives in FPP are conflicting.

3.3.5 Methodology

Location problem has been studied for several decades already. Hence, there are plethora of existing algorithms. For this initial algorithms evaluation, one representative from each category (exact methods, heuristic and

metaheuristic algorithm) is selcetd, namely, PolySCIP [51], greedy and iterative heuristic [17], and NSGA-II [60]. PolySCIP is the only exact multiobjective solver in the market. As for the iterative heuristic algorithm, even though it was proposed decades ago, it was still used in many recent works. A simple greedy algorithm is also introduced to show the difference between complex and simple behaviours. Finally, NSGA-II is selected since it was shown to be more efficient than Simple Evolution Algorithm for multi-objective Optimisation (SEAMO) [92], Strength Pareto Evolutionary Algorithm II (SPEA-II) [70] and Pareto Envelope-based Selection Algorithm (PESA) [93] in MCLP variant [26]. NSGA-III is not considered in this case because it can only be applied on problems that have three or more objectives [72]. The details and pseudocode of selected algorithms can be found in section 2.2.

3.3.5.1 Script for PolySCIP

In order to use PolySCIP, an optimisation model must be transformed into a program accepted by PolySCIP. This can be done through Zimpl [94]. The tool can translate an optimisation model into an integer programming that can be solved with PolySCIP, CPLEX and AMPL variants as well. For this research, the script is written in Zimpl as shown in Listing 3.1 to express the optimisation models. Some additional parameters are introduced to overcome the limitation in the script.

3.3.5.2 Weighted Sum for Heuristic Algorithms

Two heuristic algorithms are employed in this stage, Greedy and Iterative. In order to use both variants with the multiobjective fleet placement problem, the weighted sum approach is used. There are three weight vector for testing in this phase which are [1.0, 0.0], [0.5, 0.5], and [0.0, 1.0] where the first weight is for the user coverage objective and the second weight is for the global distance objective. With these weights, three solutions are expected from each heuristic algorithm.

3.3.5.3 Solution Encoding and Operators in NSGA-II

The focus is on explaining the method of solution encoding and employed genetic operators in NSGA-II since for this phase of the study. The solution encoding in this problem along with population initialisation, selection process, crossover and mutation are described as they are important components in NSGA-II.

Solution Encoding Due to the extensive amount of fleet locations in a city (can easily reach 100,000 locations in big cities), binary encoding is not efficient. An integer encoding based on the street node ID (unique to each location) is proposed. The length of the solution is equal to the desired amount of station to be opened. In this encoding, there is no order in the encoding, hence, the locations are sorted in descending order in the solution to reduce the search in the solution space. To elaborate on this, supposedly, there are two solutions [3,2,1,4] and [1,2,3,4] and both of them have the same fitness score. By sorting them, this incident is prevented.

Population Initialisation: A solution is initialised by choosing uniformly at random street nodes from all street nodes (in a problem instance) without replacement meaning each location in the solution is unique at the initialisation.

```
set street := {1..N_s};
set building := {1..N_b};

param Cover[street*building] = [...]

param p[N_b] = [...]

param Distance[street*building] = [...]

var st[street] binary;
var oc[building] integer >= 0 <= card(street);
var cb[building] binary;
var a[street*building] binary;
var z real >= 0 <= 500;

maximize obj1: sum <i> in building: cb[i]*p[i];
          obj2: -1*z;

subto c1:
    sum <i> in street: st[i] == N_st;

subto c2:
    forall <i> in building:
        oc[i] == sum <j> in street: Cover[j,i]*st[j];

subto c3:
    forall <i> in building:
        vif oc[i] >= 1 then
            cb[i] == 1
        else
            cb[i] == 0
        end;

subto c4:
    forall <i,j> in street*building:
        a[i,j] <= st[i]*Cover[i,j];

subto c5:
    forall <j> in building:
        sum <i> in street: a[i,j] == cb[j];

subto c6:
    forall <i,j> in street*building:
        z >= a[i,j]*Distance[i,j];

subto c7:
    sum <i,j> in street*building: a[i,j] >= 1;
```

Listing 3.1: Optimization models in Zimpl language.

Selection The selection process for crossover is based on tournamentDCD operator which is proposed in [60] which uses crowding distance to cut the tie in case both candidates are non-dominated.

Crossover The recombination process in this work is based on a two-point crossover. The process randomly selects two points in both solutions as starting and ending points for exchanging portions and recombines these portions to create two new solutions as shown in Figure 3.3.

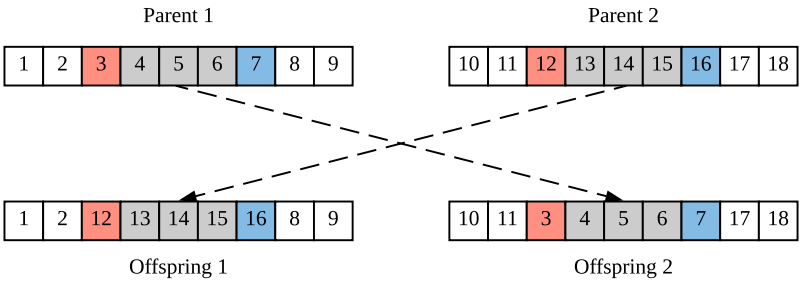


FIGURE 3.3: A two-point crossover process.

Mutation The uniform mutation is applied in this work with a condition that the replacement comes from the pool of all vehicle locations (defined by street node IDs). Figure 3.4 shows the mutation process where *Sample* is a function to randomly pick one location from the pool and $1, 2, 3, \ldots, n$ denotes all street node IDs. However, if the replacement already exists in the current solution, the process is repeated until a valid replacement is found.

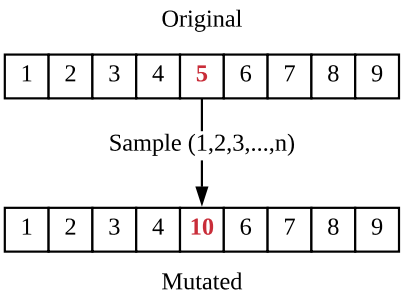


FIGURE 3.4: A uniform mutation process.

3.3.6 Experimental Setup

The performances of PolySCIP, heuristic algorithms (simple and iterative) and NSGA-II algorithms are compared in three scenarios; LU1, LU2 and MU1. LU1 contains a Luxembourg city district containing 63 street nodes and 47 buildings. LU2 is consisted of a map of two Luxembourg city districts that contain 2,026 street nodes and 1,063 buildings combined. MU1 is an inner part of the city of Munich (extracted from the Munich instance in the benchmark suite), which contains 16,075 street nodes and 21,816 buildings.

For the LU1 setup, the radius of the carsharing station (D_r) is set to zero and set the maximum walking distance for users (D_w) to be 150 meters for all evaluated algorithms. The number of stations is set to four. Being simplistic the LU1 setup is ideal for observing and understanding operational details of the evaluated algorithms.

Table 3.3: Evaluation scenarios.

	LU1	LU2	MU1
City	Luxembourg	Luxembourg	Munich
Population	561	11,439	17,486
Number of carsharing stations (N_s)	4	10	100, 72
Number of street nodes	63	2,026	16,075
Number of residential buildings	47	1,063	21,816
Maximum walking distance (D_w)	150 meters	500 meters	500 meters
Carsharing station area radius (D_r)	0 meters	100 meters	100 meters

The size of the LU2 graph is chosen to be sufficient for performance evaluation in realistic district scenarios. The maximum distance people are willing to walk (walking distance) depends on the selected mode of transportation. Daniels and Mulley [79] show that people are willing to walk significantly longer to take a train than a bus as long as they deem it worthy [79]. For carsharing a realistic acceptance walking-to-the-car distance (D_w) is around 500 meters [13]. The population is selected to match the real numbers reported by the city [95] and is distributed uniformly. The area covered by a carsharing station has a radius (D_r) of 100 meters, while the number of stations is set to 10.

The MU1 scenario aims to evaluate performance of algorithms on a city-wide scale. The population in each district is taken from OpenStreetMap [7] and distributed to only among residential buildings. The number of carsharing users is estimated to be 2% of the total population (coming from the carsharing company). The station radius (D_r) and walking distance (D_w) are the same as in the LU2 scenario, but the number of stations is increased to 100 stations.

In all three scenarios, carsharing users are distributed uniformly among available buildings on the map. Since there is a disparity in the size of scenarios, parameters are adjusted accordingly as shown in Table 3.3. Please note that all deterministic algorithms, e.g. PolySCIP and heuristic algorithms, are executed only once, while NSGA-II is executed for 30 times. In addition, heuristic algorithms and PolySCIP do not require any other parameters apart from those mentioned in Table 3.3. Therefore, only configurations for NSGA-II are mentioned. Pareto fronts from multi-objective algorithms are compared in term of convergence and diversity using inverted generational distance (IGD), spread and hypervolume indicators mentioned in Section 2.3.

The configuration of NSGA-II for LU1 is presented in Table 3.4. The population in each generation is 20 individuals. The termination condition is 400 generations which results in 8000 evaluations. Individuals are selected for the crossover process by the binary tournament selection. The crossover method is a 2-point crossover with a rate of 0.8. The mutation process replaces selected genome by a random street node. The mutation rate is 0.01.

In the other two scenarios, the same operators as in the LU1 case are used. However, the population size is increased to 50 and 100 for the LU2 and MU1 scenarios respectively. The crossover rate is set to 0.8 and the mutation rate is 0.01 for the LU2 scenario, while the crossover rate and mutation rate for the MU1 scenario are 0.8 and 0.01. Solutions from heuristic algorithms are also used as seeds for NSGA-II to improve the quality of multi-objective solutions. These configurations are shown in Table 3.4. All experiments were carried out on the high performance computing platform of University of Luxembourg (UL HPC) [96]. The machine specification is Intel Xeon Gold 6132 (2.6 GHz) with 128 GB of memory.

TABLE 3.4: NSGA-II configuration parameters. Population size is increased as a graph becoming larger. The mutation rates are also modified to accommodate the number of stations in a solution.

	LU1 scenario	LU2 scenario	MU1 scenario
Number of generations	400	400	400
Population size	20	50	100
Selection process	Tournament	Tournament	Tournament
Crossover operator	2-point crossover	2-point crossover	2-point crossover
Crossover rate	0.8	0.8	0.8
Mutation operator	uniform	uniform	uniform
Mutation rate	0.01	0.01	0.01

3.3.7 Experimental Results

The results of each phase are represented as a scatter plot where the x-axis represents the maximum walking distance (lower is better) and the y-axis represents the number of covered users (higher is better).

3.3.7.1 LU1 Scenario: Validation

Figure 3.5 presents the obtained Pareto fronts from eight different algorithms. The PolySCIP Pareto front is called the true Pareto front. Hence, it is used as the reference Pareto front in the Inverted Generational Distance (IGD) indicator. The two extreme points in the true Pareto fronts are also used in normalisation in this section. Table 3.5 contains numerical results obtained for all evaluated algorithms. To simplify the table, only two extreme points on both Pareto fronts are shown. The highest achieved coverage is 391 users which is yielded by iterative heuristic algorithm, the exact method, and NSGA-II. On the other hand, the lowest distance of 93.5 meters is achieved using PolySCIP.

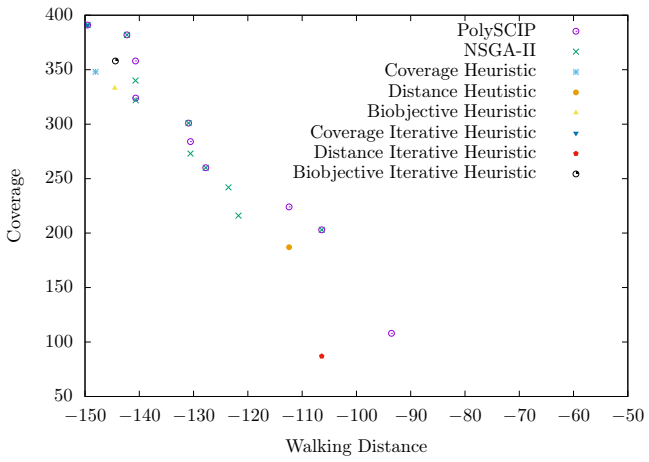


FIGURE 3.5: Pareto fronts and single objective solutions from all presented algorithms mentioned in Section **??** in the validation phase. The higher the value on x axis (moving toward the right), the better the distance objective is. The higher the value on y axis, the better the user coverage objective is.

NSGA-II performs well with both objectives showing results close to and at the optima. It can achieve the optimal result for user coverage and the extreme solution for distance is close to the optimum (106.4 from NSGA-II and distance-focused iterative heuristic and 93.546 meters from PolySCIP). In fact, the distance result from NSGA-II is even better than its iterative counterpart as it covers more users. Other heuristic algorithms achieve high-quality solutions too. Figure 3.6 shows the stations and their respective coverage in LU1 scenario.

TABLE 3.5: Numerical results in LU1 scenario. Only extreme solutions from two Pareto fronts are mentioned.

	Covered Users	Maximum Walking Distance (meters)
PolySCIP (Best coverage)	**391**	149.528
PolySCIP (Best distance)	108	**93.546**
NSGA-II (Best coverage)	**391**	149.528
NSGA-II (Best distance)	203	106.4
Coverage Heuristic	348	148.491
Distance Heuristic	187	112.398
Bi-objective Heuristic	333	144.515
Coverage Iterative Heuristic	**391**	149.528
Distance Iterative Heuristic	87	106.4
Bi-objective Iterative Heuristic	358	144.401

FIGURE 3.6: A map showing a solution from NSGA-II that yields the highest user coverage and the highest maximum walking distance. Covered buildings are depicted as nodes inside polygons.

TABLE 3.6: Comparing Pareto fronts for PolySCIP and NSGA-II.

	Inverted Generational Distance (IGD)	Spread	Hypervolume
Exact method	True Pareto front	0.488	0.449
NSGA-II	3.02	0.525	0.351

The different Pareto fronts are hard to be compared from the plot alone, hence, IGD, spread and hypervolume indicators are used (see Table 3.6). For IGD indicator, NSGA-II Pareto front yields 3.02. IGD is not available for the exact method Pareto front because it is used as a reference. This value is in accordance to the plot where the two Pareto fronts are not the same. In Figure 3.5, it can be seen that NSGA-II achieved some of the solutions on the true Pareto front, especially, the highest user coverage solution.

As for spread indicator, the true Pareto front yields 0.488, while the NSGA-II Pareto front yields 0.525. This means the diversity in the exact method Pareto front is higher than in NSGA-II. It was due to the fact that the coverage objective overwhelmed the distance objective leading to a cluster of solution in the upper right region in Figure 3.5. The hypervolume of the true Pareto front is 0.449 and the NSGA-II Pareto front yields 0.351. The difference occurs because some solutions of NSGA-II are dominated by PolySCIP's.

3.3.7.2 LU2 Scenario: Limitation Analysis

The size of LU2 did not allow PolySCIP to obtain solution for two objectives in reasonable time. Solutions were not found even after 18 days on an instance of size 1,390 street nodes and 1,063 buildings. Figure 3.7 confirms that computation time of PolySCIP is exponential. Four test scenarios from LU2 scenario are extracted

in different sizes (street nodes x residential buildings); (a) 63x47 (b) 222x307 (c) 366x584 (d) 727x876 (e) 1390x1063. The number of stations is set to four for instance a and b and 10 stations for the rest. However, It is found that the number of stations does not affect the execution time in the least. Therefore, in Figure 3.8, there is only one Pareto front from NSGA-II. It shows that NSGA-II yields a higher coverage than the iterative heuristic coverage algorithm when it takes the algorithm's solution as a seed. The highest achieved user coverage is at 8,421 users with the maximum walking distance of 399.8 meters. On the other hand, the lowest maximum walking distance achieved is 135.7 meters with only 47 covered users. It is obvious that increasing the number of covered users is a priority. It can be observed in Fig 3.8 that even though some residential buildings are located close to carsharing stations, they are not covered. This is because the entrances of those buildings (determined during snapping process) are mapped on the opposite streets which are not covered by the stations. However, the number of such buildings is marginal and can be neglected..

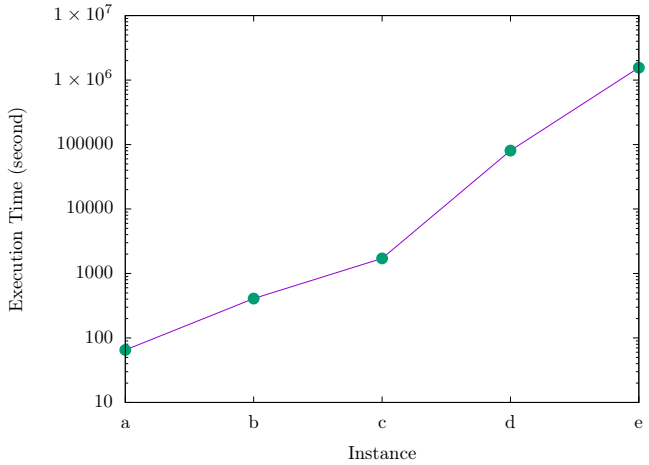


FIGURE 3.7: Computation time for various instances (from Luxembourg city) of PolySCIP. The following numbers are in "street nodes x buildings" format. (a) 63x47 (b) 222x307 (c) 366x584 (d) 727x876 (e) 1390x1063. For instance (a) and (b), the number of stations is four. Instance (c), (d) and (e), the number of stations is 10.

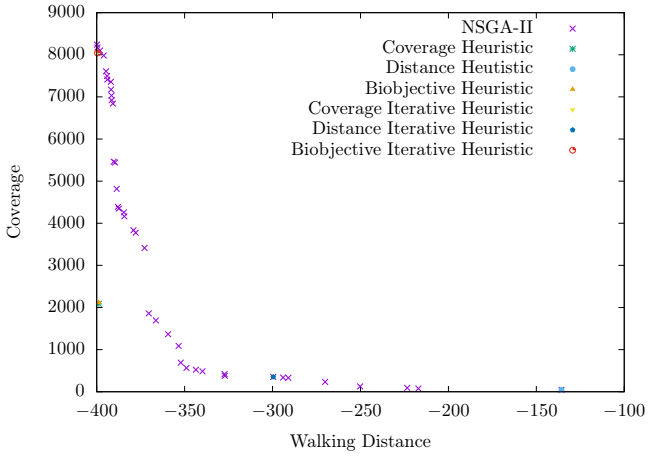


FIGURE 3.8: NSGA-II's Pareto front and heuristic algorithms' solutions for LU2. The higher the value on x axis (moving toward the right), the better the distance objective is. The higher the value on y axis, the better the user coverage objective is.

3.3.7.3 MU1: City-wide Scenario

Only NSGA-II and heuristic algorithms could obtain solutions in reasonable time due to the size of the MU1 scenario. Their respective results are presented in Figure 3.9. The obtained results are consistent with LU1 and

TABLE 3.7: Execution time for NSGA-II and heuristic algorithms on MU1. The measured time depicts an execution time each algorithm takes to locate 100 stations.

Algorithm	Execution time
NSGA-II	26 minutes
Coverage Heuristic	7 minutes
Distance Heuristic	7 minutes
Bi-objective Heuristic	7 minutes
Coverage Iterative Heuristic	17 hours
Distance Iterative Heuristic	17 hours
Bi-objective Iterative Heuristic	17 hours

LU2 scenarios. Table 3.7 presents execution time of all algorithms. NSGA-II achieves higher user coverage and shorter walking distance than the iterative approaches. The improvement shows another advantage of NSGA-II which is to absorb solutions from other algorithms and use them as seeds. Figure 3.10 shows NSGA-II fleet placement solution which maximises the number of covered users. Red pins mark locations of the carsharing stations, while green polygons show designated parking areas in the city of Munich.

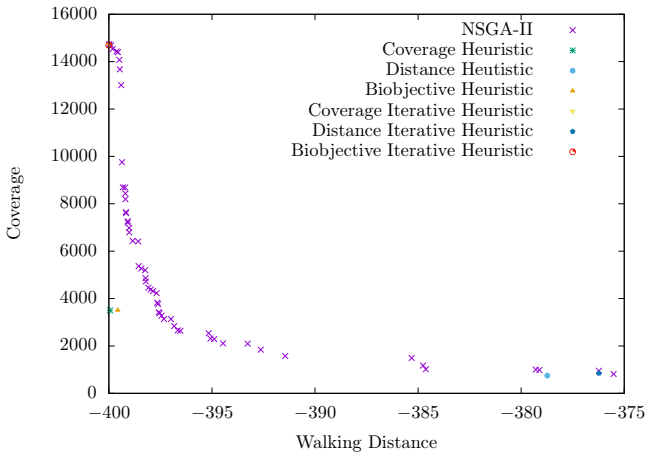


FIGURE 3.9: NSGA-II Pareto front and solutions of heuristic algorithms in MU1 scenario.

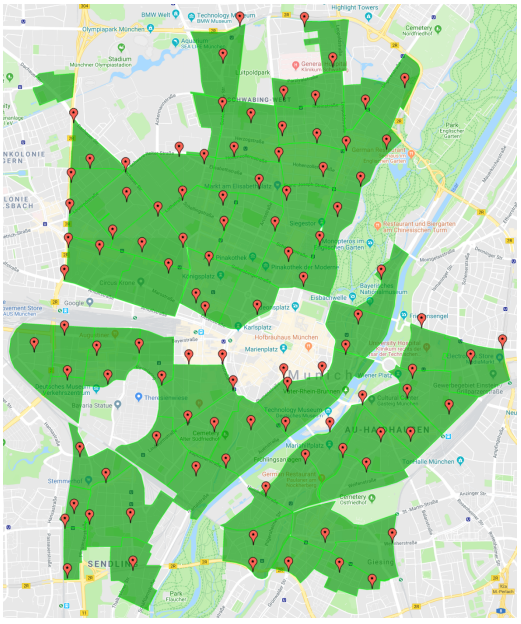


FIGURE 3.10: NSGA-II fleet placement solution which maximises user coverage in the city of Munich. Red pins are locations of the carsharing station. Green zones indicate the inner area of Munich City

Next, all heuristic algorithms and NSGA-II are executed with 72 stations in MU1 scenario to compare with the manual allocation. Figure 3.11 shows that the manual allocation is inferior to some heuristic algorithms, the iterative coverage and biobjective version in particular, and NSGA-II. The difference in user coverage is up to 50% (manual allocation being on the lower end), while the walking distance is the same. The results also further stress the benefit of Pareto front in decision making since it offers more options to choose from comparing to the heuristic algorithms.

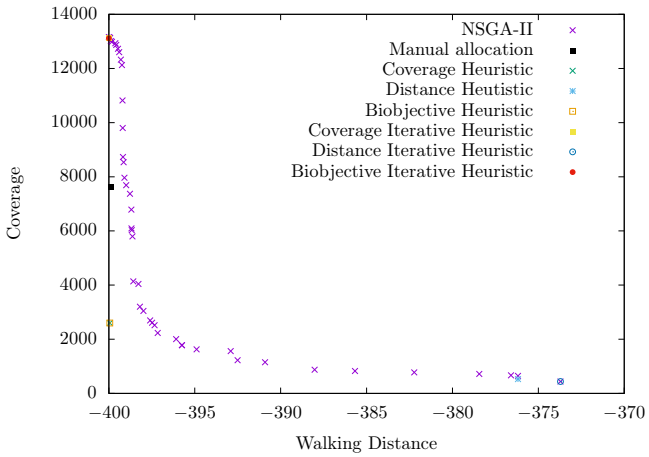


FIGURE 3.11: NSGA-II approximated Pareto front and solutions of heuristic algorithms in MU1 scenario compared to the manual allocation (72 stations).

3.3.8 Discussion

Efficiency and performance: When optimality is a key, PolySCIP must be used. However, its high execution time and memory requirements limit its applicability to only very small scenarios. For example, in the scenario with 1,390 street nodes and 1,063 buildings, PolySCIP was unable find solution in an acceptable time, and this is not even close to the size of a typical large city.

Simple heuristic algorithms, on the other hand, have the lowest execution time of all, but their results are insufficient for practical applications due to their low user coverage. The underperformance of Simple heuristic algorithms is alleviated in the iterative version, though it comes with an additional computation cost. Despite the low execution time for a small instance, it becomes an issue in a larger instance. In MU1 scenario, the simulations took 17 hours to locate 100 carsharing stations. Increasing the number of stations or increasing the size of the analysed area will increase execution time linearly and can make it impractical.

NSGA-II's main advantages are the approximated Pareto front, the ability to cope with the size of problem instance, and the ability to improve existing solutions even further (if possible). The last ability is shown through the utilisation of heuristic algorithms solutions in LU2 and MU1 scenarios. NSGA-II is 30 times faster than iterative algorithms and still able to produce other options without a need to rerun the algorithm and change weights (in bi-objective iterative algorithm). This also means if heuristic algorithms solutions are not available (due to execution time), solutions from previous runs of NSGA-II can also be used as seeds and be improved. These properties make NSGA-II an attractive choice in finding applicable fleet placement solutions.

Coverage vs. walking distance: After observing the coverage quality of distance-oriented algorithms, the walking distance effect is only marginal in supporting carsharing service. To elaborate further, if equal weights for both objectives are given in the iterative bi-objective heuristic algorithm. Ideally, the given solution should be at (-397, 3,000) in the approximated Pareto front in Figure 3.9. However, it is located at (-399, 14,700)

instead. There are also solutions where the walking distance is very low and so the user coverage because they place stations away from crowded areas and situate the stations near a few buildings, hence, claiming the low maximum walking distance and yield lower user coverage (see Figure 3.12). The decrease in walking distance (in those solutions) only translates to two to three minutes difference on foot.

FIGURE 3.12: A solution that yields a low global walking distance, but also yielded low user coverage.

This effect can be explained as the nature of the city. Users are clustered in a densely populated area. If a carsharing station is placed in such an environment, its coverage will be as far as it can reach which always leads to high maximum walking distance as shown in the results. A station can be shifted to lower the walking distance, but the user coverage is also likely to be lower in the process. On the other hand, the walking distance can be drastically low if carsharing stations are placed in an uninhabited area, but that would greatly hurt the user coverage objective and contradicts to the main purpose of carsharing service. Therefore, walking distance objective can be dismissed in carsharing station placement.

3.3.9 First Phase Summary: Algorithms and Optimization Objectives

A novel methodology for automating fleet placement in station-based round-trip carsharing is proposed. To solve this NP-hard problem, a set of heuristic, metaheuristic (NSGA-II) and PolySCIP (multi-criteria solver) algorithms are introduced and evaluated in three scenarios. Two optimization objectives, maximising the number of carsharing users and minimising maximum global walking-to-the-car distance, are chosen for optimization.

The experiment are performed on three different scenarios, each differs in size and purpose. Especially, in the third scenario, MU1 where the comparison of the obtained solutions from NSGA-II and manual allocation is presented. From the result, NSGA-II is shown to be superior to the manual allocation by a big margin in user coverage and with a yielded approximated Pareto front, there are a number of solutions for decision makers to choose from. Finally, the walking distance objective is found to be ineffective and negligible in implementation. This has been shown in all three scenarios leading to the conclusion that maximising the outreach of the stations is more important.

3.4 Second Phase: Refinement and Metaheuristic Algorithms Evaluation

The two objectives, user coverage and global walking distance are discussed in the previous section. The global walking distance is suggested to not be suitable for the fleet placement problem due to its detrimental effect.

The experiments also show that the exact methods and heuristic algorithms are not efficient when the input data is large. In this second phase, the goal is to refine the optimization objectives and perform an experiment on various metaheuristic algorithms.

For the second phase, there are two changes to the problem. The first change is the discard of the station notion. For this phase, the focus is shifted to locating the positions of the vehicle as it is more realistic from the carsharing company point of view. Hence, the name, fleet placement problem (FPP) is changed to vehicle placement problem (VPP). The second change is the objectives. Three objectives are proposed for vehicle placement problem (VPP), (1) Maximising user coverage, (2) Minimising the vehicle number, and (3) Maximising the public transportation coverage.

3.4.1 Optimization Model

Residential buildings $b_j \in B$ and public transport locations $t_k \in T$ are considered to be covered by $s_i \in S$ if the walking distance (d) is less than or equal to the set threshold. In this work, each vehicle placed on the streets can handle travel requests of all covered users. The optimization model can be described mathematically as follows.

3.4.1.1 Parameters

Unlike FPP, vehicle placement problem (VPP) discard the notion of a station and become a vehicle location. A vehicle is placed on a location on a street node in the street network $S = \{s_1, s_2, \ldots, s_i\}$. The maximum walking distance from a building to a station is d. As the assumption that users start the trip from their residences still stands, the number of users p_j is associated with the building b_j. The additional element in VPP is the public transportation t_k which has an importance factor of w_k. The coverage of each location s_i for buildings $B = \{b_1, b_2, \ldots, b_j\}$and public transportation $T = \{t_1, t_2, \ldots, t_k\}$ is defined in matrices E and G. All parameters in VPP are described in detail in this section.

- n: Number of street nodes.
- m: Number of buildings.
- q: Number of public transport locations (T).
- x: Minimum number of selected vehicle locations allowed
- y: Maximum number of selected vehicle locations allowed
- i: Index for street nodes (belongs to $\mathbb{Z}^+$).
- j: Index for buildings (belongs to $\mathbb{Z}^+$).
- k: Index for public transportation locations (belongs to $\mathbb{Z}^+$).
- d: Walking/coverage distance
- B: Building nodes, $B = \{b_1, b_2, \ldots, b_m\}$.
- P: Population of the city, $P = \{p_1, p_2, \ldots, p_m\}$, such that p_j is the population of building $b_j \in B$.
- T: Public transport of the city, $T = \{t_1, t_2, \ldots, t_p\}$.

- W: Public transport type weight, $W = \{w_1, w_2, \ldots, w_p\}$.
- $E_{n\times m}$: the user coverage matrix. Given a parking spot s_i and a building b_j, $E_{i,j} = 1$ iff $distance(s_i, b_j)$ is less than or equal to a fixed walking distance (d). In that case, the building b_j is covered by the vehicle location. When the condition is not verified, $E_{i,j} = 0$.
- $G_{n\times q}$: the public transport coverage matrix. Given a parking location s_i and a public transport t_k, $G_{i,k} = 1$ iff $distance(s_i, t_k)$ is less than or equal to a fixed walking distance (d). In that case, the public transport location t_k is covered by the vehicle location. When the condition is not verified, $G_{i,k} = 0$.

3.4.1.2 Outputs

The variables in VPP are the location on the street denoted by s'_i, the state of building denoted by e_j and the state of public transport denoted by g_k. It operates in binary, to indicate whether this location is selected or not. The location can be selected even if it does not cover any buildings or public transportations.

- e_j: represents that building b_j is/is not covered by selected locations ($b_j \in B$). If the building b_j is covered then $e_j = 1$ and $e_j = 0$ otherwise. A building is covered when (at least) there is an activated parking spot within a fixed walking distance.
- g_k: represents that public transport location t_k is/is not covered by a parking spot ($t_k \in T$). If the public transport location t_k is covered then $g_k = 1$, and $g_k = 0$ otherwise. A public transport location is covered when (at least) there is one selected location within a fixed walking distance.
- s'_i: represents the state of a vehicle locations. One location can only accommodate one vehicle. If s'_i is active (the location is selected) then $s'_i = 1$ and $s'_i = 0$ if the location is not selected.

3.4.1.3 Optimization Objectives and Description

With the introduced parameters, VPP is mathematically formulated with three objectives, (1) maximising the user coverage, (2) minimising the number of vehicles and (3) maximising the public transport location coverage, as follows.

$$
\begin{aligned}
&\text{Minimise} && \sum_{i=1}^{n} s'_i \\
&\text{Maximise} && \sum_{j=1}^{m} (e_j \times p_j) \\
&\text{Maximise} && \sum_{k=1}^{q} (g_k \times w_k) \\
&\text{Subject To} && e_j = E_{ij} \times s'_i && (1) \\
& && g_k = G_{i,j} \times s'_i && (2) \\
& && x \leq \sum_{i=1}^{n} s'_i \leq y && (3) \\
& && s_i, e_j, g_k \in \{0, 1\} && (4) \\
& && n, x, y, m, q \in \mathbb{Z}^+ && (5) \\
& && d \in \mathbb{Z}^{\geq 0} && (6)
\end{aligned}
$$

Constraint 1 indicates that a building b_j can be covered by multiple vehicle locations. Constraints 2 ensures that a building b_j is considered only once during the calculation. Constraints 2 are similar to constraints 1, but for public transport. Constraint 3 signifies the lowest and highest number of selected vehicle location allowed. Finally, Constraints 4 – 6 define the domains of the variables and parameters.

3.4.2 NP-Hardness Proof

In this section, the computational complexity of the vehicle placement problem (VPP) is analysed by introducing the decision counterpart of the VPP – VPP–D. VPP–D has the same parameters as VPP. The NP–hardness of VPP is proven by proving the NP–completeness of VPP–D. The question is to determine whether there exists a solution with n station(s) that covers a selection of buildings and public transport locations.

Theorem 1 The VPP is NP-Hard in the strong sense even if there is only one user in each building and all public transport locations have their weights equal to one.

Proof. A pseudo-polynomial transformation to the VPP–D from the strongly NP-Complete problem "Exact Cover By 3–Sets (X3C)" is introduced [88].

Exact Cover By 3-Sets, or X3C, can be defined as follows: Given a set U with cardinality of $|3L|$ and a collection $C = \{c_1, c_2, c_3, ..., c_L\}$ where C is a collection of **3-element subset** of U. The question for this problem is "Does C contain an exact cover for U, that is, a subcollection $C' \subseteq C$, such that every element of U occurs in exactly one member of C'?" [88]. In this case, if R is a solution of X3C, $|R| = L$.

Given an instance of X3C, the introduction of following instance of VPP–D is in order. Firstly, let all elements in B and T be of the same type and be members of the set U where U (VPP–D) = U (X3C). Then, let S be the direct transformation of C where $S' = \{s'_1, s'_2, s'_3, ..., s'_L\}$, such that $s'_l = c_l$, $l = 1, 2, 3, ..., L$. From the construction of U (VPP–D), E and G matrices are merged together and since this new merged matrix depicts the membership of elements in s_l where the assumption is to have an acceptable walking distance $d = 0$ to ensure that all buildings and public transportations are all covered regardless of walking distance in E and G. Then, $|s_l|$ is ssumed to be strictly equal to three and there are $n = L$ stations. In addition, $p_j = 1; j = 1, 2, 3, ..., m$ and $w_k = 1; k = 1, 2, 3, ..., q$ are assumed, thus $\sum_{j=1}^{m}(e_j \times p_j) + \sum_{k=1}^{q}(g_k \times w_k) = |U|$ where $|U| = 3L = 3n$. The threshold value for the VPP–D is then $x = y = n = L$.

Let R be a solution to X3C. A solution for VPP–D is constructed, in which the elements in set U are covered by n stations where $s'_l = c_l \in R$, such that $r_l = s'_l$, if $s'_l \in R$ and $r_l = \emptyset$ if $s'_l \notin R$. Since R is an exact cover of U (X3C), all the elements (buildings and public transport locations) in the set U (VPP–D) are covered and the number of stations in the corresponding solution is $n = |R|$.

Next, the existence of a solution O in VPP–D with $|O| \leq n$ is assumed. From this function, it implies that the number of stations with $o_l \neq \emptyset$ should not exceed n, otherwise, $|O| > n$ and the function will not hold. Another implication is that the number of selected stations should not be less than n, otherwise, some buildings and public transport locations will not be covered. Therefore, there are exactly n station(s) with $s'_l \neq \emptyset$. Since all buildings and public transport locations form the set U, the set of stations with $o_l \neq \emptyset$ represents a solution to X3C.

The transformation of X3C to VPP–D is presented, since X3C belongs to the class NP, VPP–D also belongs to the class NP. (follow lemma 2.1 in [88]). □

TABLE 3.8: Spearman correlation coefficient between each pair of objectives. Objective 1 is user coverage maximisation. Objective 2 is vehicle number minimisation. Objective 3 is public transport coverage maximisation.

	Correlation Coefficient	p-Value
Objective 1 - Objective 2	0.965	2.2e-16
Objective 2 - Objective 3	0.932	2.2e-16
Objective 1 - Objective 3	-0.829	1.98e-12

The approximability of the VPP is discussed. The special case in **Theorem 1** is restated where each building has only one user and the public transport locations' weights are one. The special case is equivalent to the "Set Cover Problem (SCP)".

The instance of SCP is defined as follows; given a universe U of L elements, a collection of subsets of U, $C = \{C_1, ..., C_L\}$, the question is whether there is a set covering (C) of size k such that $k \leq L$ that covers all elements of U [97].

SCP was proved to be polynomially non-approximable within the ratio $c \cdot \ln |C|$ [90], for some constant $c > 0$. Therefore, the following statement is proposed.

Statement 1. There exists no polynomial $(c \cdot \ln n)$-approximation algorithm for the VPP where n is the input size, unless $P = NP$.

3.4.2.1 Objective Correlation

There are three objectives in VPP. The first one is user coverage maximisation (Objective 1). The second is vehicle number minimisation (Objective 2). The last objective is public transport coverage maximisation (Objective 3). The Spearman rank correlation coefficient is utilised to determine the correlation between three objectives with the confidence interval of 95%. P-Value is used to decide whether accept or reject the null hypothesis as it reflects the probability of obtaining the observed results of a test, assuming that the null hypothesis is correct. Hence, by this definition and the confidence interval of 95%, if the observed p-value is less than 0.05, the null-hypothesis is rejected. Table 3.8 shows the correlation between objectives.

From Table 3.8, it can be observed that the vehicle number have positive correlation with both types of coverage. This means the higher number of vehicle, the higher the coverage is regardless of its type. It presents a need to balance between the number of vehicle and the goal coverage. On the other hand, the negative correlation is presented for the user and the public transport coverage objectives. It indicates that if the user coverage increases, the public transport will decrease as a consequence. This conflicting behaviour makes the problem more difficult since it is not possible to achieve the optimal in every objective and is a proof that this task requires an efficient searching algorithm to solve.

3.4.3 Methodology

The finding in the first phase shows that metaheuristic is more efficient than the exact methods and heuristic algorithms in solving the large instances of the fleet placement problem (FPP). The crux is the execution time for those algorithms comparing to metaheuristic algorithms which was also reported in [47]. In this second phase, the study of three state-of-the-art metaheuristic algorithms which were used in several variant of location problem, namely, SPEA-II, NSGA-II and NSGA-III, is conducted. All algorithms are described in section 2.2. This section explains the concept of our proposed solution encoding method. The other operators are the same

as ones in the first phase which are binary tournament (Pareto-dominance based), two point crossover, and uniform mutation.

Solution Encoding

In the second phase, two new objectives are introduced. One of them is the vehicle number minimisation. Due to this objective, the solution encoding for VPP needs to be able to accomodate the variable length aspect of the solution. The variable length encoding has been studying extensive for the past years and is still an open question as mentioned in [98].

The variable length encoding scheme can be represented in two forms. Both forms are elaborated using VPP. First, the length of the chromosome is different, meaning one solution can have three locations and another can have ten locations. This scheme is not often used since conventional operators cannot be applied. The first investigation dated back to 1989 by Goldberg et al. [99]. The report pointed out the importance of building blocks (useful pattern) in the chromosome and how it can help with the convergence. However, in another investigation by Van Veldhuizen et al., the identification of building blocks in multiobjective optimisation problem is difficult because different patterns and their combinations yields different effect [100]. Despite this difficulty, it is still used in several works, mainly the routing and clustering problem which still require complex solution repair mechanism [101, 102].

Second, the length of chromosome is fixed (e.g. length of 10, but a chromosome might contain only five usable locations). There are several existing methods such as "hidden gen" [103], "active/inactive gen" [104], "hierarchical chromosom" [105], "filtering mas" [106], "activation threshol" [107, 108], "activation mask" [104], "control gene" [105, 108] and "don't care symbo" [109].

To determine the encoding scheme in this work, an experiment using NSGA-III with three different encoding schemes is conducted:

1. Variable length chromosome
2. Sorted don't care symbol
3. Scrambled don't care symbol

The first scheme is the representative of the first form. While, the second and third scheme follow the "don't care symbol" method specifically to simplify the solution encoding in VPP since having two chromosomes as one solution does not encourage that. Each scheme is elaborated further in the followings.

Variable length chromosome

Chromosomes in this scheme encompass only active locations in VPP. The same integer-based encoding as in FPP case is used. However, the chromosome are varied in size as can be seen in Figure 3.13

Sorted don't care symbol

In this scheme, "-1" is deployed as a "don't care" symbol. The chromosome is also sorted in descending order meaning all the "-1"s are on the right side of the chromosome (see Figure 3.14).

89	70	69	4	1

Solution 1

75	55	30

Solution 2

FIGURE 3.13: Chromosome representation for variable length chromosomes.

75	32	10	7	2

Solution 1

39	22	5	-1	-1

Solution 2

FIGURE 3.14: Chromosome representation for sorted don't care chromosomes.

63	39	85	43	5

Solution 1

87	39	-1	78	-1

Solution 2

FIGURE 3.15: Chromosome representation for scramble don't care chromosomes.

Scrambled don't care symbol

Similar to the its sorted version, it employs "-" as the "don't care symbol". However, in this scheme, instead of sorting the chromosome, multiple "-1" are diffused into the locations as shown in Figure 3.15.

All schemes are subjected to the following experiment configurations in Table 3.9. In this experiment, NSGA-III is executed on VPP with three different encoding schemes. Each of them is terminated after 2000 generations. The population size is 92 and the number of locations is varied from 10 to 100. The process of population initialisation is random. For variable encoding scheme, messy one-point crossover is employed. On the other hand, for two fixed length encoding schemes, 2-point crossover is utilised. The mutation is the same as one used in FPP with the mutation rate as $\frac{1}{\#locations}$, so The "don't care" symbols are not considered in the mutation. Each encoding scheme is implemented using Python 3.6 and DEAP [110]. Each is executed for 30 times on 10 5000-node synthetic instances.

Each encoding scheme is evaluated with three indicators, IGD, spread and hypervolume. For IGD and spread, the approximated Pareto fronts are consolidated from all executions and use it as a referenced front for calculating both indicators. The normalisation to [0,1] is applied to the fitnesses for all indicators. the Kruskal-Wallis test [111] is employed for unpaired multiple non-parametric test and the Wilcoxon test [112] is for pairwise non-parametric comparison. Both methods do not assume the normal distribution of the results from each algorithm. The confidence interval for all experiments is 95%. P-Value is used to decide whether accept or reject the null hypothesis as it reflects the probability of obtaining the observed results of a test, assuming that

TABLE 3.9: Parameter configuration for each encoding scheme.

	Variable	Sort	Scramble
Number of generations	2000		
Population size	92		
Number of locations	10-100		
Crossover operator	Messy one-point crossover	2-point crossover	
Crossover rate	0.9		
Mutation operator	Uniform		
Mutation rate	$\frac{1}{\#locations}$		

TABLE 3.10: IGD for each encoding scheme.

Instance	IGD ($\times 10^{-2}$)			
	Variable	Sort	Scramble	p-Value
5000_1	0.318 (0.031)	0.305 (0.031)	0.404 (0.032)	4.91e-13
5000_2	0.302 (0.025)	0.278 (0.022)	0.383 (0.030)	1.13e-14
5000_3	0.276 (0.025)	0.293 (0.029)	0.376 (0.029)	1.33e-13
5000_4	0.275 (0.027)	0.254 (0.025)	0.347 (0.036)	1.67e-13
5000_5	0.264 (0.026)	0.262 (0.025)	0.369 (0.048)	6.87e-13
5000_6	0.278 (0.024)	0.250 (0.026)	0.346 (0.037)	8.94E-14
5000_7	0.267 (0.027)	0.276 (0.027)	0.377 (0.037)	2.60e-13
5000_8	0.267 (0.023)	0.275 (0.029)	0.372 (0.033)	2.46e-13
5000_9	0.266 (0.020)	0.262 (0.026)	0.359 (0.037)	1.63e-13
5000_10	0.276 (0.022)	0.267 (0.028)	0.368 (0.028)	1.43e-13

TABLE 3.11: Spread for each encoding scheme.

Instance	Spread			
	Variable	Sort	Scramble	p-Value
5000_1	0.593 (0.051)	0.581 (0.033)	0.570 (0.042)	0.183
5000_2	0.600 (0.036)	0.584 (0.054)	0.575 (0.040)	0.033
5000_3	0.587 (0.037)	0.571 (0.044)	0.567 (0.041)	0.135
5000_4	0.596 (0.039)	0.581 (0.046)	0.558 (0.044)	0.004
5000_5	0.602 (0.045)	0.583 (0.035)	0.592 (0.033)	0.239
5000_6	0.587 (0.045)	0.589 (0.040)	0.571 (0.041)	0.173
5000_7	0.603 (0.039)	0.593 (0.039)	0.603 (0.045)	0.610
5000_8	0.597 (0.046)	0.586 (0.042)	0.592 (0.039)	0.554
5000_9	0.608 (0.037)	0.578 (0.034)	0.577 (0.045)	0.010
5000_10	0.604 (0.039)	0.592 (0.044)	0.596 (0.042)	0.387

the null hypothesis is correct. Hence, by this definition and the confidence interval of 95%, if the observed p-value is less than 0.05, the null-hypothesis is rejected where the null hypothesis is there is no difference in the average performance among the three strategies.

The red color in all presented tables indicates that the coloured value is significantly different than the rest. Two coloured values in the same row means, while the two are significantly different from another value, there is no statistical significance between the two.

Table 3.10 shows the IGD of each encoding scheme. It can be observed that the scramble scheme yield the worst convergence in all test instances. On the other hand, variable length encoding and sorted chromosome with don't care symbol are comparable. This means they both yield approximated Pareto fronts that are close to the referenced one.

Spread is an indicator to measure the diversity of the approximated front. In Table 3.11, it can be observed that all encoding schemes yields comparable spread in all instances. It can be conclude that each encoding have no significant difference in term of the diversity of the yielded solutions.

TABLE 3.12: Hypervolume for each encoding scheme.

Instance	Hypervolume			
	Variable	Sort	Scramble	p-Value
5000_1	0.309 (0.013)	0.323 (0.016)	0.284 (0.014)	5.93e-12
5000_2	0.307 (0.011)	0.331 (0.011)	0.283 (0.011)	4.73e-16
5000_3	0.324 (0.011)	0.328 (0.016)	0.294 (0.011)	2.43E-12
5000_4	0.303 (0.012)	0.327 (0.014)	0.285 (0.015)	3.56e-13
5000_5	0.325 (0.012)	0.341 (0.015)	0.293 (0.018)	8.25e-14
5000_6	0.308 (0.011)	0.332 (0.015)	0.290 (0.014)	2.31e-13
5000_7	0.339 (0.012)	0.347 (0.014)	0.305 (0.012)	1.96e-13
5000_8	0.313 (0.011)	0.328 (0.014)	0.279 (0.014)	1.78e-14
5000_9	0.313 (0.011)	0.328 (0.016)	0.284 (0.015)	1.16e-13
5000_10	0.312 (0.009)	0.327 (0.013)	0.285 (0.012)	2.77e-14

Table 3.12 shows that the sorted don't care symbol scheme yields the highest hypervolume than the rest. This results demonstrate the importance of sorting as well since the counterpart (scramble) yield significantly lower hypervolume which indicate worse convergence and diversity in the solutions.

Next, convergences of each encoding scheme are shown through three generational plots for IGD, spread and hypervolume, respectively. Figure 3.16 depicts the fast convergence of IGD for variable encoding scheme, but eventually, the sorting scheme can catch up to it at the end of 2000 generations. Although, the scramble also show the convergence, but it is worse compared to the other two.

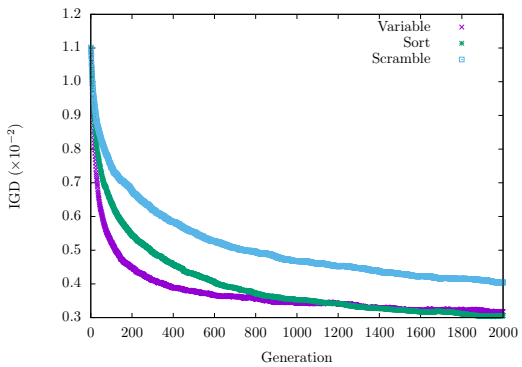


FIGURE 3.16: A generational plot of IGD for each encoding scheme on City_1 instance.

In Table 3.11, there is no statistical significance in spread for all encoding scheme. Figure 3.17 shows more details and it can be observed that there is a small difference in variable scheme compared to the two other schemes that employ "don't care" symbol.

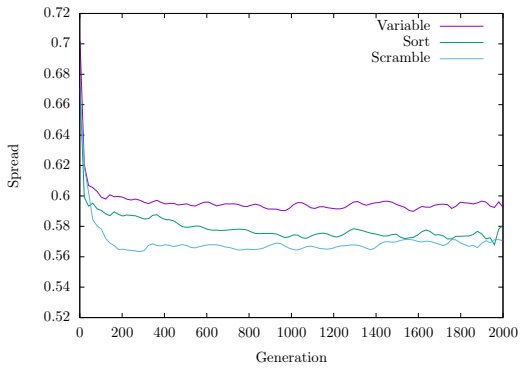


FIGURE 3.17: A generational plot of spread for each encoding scheme on City_1 instance.

Finally, Figures 3.18 present the generational plot of hypervolume for all schemes. It can be observed that the variable scheme can converge faster, but the sort scheme shows the ability to out grow it in the end. Meanwhile, the scramble scheme is the one that underperforms.

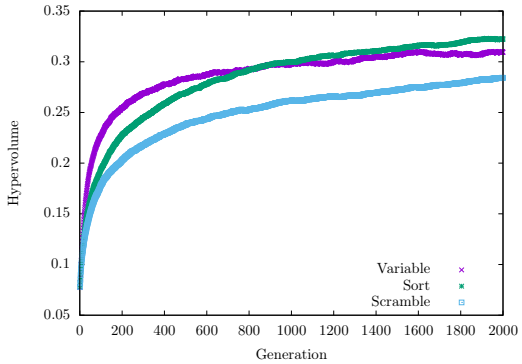


FIGURE 3.18: A generational plot of hypervolume for each encoding scheme on City_1 instance.

From all the obtained results in this experiment, the "sorted don't care symbol" is chosen since it has been show that this encoding scheme can lead to a convergence and at the same time, statistically better than the other two other schemes. Following this encoding scheme, a solution is initialised by choosing uniformly at random street nodes from all street nodes (in a problem instance) without replacement meaning each location in the solution is unique at the initialisation. The sizes of solution is also randomly assigned between the lower and upper boundary. If the solution has fewer locations than the upper bound, it is filled by -1 as mention in the solution encoding.

3.4.4 Experimental Setup

Three state-of-the-art metaheuristic algorithms, i.e. SPEA-II, NSGA-II and NSGA-III are implemented using Python 3.6 and DEAP [110]. NSGA-III, in particular, is implemented after NSGA-III module from [113], the rest are readily available in DEAP. The parameters of each algorithm are shown in Table 3.13. Each algorithm is run for 30 times on 10 50000-node synthetic and two real-world instances. The Kruskal-Wallis test [111] is employed for unpaired multiple non-parametric test and the Wilcoxon test [112] for pairwise non-parametric comparison. Both methods do not assume the normal distribution of the results from each algorithm. The confidence interval for all experiments is 95%. P-Value is used to decide whether accept or reject the null hypothesis as it reflects the probability of obtaining the observed results of a test, assuming that the null hypothesis is correct. Hence, by this definition and the confidence interval of 95%, if the observed p-value is less than 0.05, the null-hypothesis is rejected where the null hypothesis is there is no difference in the average performance among the three algorithms. All experiments were conducted on HPC platform of the University of Luxembourg [96].

The population size for all algorithms is set to 92, due to the fact that NSGA-III requires 92 individuals in a three-objective problem [72]. The archive size in SPEA-II is also set to 92. The number of vehicles is varied between 10 and 200 and the walking distance threshold is set to 500 meters. This means a building is considered to be covered by the carsharing service if the user can reach at least one shared fleet vehicle by walking towards it along the street for not more that 500 meters. The termination condition is 2,000 generations for the 10 synthetic instances and 3,000 generations for real-world instances due to the disparity in size. With the high number of generations, an extensive study on each algorithm can be conducted. The crossover method is 2-point crossover with a crossover rate of 0.9. Finally, the employed mutation method is a uniform mutation which is applicable to all genes in a chromosome.

TABLE 3.13: Parameters of SPEA-II, NSGA-II and NSGA-III.

	All algorithms
Number of generations	2,000, 3,000
Population size	92
Number of vehicles	10-200
Crossover operator	2-point crossover
Crossover rate	0.9
Mutation operator	uniform
Mutation rate	$\frac{1}{chrom\_length}$

TABLE 3.14: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
50000_1	0.209 (0.025)	0.162 (0.016)	0.168 (0.023)	1.13e-10
50000_2	0.283 (0.028)	0.184 (0.014)	0.211 (0.022)	3.45e-15
50000_3	0.296 (0.027)	0.142 (0.013)	0.209 (0.023)	2.20e-16
50000_4	0.262 (0.028)	0.163 (0.013)	0.177 (0.024)	1.15e-13
50000_5	0.246 (0.027)	0.206 (0.018)	0.187 (0.020)	2.71e-11
50000_6	0.223 (0.028)	0.173 (0.017)	0.171 (0.025)	2.67e-10
50000_7	0.246 (0.029)	0.184 (0.015)	0.167 (0.020)	4.17e-13
50000_8	0.263 (0.032)	0.181 (0.014)	0.184 (0.021)	2.79e-13
50000_9	0.219 (0.022)	0.184 (0.018)	0.167 (0.019)	1.23e-11
50000_10	0.293 (0.033)	0.178 (0.014)	0.204 (0.025)	2.76e-15
Munich	0.141 (0.011)	0.115 (0.009)	0.114 (0.025)	1.00e-10
Hamburg	0.176 (0.015)	0.114 (0.012)	0.124 (0.012)	2.57e-14

Three performance metrics in the experiments are also employed in this experiment, namely, inverted generational distance (IGD) [77], spread [60] and hypervolume [54]. All fitnesses are normalised from zero to one before the calculation. Due to the size of the problem, it is impossible to use exact methods to compute the true Pareto front, therefore, a reference Pareto front is constructed for each instance by merging all the Pareto fronts found by all the algorithms in all independent runs on that instance into a single front.

3.4.5 Experimental Results

This section presents the comparison of performance for SPEA-II, NSGA-II and NSGA-III on 12 problem instances (10 synthetic and two real-world) using IGD, spread and hypervolume (please refer to Appendix A for other instances results).

3.4.5.1 Statistical Analysis

The red colour in all presented tables indicates that the coloured value is significantly different than the rest. Two coloured values in the same row means, while the two are significantly different from another value, there is no statistical significance between the two.

Table 3.14 shows the average and standard deviation of IGD for each algorithm. From the table, NSGA-II and NSGA-III yields comparable IGDs across 12 instances. SPEA-II only produces significantly worse IGD on all test instances.

Regarding spread, SPEA-II and NSGA-III produce significantly lower, i.e. better, values than NSGA-II in all instances as shown in Table 3.15. In term of diversity, NSGA-II perform significantly worse than the other two algorithms.

TABLE 3.15: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
50000_1	0.576 (0.027)	0.626 (0.043)	0.577 (0.037)	1.68e-05
50000_2	0.580 (0.039)	0.647 (0.049)	0.598 (0.040)	4.06e-06
50000_3	0.589 (0.034)	0.630 (0.032)	0.602 (0.037)	2.10e-04
50000_4	0.585 (0.042)	0.637 (0.048)	0.585 (0.035)	1.16e-05
50000_5	0.577 (0.043)	0.637 (0.035)	0.584 (0.041)	7.98e-07
50000_6	0.573 (0.034)	0.630 (0.043)	0.578 (0.044)	3.21e-06
50000_7	0.578 (0.039)	0.627 (0.040)	0.592 (0.044)	2.68e-04
50000_8	0.584 (0.042)	0.643 (0.047)	0.600 (0.040)	2.44e-05
50000_9	0.578 (0.038)	0.639 (0.040)	0.595 (0.054)	2.55e-06
50000_10	0.580 (0.043)	0.646 (0.044)	0.610 (0.044)	2.14e-06
Munich	0.595 (0.040)	0.625 (0.046)	0.581 (0.047)	0.001
Hamburg	0.568 (0.041)	0.599 (0.037)	0.585 (0.034)	5.47e-03

TABLE 3.16: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
50000_1	0.328 (0.012)	0.351 (0.016)	0.346 (0.017)	2.84e-07
50000_2	0.323 (0.013)	0.384 (0.018)	0.359 (0.013)	3.17e-15
50000_3	0.321 (0.012)	0.390 (0.014)	0.363 (0.013)	2.20e-16
50000_4	0.316 (0.011)	0.366 (0.014)	0.353 (0.013)	2.48e-14
50000_5	0.332 (0.013)	0.367 (0.019)	0.359 (0.013)	3.32e-10
50000_6	0.328 (0.014)	0.362 (0.018)	0.355 (0.020)	2.14e-09
50000_7	0.325 (0.015)	0.367 (0.018)	0.367 (0.014)	8.26e-12
50000_8	0.306 (0.013)	0.353 (0.017)	0.343 (0.018)	9.93e-13
50000_9	0.325 (0.011)	0.347 (0.019)	0.350 (0.015)	1.40e-07
50000_10	0.318 (0.011)	0.372 (0.016)	0.349 (0.017)	3.10e-14
Munich	0.325 (0.006)	0.340 (0.006)	0.347 (0.015)	3.13e-10
Hamburg	0.271 (0.008)	0.300 (0.009)	0.300 (0.010)	3.25e-13

Hypervolume is compared in Table 3.16. SPEA-II yields the lowest hypervolume among the three test algorithms. On the other hand, NSGA-II yields similar hypervolume to NSGA-III on the test instances. Even if there are some instances that the differences are statistical significant, they are small differences.

3.4.5.2 Convergence Analysis

For convergence analysis, the average values of each metric are plotted against the number of generations. From this analysis, it is found that evaluated algorithms behave similarly in all studied instances. Therefore, an instance with the city of Munich is chosen as a representative in this section.

Figure 3.19 shows the generational plot of the IGD values of each algorithm over 3,000 generations. NSGA-II converges slightly faster than NSGA-III and greatly faster than SPEA-II. However, NSGA-III caught up in the end and yields comparable IGD.

The spread values of each algorithm fluctuate in each generation, which leads to a cloud of points, as shown in Figure 3.20. However, the trend can still be seen. SPEA-II maintains the same spread throughout all 3,000 generations, while NSGA-II spread trend is slightly decreasing over time. On the other hand, NSGA-III shows a steady decrease of spread further into the later generations.

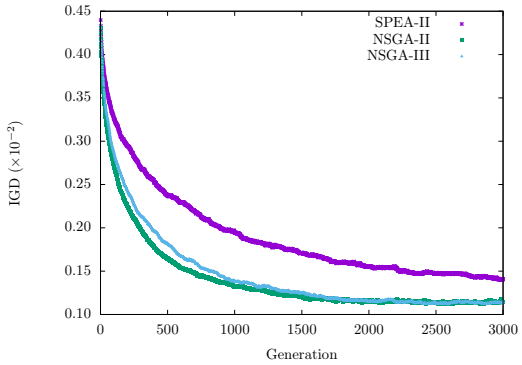


FIGURE 3.19: Generational plot of IGD of each algorithm on Munich instance.

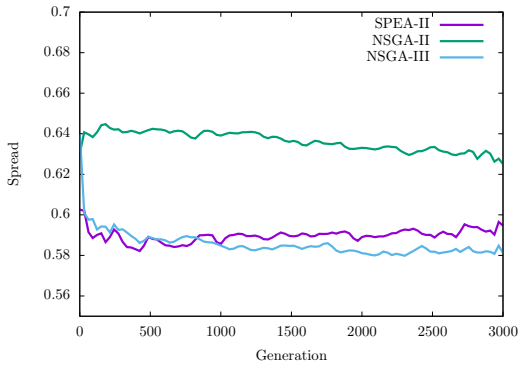


FIGURE 3.20: Generational plot of spread of each algorithm on Munich instance.

The hypervolume plot is shown in Figure 3.21. NSGA-II converges slightly faster than NSGA-III, but both converge to the comparable hypervolume at the 2000th generation. SPEA-II is worse than the other two in term of both end result and the convergence rate.

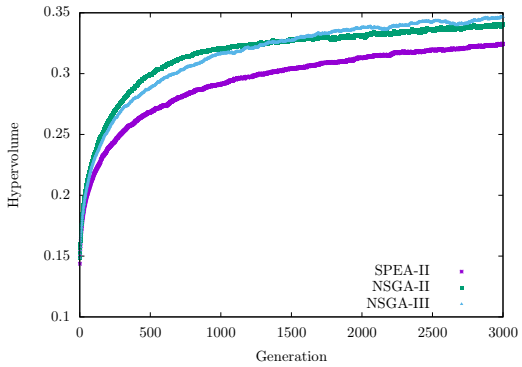


FIGURE 3.21: Generational plot of hypervolume of each algorithm on Munich instance.

By observing these plots, NSGA-II has a much faster convergence rate than the other two algorithms. If the execution time is a concern, NSGA-II can be terminated after the 500th generation and would still yield solutions closer to the reference Pareto front than the other two algorithms. However, if obtaining a diverse set of solutions is the main objective of the decision maker, i.e. the car sharing operator, NSGA-III is a better choice as it yields highly diverse solutions (low spread) and at the same time yields significantly lower IGD than SPEA-II and comparable hypervolume to NSGA-II. SPEA-II is completely outperformed by the other two algorithms in VPP. Hence, NSGA-III is selected to solve VPP and as a base algorithm into further development of the work.

3.4.6 Second Phase Summary: Observation on Algorithms

The corresponding Vehicle Placement Problem (VPP) has been modeled as a multi-objective problem with three objectives: (1) maximising the number of covered by the service citizens, (2) maximising proximity to public transport, and (3) minimising the number of vehicles in the operated fleet. The performance of three state-of-the-art algorithms is studied on 10 synthetic medium-size instances and two large-scale real-world instances. Experimental results demonstrated that NSGA-III is superior to NSGA-II and SPEA-II in term of convergence and diversity. Therefore, a hybrid algorithm is based on NSGA-III.

3.5 Third Phase: Hybridisation

In this phase, the hybridisation is touched. The aim of this phase is to utilise the problem improve the performance of the selected metaheuristic algorithm (NSGA-III) by introducing the problem specific knowledge which is not fully utilised in the state-of-the-art algorithms. According to [54], there are several strategies for creating a hybrid algorithm, however the author gave four examples which are;

1. Metaheuristics + Metaheuristics (complementary) or Heuristics
2. Metaheuristics + Exact methods
3. Metaheuristics + Constraint programmings
4. Metaheuristics + Machine learning

For this work, the first combination is pursued and it is classified as low-level teamwork hybrid (LTH) as defined in [54]. NGAP from the combination of NSGA-III (a base algorithm) with Pareto Local Search (PLS) [114] and our novel local search operator called Extensible Neighbourhood Search (ENS) for vehicle placement problem (VPP) is proposed. To the extent of my knowledge, there is no work presenting a combination of NSGA-III and PLS. In addition, the proposed local search operator takes full advantage of the street network and compute everything locally using only available local information. This section is dedicated to explain the idea and present the results of the hybrid algorithm comparing to the state-of-the-art NSGA-III.

3.5.1 Methodology

Most of the hybrid algorithms are single objective, for example, hybrid artificial bee colony [64, 65], hybrid simulated annealing [66], hybrid variable neighborhood search [67] and hybrid between GA and Variable Neighborhood Search (VNS) [115]. A few multi-objective versions have been proposed by Ishibuchi and Murata [69] and Jaszkiewicz [116]. These works use a weighted sum approach to combine and balance between the objectives. They require objectives to be normalised and several weight vectors are generated to find the approximated Pareto front. The approach from Tan et al. [117] randomly selects a single objective local search from a collection of algorithms and applies it on the whole population in each local search phase which makes it very difficult to determine the priority of local search approaches since it is highly dependent on problem instances.

Apart from the difficulties of determining the weights and the local search strategies, there is a troubling aspect of the solution neighbourhood structure. To elaborate on the concept of solution neighbourhood structure, let

us denote $N_k(x)$ where N_k is a set of solutions which can be considered as neighbours to solution x and has a size of $k, (k = 1, 2, 3, \ldots, k_{max})$. Neighbour solution be described as solutions that located closely to x in the solution space. If the neighbour solution concept is defined loosely, the neighbours of solution x can be all possible solutions. Otherwise, if it is very tight, i.e. $k = 1$, it can also hinder the exploitation process which can guide the algorithm toward the local or global optima. The aforementioned state-of-the-art algorithm suffered in this aspect greatly as there is no efficient method to determine the optimal size of the neighbourhood. This aspect has been studied extensively in [59]. The authors compared the neighbourhood strategies and reported that the best strategy to handle a large problem is to keep the size of the neighbourhood small and not to explore the whole neighbourhood, i.e. the whole solution space.

To sum it up, there are three difficulties presented in the previous work. They are (1) determining the suitable weight vectors (2) determining the right local search algorithm and (3) determining the set of neighbourhood. To overcome these difficulties, a dominance-based local search called Pareto Local Search (PLS) is proposed. With the dominance-based approach, the need of weight vectors can be eliminated. As for the local search operator, Extensible Neighbourhood Search (ENS) is then proposed.. The proposed algorithm takes the full advantage of graph structure and help intensifying the exploitation process to guide the solution toward the local optima. The idea of ENS is based on three facts reported in [118] and they are;

1. A local optimum with regard to one neighbourhood structure (N_k) is not always a local optimum in another neighbourhood.
2. A global optima is a local optimum with regard to all neighbourhoods.
3. In many problems, local optima with regard to one or several neighbourhoods are relatively close to each other.

The last observation which is the most important one, implies that local optimum often provides or shares some information with the global one. For instance, the local optima may share the same values of variables with the global one. However, until the global optimum is found, It is not known which one to look for, hence a structured study and search in various neighbourhoods must be performed and this leads us to the proposal of ENS. In this section, the building blocks of our hybrid algorithms which as Pareto Local Search (PLS) and Extensible Neighbourhood Search (ENS) are explained. As for encoding scheme and genetic operators, the same method as in the experiment in the second phase is exployed.

3.5.1.1 Pareto Local Search (PLS)

Pareto Local Search (PLS) was proposed by Paquette et al. in 2004 [114]. PLS can be considered as a general algorithm for local search. It operates on the concept of archive which is used to screen out the dominated solutions from the set of final solutions. However, it is important to note that PLS still needs local search operators as shown in [114] where the authors used 2-opt, 2h-opt and 3-opt local search algorithms for traveling salesman problem (TSP). Algorithm 6 shows the general process of PLS.

In essence, the process of PLS starts from initialising an empty archive for holding all non-dominated solutions. Then the initial solutions are generated and updated to the archive. During this process, the dominated solutions are discarded. After that, the neighbours of each solution in the archive are evaluated and added to the archive only if they are not dominated by any other solutions in the archive. This process continues until all solutions in the archive are visited. The size of the archive is also unlimited.

Algorithm 6 Pareto Local Search (PLS)

```
Archive ← ∅
Solutions ← GenerateInitialSolution ()
Archive ← UpdateArchive (Solutions)
while all solutions in Archive are not visited do
    for s′ in neighbour(s) do
        Evaluate (s′)
        if s′ is not dominated by r in Archive then
            Archive ← UpdateArchive (s′)
        end
    end
    s ← visited
    s ← s + 1
end
Return Archive
```

3.5.1.2 Extensible Neighbourhood Search (ENS)

ENS introduces the concept of neighbourhood stemmed from graph theory. The neighbourhood in a graph and the neighbourhood in the solution space mentioned earlier in this section are not the same. The neighbourhood in a graph is a set of nodes that can be reached from the chosen node. An example is shown in Figure 3.22. 1-hop neighbours are immediate neighbours of the chosen node (node 1), i.e. node 2, 3, 4, 5. While 2-hop neighbours include all 1-hop neighbours and those that are immediate neighbours of 1-hop neighbours. In this case, the 2-hop neighbours are all the nodes presented in Figure 3.22.

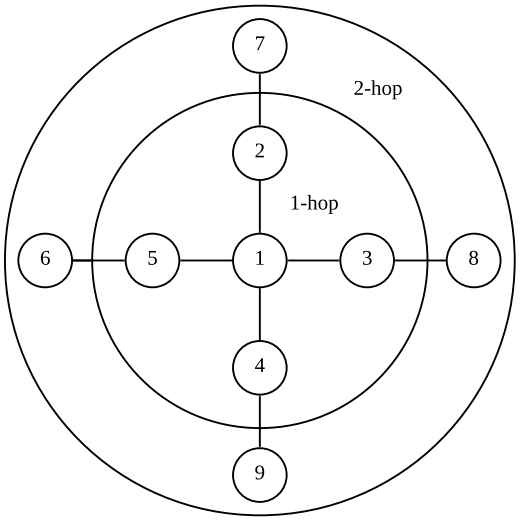


FIGURE 3.22: Neighbourhood in a graph.

In this work, ENS is designed with three problem-specific functions, Add, Cut and Improvement using Pareto-dominance as a criterion and specifically, when a neighbour is non-dominated, it means this neighbour dominates all others neighbours and current node in the neighbourhood. The algorithm is also suitable with the encoding scheme where there is no order in the encoding which local search algorithms such as Lin—Kernighan heuristic [119], swap move, or insertion move [120] cannot be applied since the change of gene order in the solution does not incur any changes. Although, these three functions are problem-specific, ENS concept is still highly versatile to accommodate any problems that are graph-based. Due to the concept of graph theory, node and location, are used interchangeably in this context.

Algorithm 7 Extensible Neighbourhood Search (ENS)

```
Data: Solution, Street Graph (G), addHop, cutHop
while termination condition is not met do
    % Add
    for location ∈ Solution do
        neighbours ← (Neighbourhood (G, location, addHop) \ Neighbourhood (G, location, cutHop))
        addLocation ← location for n in neighbours do
            if n dominates addLocation then
                addLocation ← n
            end
        end
        Add (addLocation)
    end
    % Cut
    for location ∈ Solution do
        if location == -1 then
            continue
        end
        neighbours ← Neighbourhood (G, location, cutHop)
        for n in neighbours do
            if n dominate location then
                location ← -1
                break
            end
        end
    end
    % Improvement
    for location ∈ Solution do
        if location == -1 then
            continue
        end
        neighbours ← Neighbourhood (G, location, cutHop)
        for n in neighbours do
            if n dominate location then
                location ← n
            end
        end
    end
end
```

Termination conditions In [59], the authors experiment on three termination strategies for the local search algorithms. They are;

1. Strategy 1 (S1): Terminate when the first non-dominated solution is found
2. Strategy 2 (S2): Terminate when the improvement limit is reached (e.g. after 200 rounds)
3. Strategy 3 (S3): Terminate when all solutions are explored

It is intractable in our VPP problem to explore all possible solutions since the problem is NP-hard. Therefore, the experiments on S1 and S2 are conducted. However, ENS is a deterministic algorithm by nature on any static graphs (since there is no structure change, ENS yields the same result), so the modification is made to the two strategies to make them work with ENS by terminating the local search operation once the solution cannot be improved any further regardless of the termination conditions. For example, for S1, if the solution cannot be improved, the search is terminated. As for the S2, even if the improvement limit has not been reached, but

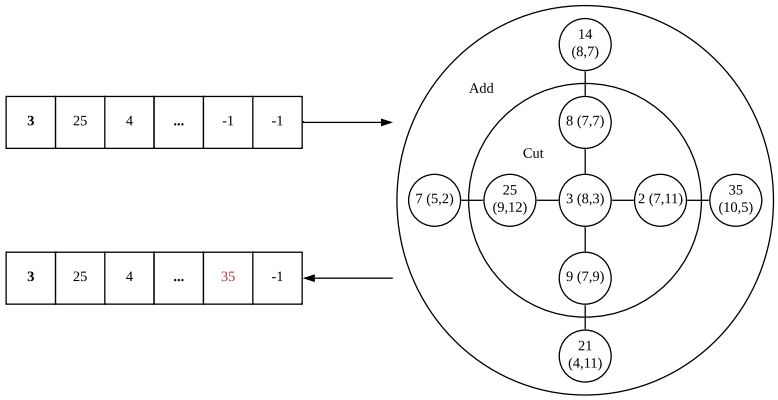


FIGURE 3.23: Adding process in ENS.

the solution cannot be improved any further, the search is also terminated. By doing this, an execution time can be reduced and let the metaheuristic algorithm does more of the exploration and let the ENS enhance the exploitation.

Add

In ENS, the purpose of add process is to introduce new non-dominating solutions into the pool through the addition of locations. Adding starts with finding a set of potential neighbours for consideration. This is done by two adjustable parameters, *addHop* and *cutHop* as mentioned in Algorithm 7. The former indicates the reach from the current node and the latter specifies that lower bound. The final neighbourhood contains only nodes that are not presented in the *cutHop* neighbourhood. For example, in Figure 3.23, the *addHop* is two and *cutHop* is one. The considered neighbourhood for adding process contains only node 7, 14, 21, and 35. The first number in the parentheses is the user coverage of this location and the second one is the public transportation coverage.

The next step is to decide whether to add new location to the solution. This is done by comparing all the nodes in the neighbourhood and current location. The non-dominated location is then added to the solution as long as there is a free space (denote as -1) in the solution. If there are more than one non-dominated locations, the function randomly selects one. Only original locations in the solution are considered for adding process, meaning, if they are 10 locations originally, the highest number of location is 20 after the process.

The whole process is shown in Figure 3.23 where node 3 is considered as the current location. Then, all the neighbour nodes and current location are compared. There are two non-dominated locations in this case and finally, node 35 is randomly selected over node 14. Since there is a "-1" in the solution, node 35 is added by replacing the "-1" gene in the solution.

Cut

Cutting location is the second process in ENS. The purpose of cutting is to streamline the solution by removing the redundancy. This process first define a set of neighbourhood for cutting on the chosen node. The size of neighbourhood is adjustable through the parameter *cutHop*. In Figure 3.24, The neighbourhood consists of node 2, 8, 9, and 25.

Once the neighbourhood is identified, the comparison for cutting starts. The current location is compared against all location in the neighbourhood. If the current location is dominated (in term of user and public transportation coverage) by at least one location in the neighbourhood presented in the solution, it is removed

from the solution and is replaced by "-1" (which will be sorted later). For instance, in the case of Figure 3.24, location 3 is dominated by location 25. Therefor, it is removed and replaced by "-1".

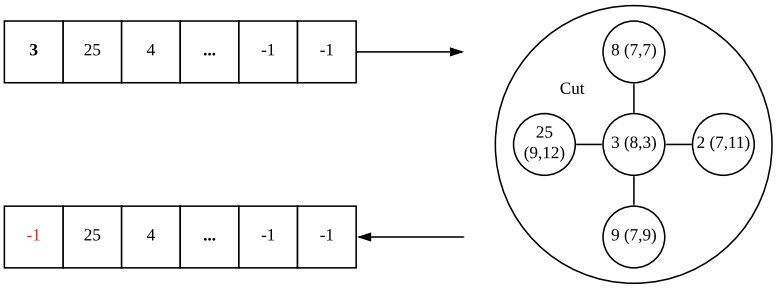


FIGURE 3.24: Cutting process in ENS.

Improvement

ENS focuses mainly on local knowledge utilisation. This holds true for all of its processes. In improvement process, the current location is relocated to another location in the neighbourhood that dominates the current location, e.g., have a better or equal user coverage and better public transportation coverage.

In Figure 3.25, the neighbourhood contains node 5, 8, 18, and 56. In this neighbourhood, node 18 dominates the current location (node 3). Therefore, the current location is relocated to node 18. Through the improvement process, the solution is being guided toward the local optimum systematically and at the same time create building blocks for reaching the global optima.

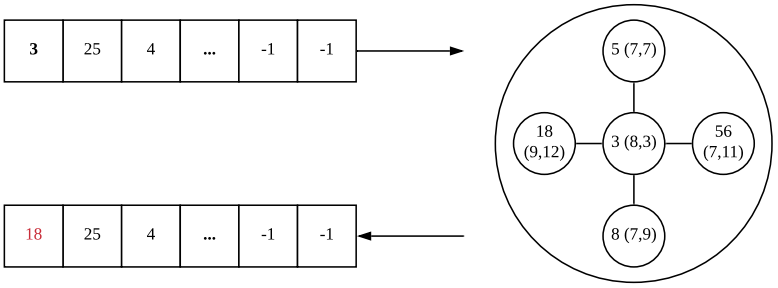


FIGURE 3.25: Improvement process in ENS.

3.5.1.3 Hybrid Algorithm: NGAP

The foundations of the proposed NGAP are already explained. Algorithm 8 shows the process of the proposed hybrid algorithm. PLS with ENS as the local search operation is executed after predefined generations. The local search improves the solutions produced from the crossover and mutation. Then all solutions are sorted by non-dominated sort for the selection process of NSGA-III afterward.

3.5.2 Hop Parameter Analysis

Extensible Neighbourhood Search (ENS) is the problem-specific local search operator proposed in this research. The main idea is to rely on the neighbourhood (in graph theory) of the street network and execute the local search based on Pareto-dominance concept. The operator take two important parameters which are *addHop* and *cutHop*. The larger the value of hop, the larger the size the search neighbourhood is. This parameter is very important as it affect the quality of the yielded solutions directly (whether for adding or cutting process

Algorithm 8 Hybrid Metaheuristic Algorithm Framework (NGAP)

```
Population ← InitialisePopulation (size)
while termination criteria are False do
    Parents ← TournamentSelection (population)
    Offsprings ← recombination+mutation(Parents)
    Archive ← ∅
    if Local Search criteria are met then
        Archive ← PLS+ENS (Offsprings)
    end
    else
        Evaluate (Offsprings)
    end
    F_1, F_2, ... ← NondominatedSort (population,Offsprings,Archive)
    while |NextGeneration| ≤ size do
        NextGeneration = NextGeneration ∪ F_i
        i = i + 1
    end
    if |NextGeneration| == size then
        Population ← NextGeneration
    end
    else
        F_l ← F_i
        NextGeneration ← (F_1 ∪ F_2 ∪ ... ∪ F_{l-1})
        ReferencePoints ← createHyperplane(NextGeneration)
        Associate(NextGeneration, ReferencePoints)
        K = size − (|tmpPopulation|)
        Population ← Niching (K, ReferencePoints, F_l, NextGeneration)
    end
end
Return Population
```

in ENS). The operator is then integrated into NGAP which is the hybrid algorithm proposed in this research as well. In this section, the effect of hop is studied to find the efficient value to be used in this work.

3.5.2.1 Experimental Setup

The effect of hops in ENS is tested with NGAP that employs strategy 2 (execute local search until the solution cannot be improved any further). There is also a condition that the parameter *addHop* cannot be smaller than *cutHop*, otherwise, the search advantage is diminished. Although, with this condition, the search space of parameters can still be very large. As it is not in our best interest to perform the surface analysis in this work, 15 combinations of *addHop* and *cutHop* are studied as shown in Table 3.17.

The study is conducted on 5000-node instances only (10 instances) and the termination criteria is set to be 5,400 seconds. The interval of local search is every 50 generations and the improvement limit for ENS is set 200 iterations. Each combination is executed for 30 times on each instance.

The performance indicators are inverted generational distance (IGD), spread and hypervolume. Kruskal-Wallis test is employed for unpaired multiple non-parametric test and Wilcoxon test is for the pairwise non-parametric comparison. The confidence interval for both tests is 95%. P-Value is used to decide whether accept or reject the null hypothesis as it reflects the probability of obtaining the observed results of a test, assuming that the null hypothesis is correct. Hence, by this definition and the confidence interval of 95%, if the observed p-value is less than 0.05, the null-hypothesis is rejected where the null hypothesis is there is no difference in the average performance among all observed parameters.

TABLE 3.17: *addHop* and *cutHop* parameters

addHop	cutHop
3	1
4	2
5	3
6	3
7	3
5	4
6	4
7	4
6	5
7	5
8	6
9	7
10	8
11	9
12	10

TABLE 3.18: Average IGD for each set of *addHop* and *cutHop* (part 1)

Instances	IGD ($\times 10^{-2}$)					
	3, 1	4, 2	5, 3	6, 3	7, 3	p-value
5000_1	0.236 (0.026)	0.217 (0.017)	0.235 (0.022)	0.228 (0.023)	0.227 (0.017)	2.56e-14
5000_2	0.220 (0.019)	0.229 (0.027)	0.220 (0.022)	0.226 (0.025)	0.228 (0.021)	4.82e-14
5000_3	0.194 (0.017)	0.187 (0.015)	0.186 (0.011)	0.193 (0.018)	0.191 (0.022)	1.22e-14
5000_4	0.193 (0.017)	0.195 (0.013)	0.201 (0.018)	0.196 (0.018)	0.193 (0.018)	5.67e-14
5000_5	0.200 (0.025)	0.193 (0.023)	0.212 (0.025)	0.209 (0.025)	0.211 (0.023)	7.84e-15
5000_6	0.215 (0.023)	0.206 (0.019)	0.202 (0.023)	0.211 (0.028)	0.212 (0.023)	9.57e-14
5000_7	0.197 (0.020)	0.193 (0.014)	0.197 (0.020)	0.190 (0.018)	0.197 (0.023)	3.62e-10
5000_8	0.216 (0.023)	0.212 (0.027)	0.210 (0.019)	0.216 (0.029)	0.204 (0.017)	2.09e-15
5000_9	0.205 (0.022)	0.207 (0.016)	0.202 (0.013)	0.206 (0.024)	0.209 (0.015)	1.06e-13
5000_10	0.208 (0.027)	0.206 (0.024)	0.219 (0.028)	0.207 (0.026)	0.217 (0.028)	4.57e-12

TABLE 3.19: Average IGD for each set of *addHop* and *cutHop* (part 2)

Instances	IGD ($\times 10^{-2}$)					
	5, 4	6, 4	7, 4	6, 5	7, 5	p-value
5000_1	0.221 (0.018)	0.161 (0.026)	0.228 (0.027)	0.223 (0.016)	0.223 (0.016)	2.56e-14
5000_2	0.230 (0.028)	0.159 (0.022)	0.228 (0.027)	0.231 (0.028)	0.231 (0.028)	4.82e-14
5000_3	0.194 (0.016)	0.134 (0.016)	0.192 (0.016)	0.189 (0.017)	0.189 (0.017)	1.22e-14
5000_4	0.194 (0.017)	0.150 (0.017)	0.202 (0.019)	0.206 (0.015)	0.206 (0.015)	5.67e-14
5000_5	0.202 (0.026)	0.144 (0.020)	0.210 (0.019)	0.202 (0.019)	0.202 (0.019)	7.84e-15
5000_6	0.207 (0.024)	0.143 (0.020)	0.210 (0.022)	0.211 (0.029)	0.211 (0.029)	9.57e-14
5000_7	0.196 (0.016)	0.149 (0.021)	0.196 (0.022)	0.200 (0.019)	0.200 (0.019)	3.62e-10
5000_8	0.219 (0.028)	0.145 (0.020)	0.212 (0.012)	0.220 (0.021)	0.220 (0.021)	2.09e-15
5000_9	0.207 (0.020)	0.143 (0.019)	0.202 (0.016)	0.208 (0.019)	0.208 (0.019)	1.06e-13
5000_10	0.205 (0.026)	0.142 (0.022)	0.223 (0.036)	0.209 (0.023)	0.209 (0.023)	4.57e-12

3.5.2.2 Experimental Results

Due to the large amount of information, the tables are separated into three parts for each metric. The red colour marks the statistically significant values among the rest. Table 3.18, 3.19, 3.20 display the average IGD of the parameters combinations on each tested instance. It can be observed that only the combination of 6 (*addHop*) and 4 (*cutHop*) yield the lowest average IGD in all instance. The obtained IGD from other combinations are comparable to each other.

TABLE 3.20: Average IGD for each set of *addHop* and *cutHop* (part 3)

Instances	IGD ($\times 10^{-2}$)					
	8, 6	9, 7	10, 8	11, 9	12, 10	p-value
5000_1	0.234 (0.026)	0.230 (0.029)	0.230 (0.020)	0.243 (0.030)	0.238 (0.032)	2.56e-14
5000_2	0.217 (0.022)	0.224 (0.026)	0.219 (0.021)	0.219 (0.020)	0.232 (0.030)	4.82e-14
5000_3	0.187 (0.011)	0.192 (0.014)	0.186 (0.018)	0.184 (0.012)	0.185 (0.010)	1.22e-14
5000_4	0.198 (0.022)	0.196 (0.014)	0.195 (0.017)	0.194 (0.021)	0.193 (0.014)	5.67e-14
5000_5	0.206 (0.026)	0.207 (0.021)	0.209 (0.025)	0.215 (0.027)	0.211 (0.022)	7.84e-15
5000_6	0.203 (0.019)	0.207 (0.023)	0.210 (0.022)	0.197 (0.024)	0.213 (0.022)	9.57e-14
5000_7	0.198 (0.023)	0.199 (0.020)	0.200 (0.019)	0.202 (0.025)	0.195 (0.017)	3.62e-10
5000_8	0.215 (0.021)	0.202 (0.020)	0.213 (0.019)	0.205 (0.021)	0.209 (0.016)	2.09e-15
5000_9	0.204 (0.022)	0.203 (0.019)	0.204 (0.018)	0.202 (0.016)	0.201 (0.012)	1.06e-13
5000_10	0.213 (0.034)	0.206 (0.022)	0.212 (0.029)	0.211 (0.035)	0.216 (0.030)	4.57e-12

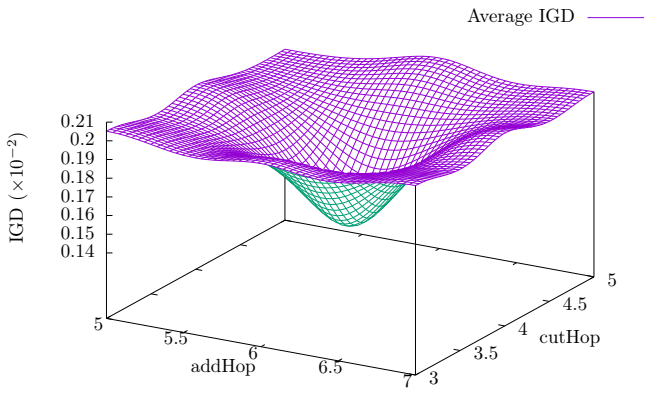


FIGURE 3.26: Surface plot of average IGD around the *addHop* and *cutHop* of (6,4).

TABLE 3.21: Average spread for each set of *addHop* and *cutHop* (part 1)

Instances	Spread					
	3, 1	4, 2	5, 3	6, 3	7, 3	p-value
5000_1	0.635 (0.049)	0.644 (0.052)	0.637 (0.051)	0.656 (0.054)	0.645 (0.043)	6.38e-06
5000_2	0.640 (0.041)	0.643 (0.045)	0.649 (0.056)	0.652 (0.048)	0.645 (0.051)	4.82e-14
5000_3	0.646 (0.054)	0.652 (0.047)	0.654 (0.049)	0.656 (0.048)	0.654 (0.044)	6.47e-07
5000_4	0.639 (0.053)	0.650 (0.044)	0.651 (0.047)	0.669 (0.048)	0.642 (0.043)	2.43e-07
5000_5	0.642 (0.050)	0.645 (0.038)	0.641 (0.047)	0.631 (0.049)	0.634 (0.047)	4.80e-02
5000_6	0.641 (0.041)	0.642 (0.041)	0.632 (0.052)	0.639 (0.049)	0.641 (0.051)	1.33e-02
5000_7	0.655 (0.052)	0.662 (0.054)	0.672 (0.054)	0.667 (0.051)	0.654 (0.046)	3.04e-05
5000_8	0.655 (0.052)	0.661 (0.039)	0.642 (0.042)	0.645 (0.039)	0.653 (0.044)	1.19e-01
5000_9	0.645 (0.048)	0.647 (0.050)	0.666 (0.041)	0.658 (0.048)	0.647 (0.048)	5.70e-05
5000_10	0.659 (0.045)	0.655 (0.043)	0.665 (0.049)	0.656 (0.047)	0.659 (0.039)	3.08e-06

The surface plot of IGD is illustrated in Figure 3.26. To simplify the plot, the mean of IGD is calculated and plotted from all tested instances. Regardless of the simplification, it can still be observed that the combinations around the point (6,4) yield higher IGDs (i.e. worse convergence). This assure the local minimality of the focused combination in IGD.

The results show that *addHop* and *cutHop* also affect the diversity. The obtained values show that the combination of 6 (*addHop*) and 4 (*cutHop*) fares best in term of diversity (see Table 3.21, 3.22, 3.23). The other combination again, yields similar spread across all instances.

Figure 3.27 display the surface of spread metric around the combination (6,4) which is shown to be the best in term of diversity in the Table 3.21, 3.22, 3.23. The surface plot depicts that the combination (6,4) is at least the local minimum in the studied sets of parameters.

TABLE 3.22: Average spread for each set of *addHop* and *cutHop* (part 2)

Instances	Spread					
	5, 4	6, 4	7, 4	6, 5	7, 5	p-value
5000_1	0.648 (0.035)	0.578 (0.039)	0.644 (0.043)	0.647 (0.042)	0.647 (0.042)	6.38e-06
5000_2	0.647 (0.047)	0.591 (0.038)	0.633 (0.053)	0.639 (0.051)	0.639 (0.051)	4.82e-14
5000_3	0.635 (0.036)	0.585 (0.034)	0.648 (0.047)	0.660 (0.040)	0.660 (0.040)	6.47e-07
5000_4	0.630 (0.047)	0.586 (0.043)	0.656 (0.045)	0.654 (0.056)	0.654 (0.056)	2.43e-07
5000_5	0.645 (0.048)	0.606 (0.036)	0.652 (0.044)	0.644 (0.046)	0.644 (0.046)	4.80e-02
5000_6	0.644 (0.051)	0.602 (0.050)	0.627 (0.052)	0.635 (0.045)	0.635 (0.045)	1.33e-02
5000_7	0.666 (0.044)	0.600 (0.039)	0.649 (0.042)	0.665 (0.049)	0.665 (0.049)	3.04e-05
5000_8	0.649 (0.060)	0.626 (0.058)	0.644 (0.041)	0.639 (0.046)	0.639 (0.046)	1.19e-01
5000_9	0.645 (0.056)	0.608 (0.034)	0.645 (0.047)	0.660 (0.045)	0.660 (0.045)	5.70e-05
5000_10	0.662 (0.041)	0.596 (0.040)	0.669 (0.059)	0.648 (0.045)	0.648 (0.045)	3.08e-06

TABLE 3.23: Average spread for each set of *addHop* and *cutHop* (part 3)

Instances	Spread					
	8, 6	9, 7	10, 8	11, 9	12, 10	p-value
5000_1	0.643 (0.050)	0.649 (0.053)	0.651 (0.052)	0.632 (0.053)	0.636 (0.047)	6.38e-06
5000_2	0.655 (0.046)	0.635 (0.041)	0.639 (0.048)	0.637 (0.052)	0.637 (0.039)	4.82e-14
5000_3	0.649 (0.048)	0.650 (0.055)	0.662 (0.049)	0.650 (0.059)	0.653 (0.046)	6.47e-07
5000_4	0.662 (0.054)	0.629 (0.041)	0.639 (0.042)	0.646 (0.041)	0.660 (0.047)	2.43e-07
5000_5	0.648 (0.039)	0.636 (0.050)	0.644 (0.048)	0.640 (0.036)	0.632 (0.037)	4.80e-02
5000_6	0.639 (0.040)	0.654 (0.044)	0.630 (0.046)	0.643 (0.043)	0.633 (0.035)	1.33e-02
5000_7	0.679 (0.054)	0.647 (0.057)	0.660 (0.064)	0.662 (0.052)	0.660 (0.047)	3.04e-05
5000_8	0.658 (0.045)	0.628 (0.059)	0.655 (0.049)	0.635 (0.046)	0.651 (0.049)	1.19e-01
5000_9	0.657 (0.047)	0.626 (0.041)	0.650 (0.061)	0.658 (0.043)	0.650 (0.046)	5.70e-05
5000_10	0.650 (0.044)	0.664 (0.047)	0.652 (0.056)	0.681 (0.049)	0.662 (0.045)	3.08e-06

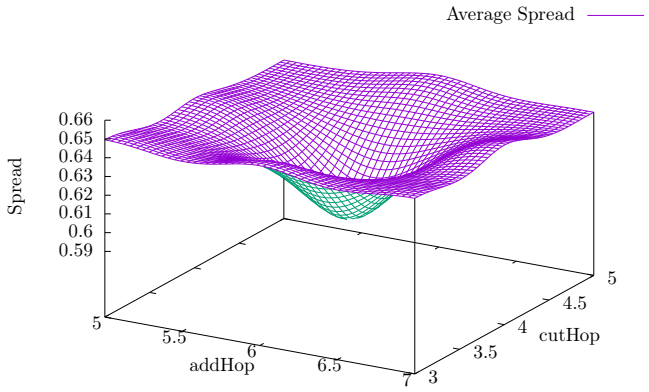


FIGURE 3.27: Surface plot of average spread around the *addHop* and *cutHop* of (6,4).

In term of hypervolume, the combination of 6 (*addHop*) and 4 (*cutHop*) is still the best combination. The performance is also consistence since it yields the highest hypervolume on all 10 instances. In hypervolume, there is no significant difference between other parameters as well.

There is a difference in the surface plot of hypervolume (see Figure 3.28 compared to the plots in Figure 3.26 and Figure 3.27 in that it depicts the local maximum. The hypervolume values of the combination (6,4) and those around it to display the peak of hypervolume are taken to study parameters.

3.5.3 NGAP Performance Analysis

The parameter analysis for ENS is shown in Section 3.5.2.1. The experiment shows that the best parameters (among the studied range) is *addHop* of 6 and *cutHop* of 4. The next step is to analyse the performance of

TABLE 3.24: Average hypervolume for each set of *addHop* and *cutHop* (part 1)

Instances	Hypervolume					
	3, 1	4, 2	5, 3	6, 3	7, 3	p-value
5000_1	0.360 (0.012)	0.369 (0.011)	0.360 (0.012)	0.366 (0.013)	0.365 (0.011)	3.12e-16
5000_2	0.346 (0.010)	0.341 (0.012)	0.347 (0.011)	0.344 (0.013)	0.342 (0.010)	4.82e-14
5000_3	0.361 (0.009)	0.366 (0.012)	0.367 (0.009)	0.360 (0.012)	0.365 (0.015)	3.10e-16
5000_4	0.358 (0.013)	0.358 (0.008)	0.353 (0.013)	0.355 (0.012)	0.359 (0.013)	2.21e-15
5000_5	0.365 (0.012)	0.370 (0.012)	0.360 (0.012)	0.360 (0.013)	0.359 (0.012)	1.78e-15
5000_6	0.344 (0.011)	0.346 (0.010)	0.350 (0.013)	0.345 (0.015)	0.344 (0.014)	4.47e-14
5000_7	0.393 (0.012)	0.395 (0.008)	0.395 (0.012)	0.397 (0.011)	0.392 (0.013)	1.89e-12
5000_8	0.340 (0.011)	0.343 (0.012)	0.344 (0.009)	0.341 (0.013)	0.347 (0.010)	3.43e-16
5000_9	0.357 (0.015)	0.357 (0.011)	0.361 (0.010)	0.357 (0.013)	0.353 (0.009)	1.48e-15
5000_10	0.357 (0.015)	0.357 (0.012)	0.352 (0.013)	0.356 (0.013)	0.353 (0.013)	9.06e-13

TABLE 3.25: Average hypervolume for each set of *addHop* and *cutHop* (part 2)

Instances	Hypervolume					
	5, 4	6, 4	7, 4	6, 5	7, 5	p-value
5000_1	0.367 (0.012)	0.416 (0.014)	0.366 (0.015)	0.365 (0.010)	0.365 (0.010)	3.12e-16
5000_2	0.342 (0.014)	0.391 (0.017)	0.343 (0.014)	0.341 (0.012)	0.341 (0.012)	4.82e-14
5000_3	0.361 (0.010)	0.426 (0.018)	0.363 (0.011)	0.364 (0.014)	0.364 (0.014)	3.10e-16
5000_4	0.357 (0.010)	0.393 (0.016)	0.352 (0.012)	0.347 (0.012)	0.347 (0.012)	2.21e-15
5000_5	0.364 (0.012)	0.409 (0.015)	0.359 (0.009)	0.364 (0.011)	0.364 (0.011)	1.78e-15
5000_6	0.346 (0.013)	0.390 (0.015)	0.343 (0.011)	0.345 (0.014)	0.345 (0.014)	4.47e-14
5000_7	0.392 (0.009)	0.433 (0.017)	0.392 (0.014)	0.391 (0.008)	0.391 (0.008)	1.89e-12
5000_8	0.340 (0.014)	0.394 (0.017)	0.342 (0.007)	0.338 (0.011)	0.338 (0.011)	3.43e-16
5000_9	0.356 (0.012)	0.404 (0.017)	0.359 (0.013)	0.356 (0.012)	0.356 (0.012)	1.48e-15
5000_10	0.357 (0.014)	0.402 (0.014)	0.353 (0.014)	0.357 (0.011)	0.357 (0.011)	9.06e-13

TABLE 3.26: Average hypervolume for each set of *addHop* and *cutHop* (part 3)

Instances	Hypervolume					
	8, 6	9, 7	10, 8	11, 9	12, 10	p-value
5000_1	0.359 (0.013)	0.363 (0.012)	0.363 (0.013)	0.358 (0.014)	0.360 (0.015)	3.12e-16
5000_2	0.348 (0.010)	0.346 (0.014)	0.348 (0.010)	0.348 (0.010)	0.343 (0.013)	4.82e-14
5000_3	0.367 (0.009)	0.365 (0.011)	0.368 (0.012)	0.370 (0.010)	0.368 (0.009)	3.10e-16
5000_4	0.355 (0.014)	0.355 (0.010)	0.355 (0.012)	0.357 (0.013)	0.357 (0.010)	2.21e-15
5000_5	0.364 (0.013)	0.361 (0.012)	0.360 (0.013)	0.360 (0.013)	0.359 (0.014)	1.78e-15
5000_6	0.346 (0.013)	0.345 (0.012)	0.343 (0.011)	0.351 (0.012)	0.342 (0.010)	4.47e-14
5000_7	0.394 (0.012)	0.393 (0.011)	0.391 (0.011)	0.391 (0.012)	0.395 (0.011)	1.89e-12
5000_8	0.341 (0.010)	0.349 (0.011)	0.343 (0.010)	0.347 (0.011)	0.345 (0.010)	3.43e-16
5000_9	0.361 (0.013)	0.360 (0.012)	0.360 (0.012)	0.362 (0.012)	0.360 (0.010)	1.48e-15
5000_10	0.354 (0.017)	0.357 (0.012)	0.354 (0.014)	0.357 (0.015)	0.355 (0.013)	9.06e-13

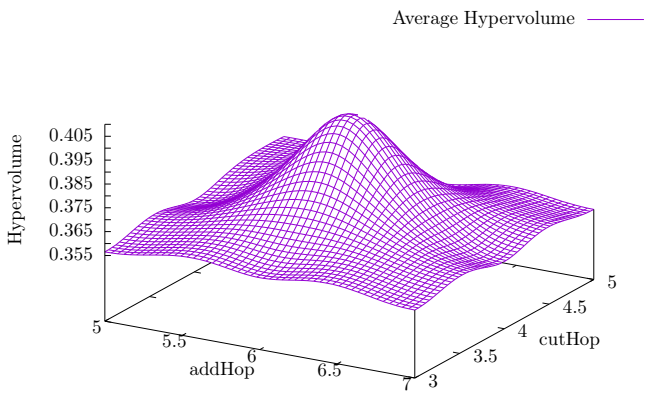


FIGURE 3.28: Surface plot of Average IGD around the *addHop* and *cutHop* of (6,4).

Table 3.27: Termination criteria for each instance.

Instances	Termination Time (seconds)
5000-25000 nodes	5400
25000-50000 nodes	7200
Dusseldorf, Hamburg, Munich, Berlin	11800

Table 3.28: Parameter configuration for experimental algorithms.

	NSGA-III	NGAP (S1, S2)
Population	92	
Number of vehicles	10-100 ($\leq$ 25000 nodes), 10-200 (>25000 nodes))	
Walking distance	500 meters	
Crossover method	2-point crossover	
Crossover rate	0.8	
Mutation method	uniform	
Mutation rate	$\frac{1}{\#location}$	
addHop	-	6
cutHop	-	4
Local search interval	-	every 50 generations
Improvement limit	-	200

NGAP itself compared to the original NSGA-III. In this analysis, real-world instances are included to observe how the proposed algorithm fare in the real world situation.

3.5.3.1 Experimental Setup

The experiment in this phase is to demonstrate the efficiency of the original NSGA-III and the proposed hybrid algorithm NGAP (with strategy 1 and 2). To make a fair comparison, the termination condition is set to be the execution time to ensure the fairness in the comparison since the hybrid algorithm require more time in some generations than the original NSGA-III. The time is varied depending on the size of the instance as shown in Table 3.27. The experiment is implemented using Python 3.6 and DEAP library. In general, both algorithms have similar setting except that in the hybrid algorithm, new parameters to facilitate the process are introduced as shown in Table 3.28. The *addHop* is set to six and *cutHop* is set to four. The local search algorithm is executed at every 50 generations and the improvement limit for ENS is set 200 iterations.

Each algorithm is run for 30 times on all 24 instances (10 5000-node, 10 50000-node, and four real-world instances). This is to demonstrate the performance of the proposed algorithm on several instance sizes. The Kruskal-Wallis test [111] is used for unpaired multiple non-parametric test and the Wilcoxon test [112] is for pairwise non-parametric comparison. Both methods do not assume the normal distribution of the results from each algorithm. The confidence interval for all experiments is 95%. P-Value is used to decide whether accept or reject the null hypothesis as it reflects the probability of obtaining the observed results of a test, assuming that the null hypothesis is correct. Hence, by this definition and the confidence interval of 95%, if the observed p-value is less than 0.05, the null-hypothesis is rejected where the null hypothesis is there is no difference in the average performance among all observed parameters. Due to the termination criteria being an execution time, the convergence analysis in this experiment is not presented due to the difficulty in calculating the exact average at any given times in the execution. All experiments were conducted on HPC platform of the University of Luxembourg [96].

TABLE 3.29: Average results of evaluated algorithm for the IGD metric.

Instance	IGD (x 10^{-2})			
	Original	Strategy 1	Strategy 2	p-Value
5000_1	0.192 (0.025)	0.161 (0.025)	0.161 (0.026)	1.01e-06
5000_2	0.182 (0.016)	0.167 (0.023)	0.159 (0.022)	3.98e-04
5000_3	0.173 (0.022)	0.142 (0.022)	0.134 (0.016)	2.63e-08
5000_4	0.169 (0.019)	0.148 (0.016)	0.150 (0.017)	3.63e-05
5000_5	0.185 (0.027)	0.152 (0.024)	0.144 (0.020)	2.33e-07
5000_6	0.164 (0.021)	0.148 (0.019)	0.143 (0.020)	6.82e-04
5000_7	0.172 (0.021)	0.152 (0.021)	0.149 (0.021)	1.76e-04
5000_8	0.185 (0.024)	0.150 (0.021)	0.145 (0.020)	1.48e-08
5000_9	0.169 (0.024)	0.151 (0.016)	0.143 (0.019)	5.66e-05
5000_10	0.174 (0.026)	0.149 (0.025)	0.142 (0.022)	1.91e-06
50000_1	0.350 (0.049)	0.172 (0.024)	0.164 (0.018)	3.88e-11
50000_2	0.373 (0.057)	0.158 (0.021)	0.143 (0.019)	1.37e-13
50000_3	0.316 (0.055)	0.134 (0.014)	0.122 (0.019)	2.82e-14
50000_4	0.409 (0.050)	0.159 (0.019)	0.144 (0.016)	4.52e-14
50000_5	0.435 (0.044)	0.168 (0.012)	0.161 (0.016)	2.27e-13
50000_6	0.407 (0.059)	0.163 (0.017)	0.153 (0.016)	5.32e-14
50000_7	0.404 (0.040)	0.163 (0.017	0.158 (0.019)	2.22e-13
50000_8	0.352 (0.029)	0.171 (0.021)	0.172 (0.020)	1.20e-13
50000_9	0.395 (0.039)	0.170 (0.018)	0.151 (0.017)	9.18e-15
50000_10	0.394 (0.073)	0.155 (0.020)	0.146 (0.019)	6.67e-14
Munich	0.185 (0.023)	0.156 (0.021)	0.142 (0.017)	2.50e-10
Hamburg	0.254 (0.028)	0.186 (0.015)	0.156 (0.013)	2.20e-16
Berlin	0.307 (0.027)	0.272 (0.022)	0.264 (0.020)	1.56e-08
Dusseldorf	0.225 (0.048)	0.218 (0.014)	0.213 (0.023)	0.3915

3.5.3.2 Experimental Results

This section presents two analyses to show the performance of the NGAP with both focused strategies against the original NSGA-III. The two analyses are statistical analysis and the convergence analysis. However, the convergence analysis in this phase is done differently from the second phase. Due to using the time as a termination criterion, It becomes increasingly difficult as the execution time in each generation is not the same, i.e., a generation which local search is performed. Therefore, only generational plots from selected executions (for more results on other instances, please refer to Appendix B) are shown.

Statistical Analysis

The red colour in all presented tables indicates that the coloured value is significantly different than the rest. Two coloured values in the same row means, while the two are significantly different from another value, there is no statistical significance between the two.

Table 3.29 show the average IGD measured from all 24 instances. The NGAP with strategy 2 (S2) dominates the original NSGA-III in all instances even though, they are given the same execution time. It is also interesting to see that there is no statistical difference in term of IGD between the NGAP with strategy 1 (S1) and strategy 2 (S2) in 5000-node instances, but with bigger instances (except Dusseldorf), the disparity between S1 and S2 is large. This finding contradicts to the outcome from [59] and is discussed later in the following section.

Spread indicator is displayed in Table 3.30. All algorithms yield comparable spread in all instances. This is to be expected since the local search aim is not to increase the diversity to the population, but focuses more on the quality of the solutions.

TABLE 3.30: Average results of evaluated algorithm for the spread metric.

Instance	Spread			
	Original	Strategy 1	Strategy 2	p-Value
5000_1	0.593 (0.040)	0.594 (0.037)	0.578 (0.039)	1.60e-01
5000_2	0.588 (0.032)	0.586 (0.039)	0.591 (0.038)	0.778
5000_3	0.598 (0.040)	0.585 (0.038)	0.585 (0.034)	0.222
5000_4	0.594 (0.035)	0.594 (0.042)	0.586 (0.043)	5.64e-01
5000_5	0.601 (0.052)	0.600 (0.048)	0.606 (0.036)	0.879
5000_6	0.590 (0.043)	0.598 (0.033)	0.602 (0.050)	0.608
5000_7	0.604 (0.032)	0.618 (0.049)	0.600 (0.039)	4.71e-01
5000_8	0.604 (0.052)	0.615 (0.040)	0.626 (0.058)	0.327
5000_9	0.607 (0.039)	0.605 (0.041)	0.608 (0.034)	0.883
5000_10	0.592 (0.042)	0.597 (0.044)	0.596 (0.040)	9.25e-01
50000_1	0.606 (0.032)	0.626 (0.051)	0.629 (0.039)	1.07e-01
50000_2	0.626 (0.043)	0.640 (0.053)	0.647 (0.050)	0.167
50000_3	0.650 (0.039)	0.629 (0.060)	0.636 (0.049)	0.233
50000_4	0.623 (0.047)	0.618 (0.042)	0.606 (0.044)	3.80e-01
50000_5	0.614 (0.049)	0.623 (0.044)	0.631 (0.042)	0.524
50000_6	0.611 (0.041)	0.627 (0.053)	0.629 (0.037)	0.182
50000_7	0.620 (0.036)	0.637 (0.046)	0.634 (0.047)	2.79e-01
50000_8	0.619 (0.047)	0.617 (0.038)	0.613 (0.048)	0.693
50000_9	0.616 (0.049)	0.618 (0.046)	0.624 (0.045)	0.951
50000_10	0.627 (0.037)	0.644 (0.043)	0.649 (0.050)	1.11e-01
Munich	0.571 (0.045)	0.623 (0.037)	0.588 (0.035)	3.59e-05
Hamburg	0.577 (0.041)	0.597 (0.051)	0.596 (0.036)	0.159
Berlin	0.595 (0.049)	0.586 (0.040)	0.567 (0.038)	0.04164
Dusseldorf	0.582 (0.041)	0.579 (0.049)	0.585 (0.046)	0.5909

The same trend in IGD is also showing in hypervolume (see Table 3.31). The proposed NGAPs are superior to the original NSGA-III in all 20 synthetic instances and four real-world instance. Moreover, NGAP with S2 is shown to yield better hypervolume than the one with S1 in big instances except on Dusseldorf instance where there is no statistical significance..

Convergence Analysis

Due to the execution time criterion, the averages of each indicator in each generation cannot be provided. Hence, the generational plots from some execution round are shown to demonstrate the trend of convergence of the algorithms involved in this experiment.

Figure 3.29 shows the generational plot of IGD of thee algorithms on the Hamburg instance. From the plot, it can be observed that both proposed NGAPs converge faster than the original NSGA-III. NGAP with strategy 2 is, in fact, yield the fastest convergence after only 6000 seconds of execution.

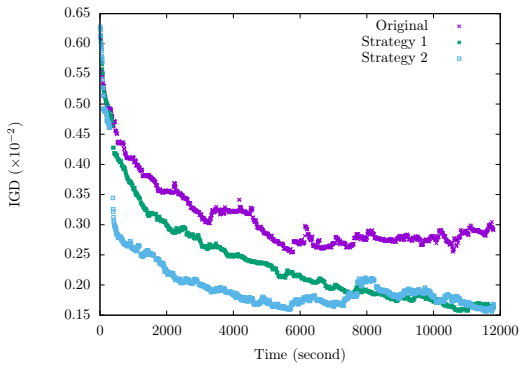


FIGURE 3.29: Generational plot of IGD of each algorithm on Hamburg instance.

TABLE 3.31: Average results of evaluated algorithm for the hypervolume metric.

Instance	Hypervolume			
	Original	Strategy 1	Strategy 2	p-Value
5000_1	0.390 (0.014)	0.414 (0.018)	0.416 (0.014)	5.61e-08
5000_2	0.375 (0.010)	0.387 (0.016)	0.391 (0.017)	6.09e-04
5000_3	0.394 (0.017)	0.417 (0.019)	0.426 (0.018)	3.89e-08
5000_4	0.376 (0.014)	0.394 (0.012)	0.393 (0.016)	1.84E-05
5000_5	0.378 (0.014)	0.405 (0.018)	0.409 (0.015)	7.58e-10
5000_6	0.376 (0.015)	0.388 (0.014)	0.390 (0.015)	0.002
5000_7	0.416 (0.015)	0.432 (0.018)	0.433 (0.017)	0.001
5000_8	0.364 (0.016)	0.390 (0.016)	0.394 (0.017)	1.35e-08
5000_9	0.385 (0.018)	0.399 (0.013)	0.404 (0.017)	0.001
5000_10	0.377 (0.016)	0.397 (0.014)	0.402 (0.014)	2.01e-07
50000_1	0.250 (0.015)	0.327 (0.018)	0.337 (0.018)	2.75e-11
50000_2	0.265 (0.013)	0.356 (0.016)	0.369 (0.016)	1.11e-13
50000_3	0.273 (0.012)	0.356 (0.012)	0.368 (0.014)	1.54e-14
50000_4	0.255 (0.014)	0.336 (0.013)	0.351 (0.012)	1.20e-14
50000_5	0.247 (0.012)	0.332 (0.012)	0.344 (0.017)	8.66e-14
50000_6	0.251 (0.014)	0.328 (0.013)	0.344 (0.013)	4.86e-15
50000_7	0.259 (0.011)	0.350 (0.013)	0.358 (0.016)	1.24e-13
50000_8	0.252 (0.010)	0.324 (0.014)	0.329 (0.014)	9.52e-14
50000_9	0.255 (0.012)	0.331 (0.012)	0.348 (0.016)	3.60e-15
50000_10	0.265 (0.016)	0.353 (0.015)	0.364 (0.015)	3.61e-14
Munich	0.296 (0.011)	0.306 (0.012)	0.320 (0.011)	2.53e-09
Hamburg	0.241 (0.009)	0.267 (0.009)	0.300 (0.009)	2.20e-16
Berlin	0.293 (0.014)	0.310 (0.014)	0.316 (0.012)	5.92e-08
Dusseldorf	0.392 (0.015)	0.387 (0.010)	0.394 (0.010)	0.05291

The generational plot reflects the same result showed in Table 3.30. In Figure 3.30, the spread over the whole time of execution, although fluctuate, they do not deviate from each other and stable at a specific range of values.

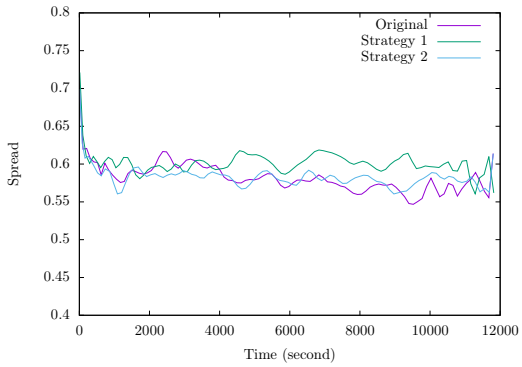


FIGURE 3.30: Generational plot of spread of each algorithm on Hamburg instance.

Figure 3.31 is the generational plot of hypervolume for each algorithm. In the figure, a sharp rise of hypervolume at the beginning of the execution from the NGAP with strategy 2 can be observed. While, the change in the hybrid algorithm with strategy 1 is more subdued, but still the growth is steady. Both hybrid algorithms show superior results to the original NSGA-III.

3.5.3.3 Results Discussion

In this section, the results from the performed analyses is clarified. There are 3 points to discuss in this context. The first one is the superior performance of NGAPs to the original NSGA-III. The second point is the similar

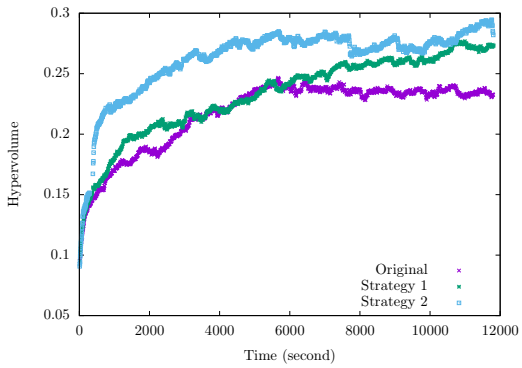


FIGURE 3.31: Generational plot of hypervolume of each algorithm on Hamburg instance.

diversity in solutions from all algorithms. Finally, the third point is the reason for better performance of S2 over S1 in the big instances.

NSGA-III and NGAPs

From the obtained results in this section, the hybrid algorithms show a much better solutions in term of IGD and hypervolume than the original NSGA-III. This is proved by the statistical analysis and the convergence analysis. This better solution quality can be contributed to the local search process, Pareto Local Search (PLS) and the proposed Extensible Neighbourhood Search (ENS).

To break the whole process down, ENS creates good building blocks (strings of high coverage locations) by changing locations in a solution to better ones (higher coverage in user and public transportation criteria) and these building blocks help in moving the solution toward the local optima (see Figure 3.32). As mention in [118], the global optima shares similar building blocks with many local optimum and local optimum are generally close to each other. From these facts, ENS can help in improving the solutions at the micro level. On the upper level, PLS screens out the dominated solutions since ENS does not guarantee that the quality of the improved solutions is better than the ones in the pool. Having the screening process offered in PLS reduces the time for unnecessary search and guides the improvement on a macro level.

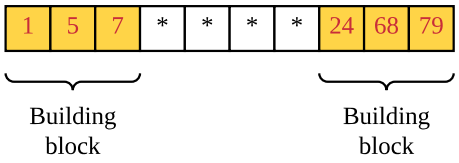


FIGURE 3.32: Building blocks in a solution.

Although the local search can contribute greatly to the algorithm, its effect can be observed at the beginning through the middle of the execution as seen in Figure 3.29 and Figure 3.31. Toward the end of the execution, the growth is stagnant. The reason is because the saturation of building blocks. once the solutions start to converge to one point, the growth is bound to be minimal. However, the sharp growth from the local search in the beginning definitely positively affect the outcomes compared to the original NSGA-III since even until the end, the original algorithm still cannot catch up with the hybrid algorithms performance.

Spread Similarity

One of the aspect that is not improved from the local search is the diversity in the solutions as seen in Figure 3.30. This is within our expectation. ENS (as the locals search operator) does not focus on create diversified solutions,

but on the two coverage objectives and number of vehicle minimisation. This can be explained through the three processes in ENS; *Add*, *Cut* and *improve*.

First of all, *Add*, true to its name, focus on adding locations to increase the user and public transportation coverage. The *Add* function is designed to work in conjunction with the *Improve* function, that is, the added locations are good enough to be exploited by the *Improve* function. The *Improve* function is also deterministic, so it is highly possible that the improved solutions end up with similar locations to achieve high coverage. Furthermore, the added locations from *Add* can even be removed by the *Cut* function. Hence, these two functions are not designed to enhance diversity.

The last function in ENS, *Cut*, is designed to reduce the redundancy of locations in a solution. From this idea, it also does not introduce any diversity concept to a solution. Hence, these are the reasons why the diversity aspect is not improved with the our local search process.

Performance of Strategy 1 and Strategy 2

When the obtained results of NGAP are observed closely, it is found that the results contradict the finding in [59]. According to [59], the hybrid algorithm with strategy 1 should yield the best result in large instances. However, it turned out that strategy 2 outperform it.

After the analysis of this event, two bases are found. The first basis is the different improvement methods in two strategies and the second basis is the deterministic property of ENS. Firstly, strategy 1 stop the ENS process once the first non-dominated solution is found, while strategy 2 keep improving the solution until the iteration limit. In this context, by stopping the improvement process as done in strategy 1, it ceases the chance to come up with better building blocks through further search leading to the founded building blocks to be not strong enough to be passed on or become diluted during the recombination phase. In contrast, strategy 2 keeps improving solutions to the limit which leads to high quality building blocks to stay throughout the execution. This can be seen from the convergence analysis, strategy 2 yields higher impact than the strategy 1 at the beginning and this impact carries on to the later generation.

The second basis that amplifies strategy 2 to outperform strategy 1 as contradict to the finding in the previous work is the deterministic property of ENS. In ENS, all rules are set to compare the current location with all locations in the considered neighbourhood. These rules are simple and based on the Pareto-dominance concept, that is, if a better location exists (it dominates the current location), then add it to the solution, or remove the current position, or move the current location to that better location. This means that during ENS process with strategy 2, if the solution cannot be improved, there is no need to repeat the process because the same neighbourhoods will be considered again and the same solution will be obtained. By being able to determine when to stop automatically, a lot of execution time can be saved in strategy 2 leading to more actions in recombination phase which in turn, new founded solutions are then exploited by the next local search process. Therefore, it is better to use NGAP with strategy 2 in VPP.

3.5.4 Third Phase Summary: Hybridisation and Local Search

In this phase, two novel algorithms are introduced. The first one is the novel hybrid algorithm, NGAP, that consists NSGA-III as a base algorithm and Pareto local Search (PLS). The second one is the novel local search operator called Extensible Neighbourhood Search (ENS). The hybrid algorithm follows the low-level teamwork hybrid strategy mentioned in [54]. While ENS utilises the neighbourhood concept in the graph theory to improve the solution. There are three functions in ENS, *Add*, *Cut* and *Improve*. An experiment are conducted

to compare the original NSGA-III with the hybrid NSGA-III employing two strategies; (1) Stop the local search once the first dominating solution is found and (2) Stop the local search once the limit threshold is reached. The experiment is perform on 24 instances varied in size. The obtained results show that NGAP with strategy 2 outperforms the original NSGA-III in all 24 instances and is better than its counterpart with strategy 1 in all large instances. These results are collected after given each algorithm the same execution time. Hence, it is advisable to employ the proposed hybrid algorithm in VPP since it yields a better result than the original NSGA-III.

Chapter 4

Chaos Theory and Large Scale Graph Discovery

In the previous chapters, the novel hybrid metaheuristic algorithm, NGAP, is presented. The research also illustrates a method for graph utlisation as a facilitator to formulating an optimisation model and how it can improve the quality of the solutions significantly. Yet, It must be noted that, those graphs are only considered to be medium-sized. And most likely, the application is going to be fixated on only city-level since that is the main interest. On the other hand, with a global outlook, a graph can represent data in several domains, not just a city street network and of course, can be several times larger than those instances in the fleet placement problem. Although, it is a fact that a real-world graph can hold a lot of precious information, its potential size and ever changing aspect make it difficult to make use of. Examples of graphs that demonstrate these aspects are found in in biology, social network and World Wide Web [121, 122]. Coupling with the "Big Data" era, the size of these graphs can be extremely large. A very prominent example of large graph is the Internet which was estimated to be over 30 billion nodes. Another interesting example from biology is the human brain which consisted of 10^{11} nodes [122]. These numbers were just estimations. In reality, in such huge graphs, their available information is never complete, or even worse, completely unknown [122], hence, in this research, define such graphs "unknown" graphs. A lot of efforts were also put into large graph analysis as the application can cause a great impact on many communities. Taking a benchmark suite called "Graph500" as an example, it contains a graph with around 67 million nodes [123] and this benchmark has been widely used as another index to test "High Performance Computing" (HPC) clusters performance. However, unlike previously mentioned large graphs, this graph was synthesised and all the information was known after the generating process according to [122]. As large real world graphs are not completely known, graph exploration or graph traversal algorithms are needed to learn more about the graphs in order to extract meaningful information. Therefore, graph traversal is the first step for knowledge extraction from the real world graphs. In this respect, the focus of this paper aims at a large graph discovery. The focus is on large graphs where in nature, global information and properties are unknown prior to the discovery.

A Graph G is a set of vertices and edges defined as follows $G = (V, E)$. Edges are unweighted and undirected. The Graph Traversal problem in this work is formulated similarly to a problem introduced in [124] which created the shortest tour based on local decisions. An *agent* is an entity visiting the various nodes of the graph. Under this formulation, the agent does not have any information about a topology of the graph nor its size, it can only learn about the neighbours of the vertex v when it visits the vertex. This is described as the "fixed graph scenario" [124]. The agent starts from an arbitrary (random) node and traverses to the next

node crossing an existing edge. Classical termination criteria for the graph traversal problem are when all the nodes or links have been fully discovered or when a tour has been constructed which relies on the fact that the traversed graph is known and memory is unlimited [125, 124, 126]. However, in our scenario this criterion is not applicable considering that the graph size is unknown. Hence, in order to compare the exploration performance of the algorithms, a Coverage Percentage is used. The metric is the ratio of discovered nodes after a predefined number of iterations to the number of nodes in the tested graphs. Here, the research question is formulated, i.e. what is the best agent strategy for node selection given that the agent is memoryless and that the objective is to maximise the discovery of unvisited nodes at each step.

Generally, algorithms using random numbers are used to solve the graph traversal problem [127, 128]. This gives rise to a question whether there are any other strategies to solve this problem. As a result, our research on chaotic dynamics is conducted. In the optimisation field, recent works include chaotic dynamics to replace the random part of algorithms to enhance their performance [129, 130]. For instance, considering the "Particle Swarm Optimisation" (PSO) algorithm, chaotic maps are used for the diversification: logistic maps [131, 132] or more complicated maps [133, 134, 135]. In this work, two chaotic dynamics are chosen, namely, Lozi map and Rössler system. The complicated dynamics obtained from both systems are explored and the study on their effects on the traversal problem is also conducted. Moreover, the in-depth study of the Rössler system is also made possible as there are already tools to compare these non equivalent dynamics using a bifurcation diagram [136].

4.1 Graph Discovery

The topological structures of graphs used in literature to highlight the potential application of our algorithm in various fields are described. Then, state of the art of the graph traversal algorithms are reviewed.

4.1.1 Graph Topologies

In this research, five topologies are studied; ring, small world, random, grid and power-law because these topologies are found widely in the applications in the real world. Brief example usages of each topology to establish the understanding of wide applications for these graphs are given in this section. Starting from the Ring topology, ring topology can be found mainly in Network field as it is relatively simple and reduce the packet collision in the network. This kind of network also allows the growth in the network without impacting the performance. An obvious example to show that ring network can be large is the "Metropolitan Area Network" (MAN). This network has to accommodate to the increase usage of traffic data in the metropolitan area. There were some projects that proposed the architecture of the MAN in a ring topology which were HORNET [137], RINGO [138] and KOMNET [139].

The second topology is a small world Topology. Six degree of separation is the main concept of the small world topology. It means that each node or entity in the graph can be reached within six hops (on average) from any nodes[140]. This topology is prominence in the social network graphs where the small world properties can be found [141]. It was also shown in [142] that the World Wide Web (WWW) has the small world properties. In addition, the small world effect also inhibits in biological networks such as "Protein-Protein Interaction" (PPI) [143] and "Metabolic Network" [144] and in infrastructure networks such as the railway network mentioned in [145]. An interesting use case is the use of the loss of small world effect in the brain graph to analyse the Alzheimer disease in [146]. This demonstrates that the small world effect actually exists in the real world problems.

The next topology is the topology that appeared in many works: random topology. One of the most popular random graph is Erdős-Rényi random graph proposed in [147]. The random graph might lack the transitivity and has unrealistic Poisson degree distribution, but it is a staple for graph problems since it is one of the graphs that has been used to test algorithms [122, 148, 149]. Not to mention that random graphs with some degree distributions, can be used to predict and model the real world graph as well [150]. This property makes random topology a good choice for a preliminary testing of an algorithm. That is not all to the random graph. In [151], the authors use the merging of random graphs to generate patterns and use them in classification. The random graph was also introduced as a model to analyse the programming code and refined it to have a low error-rate [152].

Another topology that is heavily used in robotic field is a grid topology. Grid topology is used as a problem model in covering problem. The plain is usually modeled in grid with or without obstructions [153, 154, 155, 156]. Even though, in the mentioned work, they used "Grid Map", the map can be translated into a graph without any loss of information. It is for this reason that this topology is included into the experiment to test the versatility of our algorithm.

The last topology is power-law topology. Power-Law graphs have the degree distribution that conforms to the power-law distribution [157, 158]. The property of power-law topology can be found in several real world network such as the Internet [159, 160] and biology graphs. In fact, past research pointed out that power-law distribution exists in many biological graphs such as Yeast-Protein Interaction networks, Metabolic network [161], "Protein-Protein Interaction" (PPI) networks [143] and Cellular networks [162]. From these findings, it is important to point out that it is common to find graphs that contain several properties (from various topologies) such as a PPI network where a small world effect can be found along with the power-law distribution. Moreover, these networks are growing each day due to the new discoveries, even existing ones are already large, e.g. brain network that contains a few hundred billions nodes [122].

Given the applications of each topology and their associations with the real world use cases, there is a possibility that these graphs can be large and contains a lot of useful information. These topologies represent real problems in various fields; networking, social network analysis, biology, and robotic. Furthermore, It is not always the case that these graphs are known beforehand since they are growing rapidly due to the age of discovery and become very hard to track. hence, these are reasons why these topologies are selected in this research experiment to demonstrate the potential of application in the real world.

4.1.2 Graph Traversal Algorithms

The purpose of the graph traversal problem is to discover vertices (or nodes) in the graph and at the end, learn about the information that a graph represents. Graphs can be simply classified into two types based on their available information, i.e. known and unknown graphs. In the case of known graph, the global information of the graph, such as the size or the topology, is available beforehand. An algorithm can then exploit this knowledge to efficiently execute its traversal on the graph. On the other hand in an unknown graph, no information is available which makes it more difficult to traverse it. In this work, the interest lies in the latter problem with the aim of using no memory or as minimal memory as possible.

The unknown graph traversal mostly finds its roots in the robotic field as the robot needs to explore an unknown terrain and is known as the covering problem [125, 126, 163, 164]. It was started with a Random Walk algorithm which let a robot choose the next destination randomly through a uniform random function [127, 128]. Then, the heuristic algorithm called the "Nearest Neighbour Approach" [165] was introduced. It is a greedy algorithm that creates the shortest path from the current node to the destination based on the retrieved local information

(cost or weight) of the current node. After that, more heuristic algorithms: Depth First Search (DFS) [166] and Breadth First Search (BFS) [126] were introduced. Breadth First Search and Depth First Search performed extremely well on known graphs, however, they fell short in the unknown graph traversal and both need an additional feature called relocation. This feature relocated the starting point to the nearest neighbour of the previous starting point to prevent the algorithm from being stuck in a loop. In [128], the authors showed the performance comparison between those algorithms on connected directed random graphs. They also pointed out that randomised algorithms actually did not compromised any performance comparing to DFS and BFS algorithms. Yet, these works were tested on small graphs only. The memory issue will show itself when algorithms like BFS and DFS are used to traverse the graph as the required memory for both BFS and DFS algorithms is $O(b^{d+1})$ where b is the average out-degree and d is the distance from the starting node. This distance d can be viewed in the same fashion as hops from the starting node or levels in a tree graph from the root [167].

Even though memory is a big issue in a big graph traversal, it is not the only one; traversing time is also an issue. Several studies contributed to parallelise those algorithms. For example, in [168], the author parallelised Breadth First Search (BFS) on Central Processing Unit (CPU) and Graphic Processing Unit (GPU) and also proposed a hybrid method which alternates between sequential, CPU and GPU implementations depending on the situation. However, it can be applied on known graph only which is a primary constraint of the BFS algorithm since the algorithm need to know when all nodes are discovered to stop the search which is not the case in unknown graph scenarios. Looking into an unknown graph traversal problem, agent-based systems are more widespread since they require little resource and relatively easy to parallelise [169, 154, 155]. For example, Ilcinkas et al. proposed a "Port Selection Technique" which requires each node to have a set of memory [170]. Then, this memory is used to remember which port has already been selected by agents and not to be repeated. The algorithm can also accommodate multiple agents. Apart from this technique, ant-based algorithms are also popular. Wagner et al. proposed a covering method based on ants and pheromones [169]. The pheromones were used to enhance the exploration. They proposed and compared two approaches: the first one leaves pheromones on edges and the other one leaves pheromones on vertices. The result showed that in the latter, the covering time was significantly lower. The authors also argued that this method is efficient even for dynamic graphs where edges or nodes can be altered unlike the DFS. Koenig et al. proposed a repeated coverage method using ant robot [154] . The authors experimented on the greedy method in which the agent always take the node with the least visiting number as the next destination. Another method is called Learning Real-Time A* (LRTA*) where the agent calculates the attractiveness of the node based on its unvisited or least visited neighbours. From the experiment, the author concluded that in most situations, LRTA* is more efficient than a greedy algorithm. Another work proposed a simple multi-agent system to keep patrolling (continuous exploration) and aimed to achieve a uniform visiting frequency for all edges [169]. As there are many classes of graph, other authors also proposed the agent-based graph traversal algorithm called "BEER" on cyclic and tree graphs [171]. The authors also pointed out that a greedy algorithm can still work very well on tree without any need of sophisticated methods. Still, all aforementioned approaches need memory to store the already selected paths or the pheromones on all vertices.

Some examples of graph traversal algorithm applications can be found in [172, 173]. Dentler et al. proposed an ant colony based algorithm to create pattern-based inference rules on the Resource Description Framework (RDF) Graph for Semantic Web Reasoning whereas the ant-based system performed better than a random strategy [172]. To obtain the inference rules, ants needed to traverse the graph first and build rules simultaneously making traversal algorithm a crucial part of the process. There is another work related to knowledge discovery: Tiddi et al. proposed "Linked Data Traversal" which uncovered the unknown graph on real time [173]. In addition, the traversal problem is also studied in the dynamic graph domain where the graph is

changed over time. In [174, 175], the authors used the token traversal method to find the network backbone in a dynamic ad-hoc network. The algorithm only utilised a single hop knowledge to determine the next node to traverse which made it completely based on local information. This idea is very suitable for a large unknown graph. On that account, the application of various graph traversal algorithms can potentially be extended to more of the real world use cases.

Due to the fact that chaotic dynamics had been reported to improve the performance of optimisation algorithms [129, 176, 134] and covering algorithm [156] over the usage of a typical uniform random function, the intention is to test it against a Random Walk algorithm which is memoryless and employs a uniform random function as well to study the result and the chaotic dynamic contribution on the graph traversal problem enhancement. Moreover, with the deterministic property of a chaotic dynamic, it also promotes reproducibility of the work. From the best of authors knowledge, there is not yet a work that integrates chaotic behaviour into the graph traversal algorithm and requires no memory or constant memory to process. Therefore, this research broaden the horizon of chaos theory. In the next section, chaotic behaviour and dynamics are elaborated in order to explain the algorithm of such a unique character.

4.2 Chaos Theory

As defined by the Encyclopaedia Britannica: "Chaos theory, in mechanics and mathematics, is the study of apparently random or unpredictable behaviour in systems governed by deterministic laws.". Chaos theory is not only limited to the study of the system alone, but also had long been applied in many problems such as optimisation problems [129] where the chaotic maps were tested against Random process. An improvement in performance was also reported. In effect, chaotic systems were introduced in several works related to optimisation processes such as [133, 177, 176] where the uniform random function was replaced by the chaotic system and performance improved. However, the optimisation is not the only field that has seen a use of chaotic dynamics, it can also be used to generate random numbers as appeared in [178] where the Lozi map is used for "Chaotic Pseudo Random Number Generator" (CPRNG). Hence, there is another kind of work where, instead of utilising chaotic map directly, CPRNG is used in place of the typical uniform random number generator, for example, Pluhacek et al. fine-tuned Lozi map and CPRNG, and improved the performance of "Particle Swarm Optimisation" (PSO) algorithm over the one that employed the uniform random function [135]. Another recent work which employed chaotic dynamics was the coverage problem by [156]. The authors integrated the chaotic behaviour to enhance the drones' searching capability. The chaotic behaviour has demonstrated its ability to enhance the algorithmic performance over purely random behaviour in optimisation and covering problems [133, 129, 176, 177, 156]. With similarity between a covering problem and a graph traversal problem along with studies that confirmed an improved performance of chaotic systems over a random behaviour, a chaotic system is used as a based for our proposed graph traversal algorithm. Two chaotic dynamics are utilised in our algorithm: the first one is the Lozi Map and the second one is the Rössler System. Lozi map is a simple map which contains only piecewise linear equations in two-dimensional space while, the Rössler System is defined by three differential nonlinear equations in three-dimensional space. The two dynamics are introduced in this section.

4.2.1 Solution of a Dynamical System and its Properties

The deterministic chaotic solution of a system has three important properties, these are:

1. globally time invariant;
2. highly sensitive to the initial conditions;
3. aperiodic.

The first property implies that, given the same parameters, the attractor at $t = 0 \rightarrow t = 10$ is the same as the attractor at $t = n \rightarrow t = n + 10$. The second property implies that even considering a difference in the scale of 10^{-10} for the initial conditions, the system will divert from the original track. The last property points out that the chaotic system always gives out non-periodic solution. However, it is important to note that the solution can be chaotic or periodic: it depends on the system's parameters.

In this work, two chaotic dynamics are involved in he experiment; Lozi map [179] and Rössler system [180]. Lozi map is a discrete chaotic map made of two dimensions which depends only on two parameters. While, on the other hand, Rössler system is a continuous system of three differential equations (three dimensions) that depends on three parameters.

4.2.1.1 Lozi Map

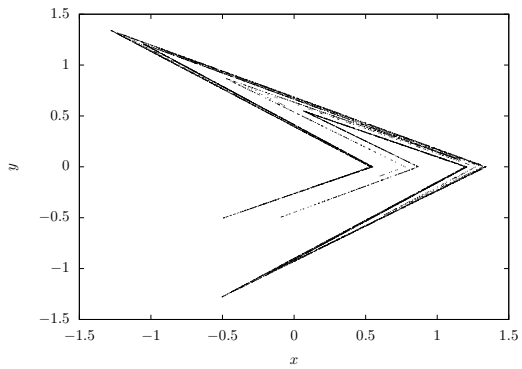


FIGURE 4.1: Lozi attractor with the parameters $a = 1.7$ and $b = 0.5$ solution to Eq. (4.1).

the Lozi map is employed since it is also relatively simple by its definition, yet offer a certain degree of alteration to the behaviour of the dynamics. The Lozi attractor is a strange attractor similar to the Hénon attractor. It is defined by two equations in Eq. (4.1) [179].

$$\begin{cases} x_{n+1} = 1 - a|x_n| + y_n \\ y_{n+1} = bx_n \ . \end{cases} \tag{4.1}$$

For the Lozi attractor, x and y are variables. At time $t = 0$, these variables are also called initial conditions. The attractor is modified through the parameter a and b. The Lozi attractor with $a = 1.7$ and $b = 0.5$ is shown in Fig. 4.1. For further reading, the reader are referred to [181] for in-depth analysis of the behaviour of Lozi map.

4.2.1.2 Rössler System

First of all, it is important to note that in our work, Rössler system is defined by three "Ordinary Differential Equations" (ODE). This aspect makes the Rössler system in our work a continuous system. There are also tools to solve ODE from several fields that deal with nonlinear systems. In this research, "Runge-Kutta method (fourth order)" (RK4) is selected. This step increases the computation time. However, combining it with the

Poincaré section (discretisation method), the analysis of chaotic dynamics can be done which is needed in our work since the aim is to study the link between chaotic dynamics and graph traversal problem. Three major steps for calculating the solution from a chaotic system which are explained in detail later in this section are then presented. These steps focus only on the outcome of the system, further details can be found in [136] for the Rössler system.

1. Choose the parameters and initial conditions of the system that give an attractor as the solution for the system (Fig. 4.2).
2. Discretise the system using Poincaré section (Eq. (4.4)).
3. Compute the First Return Map (the dynamical signature of the system) (Fig. 4.3).

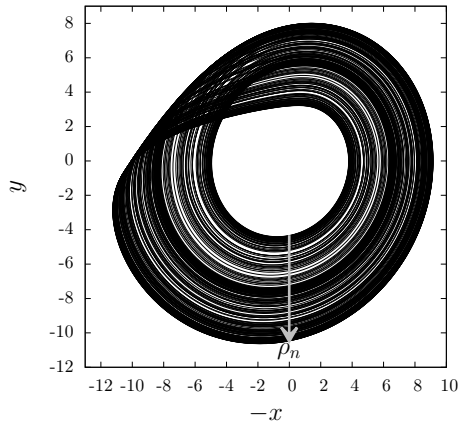


FIGURE 4.2: Attractor solution to the Rössler system (Eq. (4.2)) for $\alpha = 0$ (Eq. (4.3)) with Poincaré section. The arrow indicates the orientation of the normalised value ρ_n.

The Rössler system [180] is a three differential equations system:

$$\begin{cases} \dot{x} = -y - z \\ \dot{y} = x + ay \\ \dot{z} = b + z(x - c) \ . \end{cases} \tag{4.2}$$

This system provides the solution which can be an attractor or a periodic solution. The first derivative $\dot{x}$, $\dot{y}$ and $\dot{z}$ is the definition of the problem where the x, y and z are the variables of the system where all variables at $t = 0$ are called initial conditions. A series of variables x, y and z from $t = 0 \to \infty$ is the solution of the system. Lastly, a, b and c are parameters to alter the behaviour of the system. However, [182] already came up with a method to generalise these a, b and c parameters modification with α as seen in:

$$\begin{cases} a = 0.2 + 0.09\alpha \\ b = 0.2 - 0.06\alpha \\ c = 5.7 - 1.18\alpha \ . \end{cases} \tag{4.3}$$

The bifurcation diagram (Fig. 4.4) illustrates the effect of how the parameter α can alter the system. On the x-axis is the value of α. On the y-axis is the value of the solution in the Poincaré section

$$P \equiv \{(y_n, -z_n) \in \mathbb{R}^2 | x_n = x_-, -\dot{x} < 0\} \tag{4.4}$$

where x_- is the x value of the singular point of the Rössler system in the center of the attractor (see [136] for details). Such a parametrisation gives a bifurcation diagram following the topological properties of the

attractor (Fig. 4.2). From y_n in Eq. (4.4), it is normalised to obtain ρ_n and plot it to visualise the first return map (Fig. 4.3). The detailed method to obtain first return map and its application is explained in Section 4.2.2.

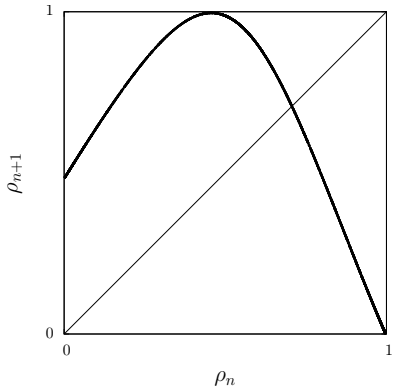


FIGURE 4.3: First Return Map to the Poincaré section for $\alpha = -0.25$.

4.2.2 Bifurcation Diagram for Optimisation

The bifurcation for the Rössler system and the Lozi map are presented in this section. In this work, the bifurcation diagram is used to represent states of a solution for each parameter whether it is periodic or chaotic. For the Rössler system, the parameters are generalised with α as shown in Eq. (4.3). In this section, a Rössler system bifurcation diagram where $\alpha = [-0.5, 1.2]$ is presented. On the other hand, Lozi map has two parameters, a and b. Two sets of parameters are selected, $a = 1.5, b = [-0.5, 0.47]$ and $a = [1.1, 1.8], b = 0.1$. Both were mentioned in [181],

4.2.2.1 Rössler System Bifurcation Diagram

A bifurcation diagram of the Rössler system is obtained through Poincaré Section. From the bifurcation diagram, the behaviour of the system at each α can be observed which means that first return maps can be provided. Then, it enables the research to proceed with the optimisation of an algorithm afterward depending on first return maps. The bifurcation diagram of the Rössler system with $\alpha = [-0.8, 1.2]$ is shown in Fig. 4.4. There are dense periods and sparse periods. The dense periods are the chaotic attractors which are the focus of this work and the sparse periods are the periodic solutions where the solution alternates between few values. For example, at $\alpha = -1.5$, there is only one point and is called a period-one solution. Another example is $\alpha = -1$, there are only two points and is called a period-two solution. These two α (and the likes) do not show the chaotic dynamic behaviour. Thus, it can be clearly seen in Fig. 4.4 that by varying the value of α, chaotic or periodic solutions can be provided.

Regarding the chaotic solutions, it is also worth mentioning that there is a special case among chaotic solutions as well, called "Banded Chaos". It is an attractor or a solution that is composed of stripes or bands. In Fig. 4.4, around $\alpha = 1.1$, there is a gap in the middle which is different from the area where $\alpha = 0$ and that is an example of banded chaos area. As for the characteristic of the banded chaos, it can be regarded as a more complex than a normal chaotic attractor. Readers who would like to pursue further detail are referred to [136]. Regardless of chaotic or periodic solutions, once the system is solved, a continuous solution is obtained.

For discretisation, a Poincaré section is utilised as can be seen in Eq. (4.4). The interpolation between two sequential discretised points is calculated. This process is illustrated in Fig. 4.2. The line is drawn on the plane and cut though the solution and then the interpolation can be computed.

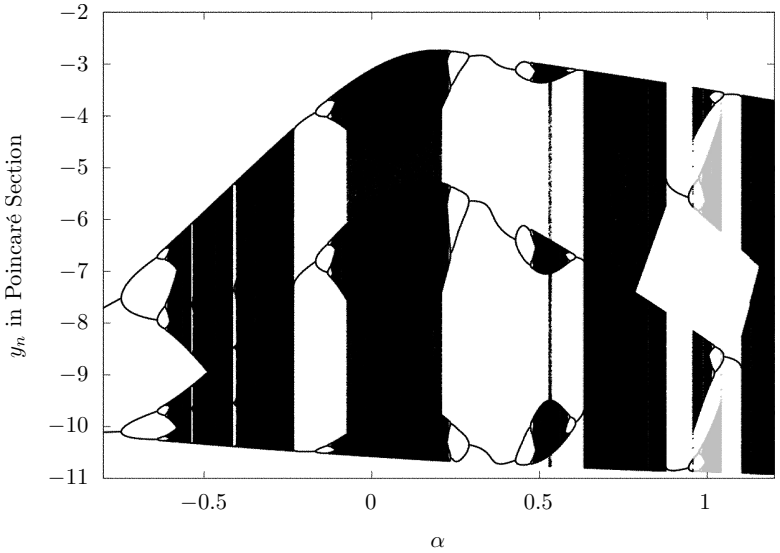


FIGURE 4.4: The bifurcation diagram where α is varied in Eq. (4.3) for the Rössler system Eq. (4.2).

After the discretisation, the computation of the first return map follows. The discretised values from Poincaré section are normalised to obtain $\rho_n \in [0, 1]$ from y_n:

$$\rho_n = (y_n - UB)/(LB - UB) \ . \tag{4.5}$$

The normalisation is based on the bifurcation diagram and equations are used to estimate the upper bound and lower bound. However, bifurcation diagram is not in linear shape, thus it is divided into several parts to ensure that the retrieved values of ρ_n are between 0 and 1. LB and UB of Eq. (4.5) can be found in Eq. (4.6) and Eq. (4.7) respectively. The partitions on the bifurcation diagram are also shown in Fig. 4.5.

$$LB = 0.110626\alpha^2 + 0.4141\alpha - 10.585 \tag{4.6}$$

$$UB = \begin{cases} 6.04162\alpha - 2.8 & \text{if } \alpha = [-0.5, -0.22] \\ -7.16446\alpha^2 + 3.31662\alpha - 3.05252 & \text{if } \alpha = (-0.22, 0.192] \\ -0.99127\alpha - 2.49 & \text{if } \alpha = (0.192, 0.790) \\ -4.31521\alpha - 4.01651 & \text{if } \alpha = [0.790, 0.876] \\ -0.99127\alpha - 2.49 & \text{if } \alpha = (0.876, 1.102) \\ 23.673\alpha - 34.2435 & \text{if } \alpha = [1.102, 1.155] \\ -0.99127\alpha - 2.49 & \text{if } \alpha = (1.155, 1.2] \end{cases} \tag{4.7}$$

In [136], it had been shown that the Rössler dynamics can be considered similar as a whole using templates and subtemplates. Thus, to generate non-equivalent dynamics, a partition out of a bifurcation diagram is needed to be created. Then, it is also very important to note that the fourth and sixth equation in Eq. (4.7) only considers one band from a whole banded chaos as appeared in [136] in first return maps section. Otherwise, if the whole bands are taken, it will result in discontinuous first return maps and shows a behaviour that is similar to periodic solutions which is not our focus. However, this specific partition is not required for $\alpha = 1 \pm 0.05$ because the two co-existing attractors have the topological structure as it is for the beginning of the bifurcation diagram. The templates of the two co-existing attractors are symmetric by inversion for these values of α. When α increases, the templates are the same for the co-existing attractor when α decreases. In addition, these templates have the same topological structure of those observed at the beginning ($\alpha < -0.25$) of the bifurcation diagram: templates with two branches with one increasing and one decreasing [136]. Finally, there

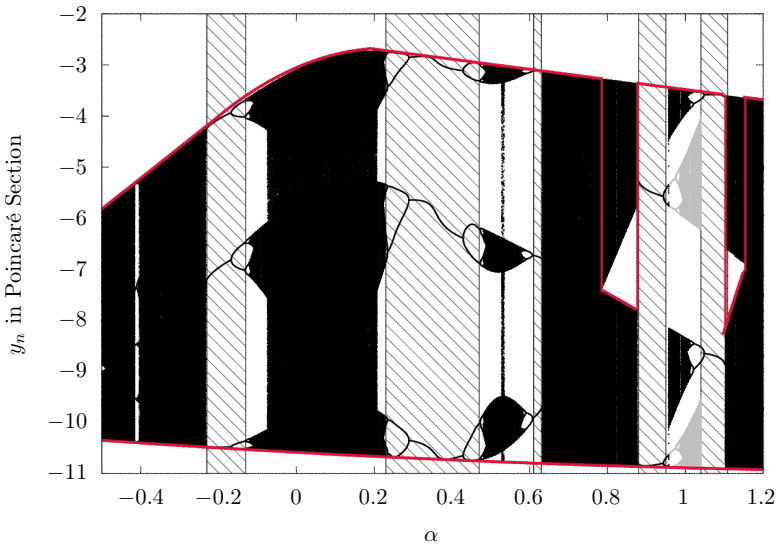


FIGURE 4.5: Partitions in the bifurcation diagram according to Eq. (4.6) and Eq. (4.7)

is a redundancy in the templates if all of them are considered and the partition performed using UB in Eq. (4.7) permits to variate α to produce non equivalent dynamics.

The first return map which is a dynamical signature of the system can then be realised by plotting the current discretised solution against the next. The first return map can have many shapes depending on the provided α. Fig. 4.3 is just one of many possibilities available from the Rössler system. After an extensive study of the system, it is found that the system itself, regardless of the first return map, incorporates prominent patterns. So far, three prominent patterns are found. They are related to the periodic orbits of the system that are visited more often. These patterns are discovered through observing the plot of extracted solutions of the studied systems.

1. **Growing Pattern**: The value of the solution starts from a very low value and gradually increasing to the high value. Once the high value is reached, the solution value drops to low and the process start again as can be seen in Fig. 4.6.

2. **Oscillation Pattern**: The value of the solution alternates between high and low values. The example is shown in Fig. 4.7. This patterns corresponds the period-2 orbit for Fig. 4.7.

3. **Plateau Pattern**: There is little change in term of value of solution during this pattern period. The Fig. 4.8 might not be totally linear, but the change in value is still small. This value corresponds to the period-1 orbit of the system for Fig. 4.8.

It is important to note that each α yields a different first return map. This corresponds to the frequency of the patterns and as a consequence, a different behaviour. In addition, these patterns correspond to the unstable periodic orbits of the attractor, but since the value of α is varied, the capability to demonstrate more links between the orbits and the patterns is limited (see Fig. 13 of [136] for more details about the dynamical partition of this bifurcation diagram).

4.2.2.2 Lozi Map Bifurcation Diagram

The bifurcation diagrams of the Lozi map used in this work are based on the value of x from the map. Two bifurcation diagrams of the selected parameters ($a = 1.5, b = [-0.5, 4.7]$ and $a = [1.1, 1.8], b = 0.1$) are shown

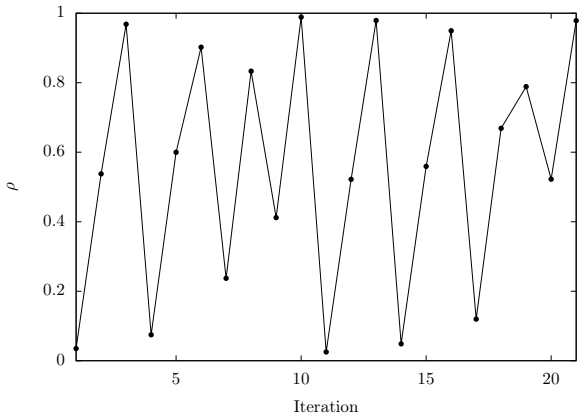


FIGURE 4.6: A plot of ρ_n against iteration for growing pattern.

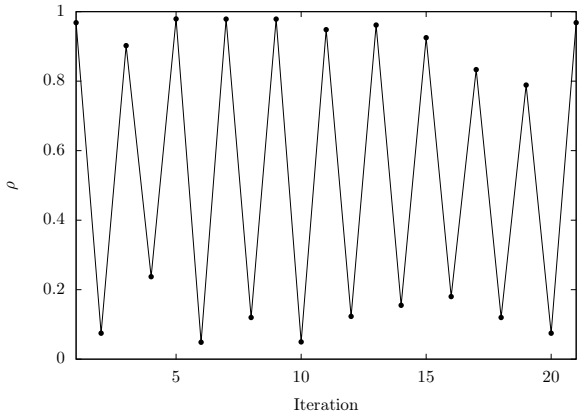


FIGURE 4.7: A plot of ρ_n against iteration for oscillation pattern.

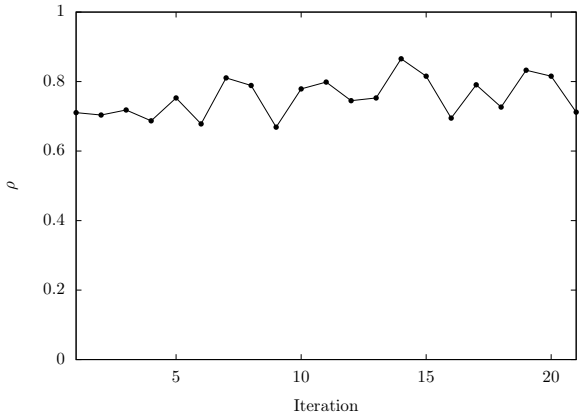


FIGURE 4.8: A plot of ρ_n against iteration for plateau pattern.

in Fig. 4.9 and Fig. 4.10. As far as the authors know, there is no possible comparison between the chaotic mechanisms as it has been done for the Rössler system using topological analysis. Some works in that direction are still in progress using suspension of the map [183, 184]. Thus, only the same partitioning methodology in Section 4.2.2.1 is applied to prevent the gap in the yielded x sequence. As a result from partitioning and Eq. (4.5) where x_n is used in place of y_n, the values of ρ_n from x_n are in $[0, 1]$.

The upper bound (UB) and lower bound (LB) in Fig. 4.9 for the Lozi map with $a = 1.5$ and $b = [-0.5, 0.47]$ are defined as follows;

$$LB = \begin{cases} -0.58728b^3 - 0.20915b^2 - 0.94836b - 0.50321 & \text{if } b = [-0.5, 0.4225] \\ 0.03116b - 0.831737 & \text{if } b = (0.4225, 0.47] \end{cases} \tag{4.8}$$

$$LB = \begin{cases} -4.82296b^2 - 1.47635b + 0.46621 & \text{if } b = [-0.5, -0.37) \\ 0.15227b^2 + 0.695025b + 1.00494 & \text{if } b = [-0.37, 0.4225) \\ -2.55404b + 1.30011 & \text{if } b = [0.4225, 0.47] \end{cases} \tag{4.9}$$

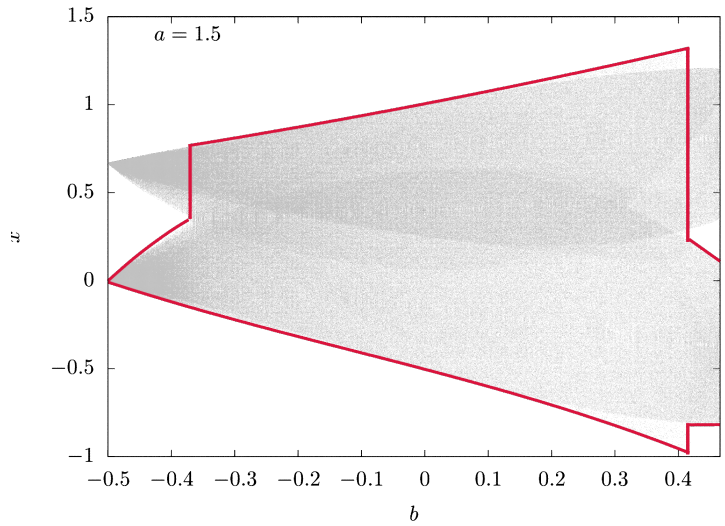


FIGURE 4.9: Partitions in the bifurcation diagram according to Eq. (4.8) and Eq. (4.9) for the Lozi map with $a = 1.5$ and $b = [-0.5, 0.47]$.

Upper bound (UB) and lower bound (LB) in Fig. 4.10 for Lozi map with $a = [1.1, 1.8]$ and $b = 0.1$ are defined as follows;

$$LB = -1.00691a + 0.90294 \tag{4.10}$$

$$LB = \begin{cases} 6.31887a^2 - 14.4394a + 8.04677 & \text{if } a = [1.1, 1.23703) \\ 2.8679a^2 - 5.92552a + 3.05 & \text{if } a = [1.23703, 1.40334) \\ -0.05148a + 1.14403 & \text{if } b = [1.40334, 1.8] \end{cases} \tag{4.11}$$

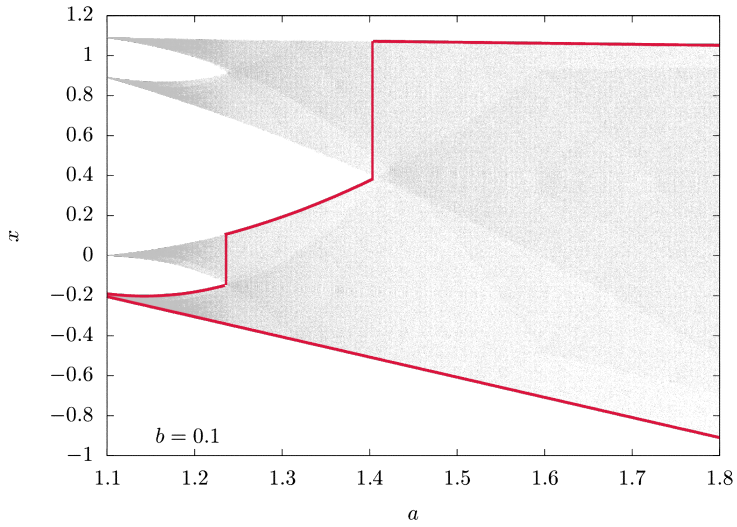


FIGURE 4.10: Partitions in the bifurcation diagram according to Eq. (4.10) and Eq. (4.11) for Lozi map with $a = [1.1, 1.8]$ and $b = 0.1$.

From all presented bifurcation diagrams of the Lozi map, all solutions are chaotic. In contrast, it is not the case for the Rössler system. This emphasises the importance of the bifurcation diagram as a preliminary filter for

chaotic solutions. Moreover, with the partition method, more complex chaotic behaviours from the continuous chaotic system, e.g. Rössler system, can be explored. At the same time, it also benefits the discrete chaotic system, e.g. Lozi map, by eliminating the gap in the yielded sequence of ρ_n if needed.

4.3 Chaotic Traversal (CHAT)

This section presents a novel chaotic agent-based graph traversal algorithm that focuses on exploring unknown graphs called "Chaotic Traversal (CHAT)". CHAT is actually similar to the Random Walk algorithm except that it does not use a uniform random function. Instead, CHAT chooses the next destination using a chaotic system (Lozi map or Rössler system in this case). With the same trait as Random Walk, CHAT carries over the same features as Random Walk, while having its performance enhanced thanks to the chaotic system integration. The whole process of CHAT is shown in Alg. 9.

Algorithm 9 CHAT Process.

Algorithm: Main Algorithm
Data: α, Termination Condition, Graph, Strategy
Result: Discovery nodes List
Initialise initial conditions for Rössler system
  Initialise an Agent with a starting point
  Initiate `Lozi Map` (Initial condition, a, b)
  Initiate `Rössler System` (Initial Condition, α)
  while *termination condition is not met* **do**
    `Chaotic Traversal` (Agent, Strategy)
Return out the result

Algorithm 10 Generating Lozi map solutions.

Algorithm: Lozi Map
Data: Initial conditions, a, b
Result: ρ (normalised solution)
Loop
  Simulate and solve Lozi map equations numerically
    Normalize x_n to obtain ρ_n (Normalised solution)
    Return ρ_n

Algorithm 11 Generating Rössler system discretised solutions.

Algorithm: Rössler System
Data: Initial conditions, α
Result: ρ (discretised solution)
Loop
  Simulate and solve Rössler system numerically
    if *the solution is in the Poincaré Section Eq.* (4.4) **then**
      Normalise y_n to obtain ρ_n (Normalised discretised solution)
        Return ρ_n

There are two versions of our algorithms: *Vanilla version* and *Circular version*. The Vanilla version is a completely memoryless algorithm, while the Circular version, being an improved algorithm, requires a very small amount (almost negligible) of constant memory in order to enhance the coverage performance of the Vanilla version. This section also presents the performance metrics used to compare the performance of CHAT and Random Walk in this work.

Algorithm 12 Agent Movement.

Algorithm: Chaotic Traversal

Data: Agent, Strategy

Get neighbour nodes from an agent position
  Assign equal priority to each neighbour
  Retrieve ρ from a Rössler system or a Lozi map
  if *Strategy == Circular* **then**
    Circulate the neighbour (Fig. 4.12)

Choose the next destination based on retrieved ρ
  Move to the destination

4.3.1 Vanilla version

In this version, the agent decides its next move based on the first return map values that the chaotic system gives out from Alg. 10 or Alg. 11. This version only utilises the local information such as the list of neighbours and edges without storing any path history, making it completely memoryless. To elaborate how it works, the Fig. 4.11 is given and the whole process of movement is shown in Alg. 12. From the current node 1, the agent had four choices. For each choice, it was given the same priority as can be represented by this set; $Neighbours = (2 : 0.25, 3 : 0.25, 4 : 0.25, 5 : 0.25)$. As mentioned in section 4.2, the solution from the system is normalised to be in range $[0, 1]$. Therefore, if the system gives out 0.7814 after discretisation, the agent would go to node 5. From there, the agent learns of the neighbours of node 5 and continues the process accordingly.

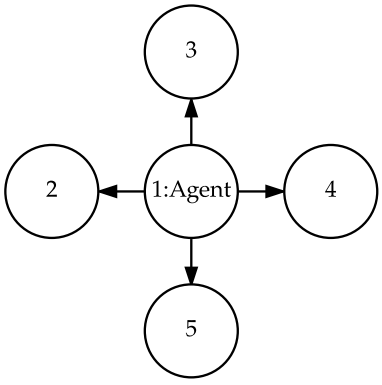


FIGURE 4.11: Agent comes to the node 1 from node 4.

However, one flaw in using the chaotic system is found for the graph traversal problem. The solution from the chaotic system is aperiodic or in other words, never gives out two same values periodically. However, the first return map is a combination of patterns where the pattern itself might repeat. With a very low chances of happening, the agent might enter in a loop. Considering this situation, the agent makes a tour, and when it arrives at the starting point again, for the next iterations, the system gives out the same patterns. Hence, the agent repeat the same route all over again leading to failure in a graph discovery.this problem is anticipated and a solution which incurs a negligible cost of memory is prepared.

4.3.2 Circular version

The word circular comes from the fact that the order of the neighbours in the list is circulated as in Alg. 12. In this Circular version, the chaotic system is still employed as a core function for choosing a next destination but add an ability to remember the latest node the agent has visited to prevent a loop traversal. The circulation

goes by the deterministic rules. Still the advantage of circulation is not only to prevent the loop traversal, but it also improves the Coverage Percentage since the previously visited node can be placed in the position that is not likely to be selected again so that the agent can avoid repetition. It can be explained using the Fig. 4.12.

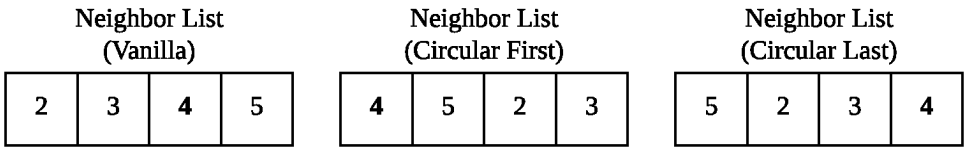


FIGURE 4.12: The difference between Vanilla, Circular First and Circular Last neighbour list in the situation of Fig. 4.11.

From the Fig. 4.12, it shows the list of neighbours in the situation of Fig. 4.11. The length of the list equals to the degree of the node. From the figure, supposedly an agent comes to the node 1 from node 4 (in bold) and the current node has node 2, 3, 4 and 5 as its neighbours. In Vanilla version, the neighbours order is not reorganised, hence, the loop traversing can occur. It is different in the Circular version where the order is reorganised by shifting the previously visited node (in this case, node 4) to the first position in the neighbour list. Here, two neighbour circulation methods are proposed, Circular First (as shown in Fig. 4.12) and Circular Last. The difference is the next position of the visited nodes: the Circular Last method left-shifts the previously visited node to the last position in the neighbour list. Combining these strategies with a first return map, the number of times that an agent revisits the already discovered nodes in a graph can be reduced. Moreover, with this approach, even if the same pattern is to be revisited, the agent can still prevent itself from traversing in a loop.

4.3.3 Performance Metric

In this work, two performance metrics, Coverage Percentage and Mean Time Coverage, are proposed. The former is used to measure the exploration performance of an algorithm while the latter measures the exploration efficiency by considering discovered nodes in the graph and the algorithm execution time.

4.3.3.1 Coverage Percentage

"Coverage Percentage" is used as a metric of exploration performance since typical performance metrics like "Traversal Time" which count how many iterations an algorithm takes to fully discover a whole graph cannot be used. This is due to the fact that real world graphs are unknown and the "Traversal Time" requires the size of a graph to terminate an algorithm and measure the performance. Therefore, "Coverage Percentage" is used instead since it reflects how many nodes an algorithm can discovered given a certain amount of algorithm iterations as shown in Eq. (4.12). For this metric, an agent run for n iterations where n is a number of nodes in a graph. The chaotic system solving time and algorithm execution time are not considered since the chaotic solution sequences can be generated separately and the execution time is varied from machine to machine. The metric is defined as follows:

$$\text{Coverage Percentage} = \frac{\text{Number of Discovered Nodes}}{\text{Total Number of Nodes}} \times 100 \ . \tag{4.12}$$

4.3.3.2 Mean Time Coverage

This metric is used to measure the time efficiency of an algorithm. It calculates how many (different) nodes can be discovered in one second (CPU time). Regarding the execution, the algorithm can be viewed in two separated

TABLE 4.1: Experiments formulation. The best values of α obtained from the small graph experiments are used to explore large graphs. Every α is tested for 300 runs.

Topology	Small Graph Experiment	Large Graph Experiment
Ring	$n = 2000$, $k = 10$, $p = 0$	$n = 1000000$, $k = 10$, $p = 0$ Vanilla's $\alpha = 1.123$ Circular First's $\alpha = 0.78$
Smalll World	$n = 2000$, $k = 10$, $p = 0.01$	$n = 1000000$, $k = 10$, $p = 0.05$ Vanilla's $\alpha = 1.123$ Circular First's $\alpha = 0.78$
Random	$n = 2000$, $k = 10$, $p = 1$	$n = 1000000$, $k = 10$, $p = 1$ Vanilla's $\alpha = 1.123$ Circular First's $\alpha = 1.124$
Grid	dimensions: 5×500, 10×250, 20×125, 25×100 and 50×50	dimensions: $50 \times 20000, 100 \times 10000, 160 \times 6250, 200 \times 5000, 320 \times 3125, 400 \times 2500, 500 \times 2000, 625 \times 1600, 800 \times 1250$ and 1000×1000 Vanilla's $\alpha = 1.123$ Circular First's $\alpha = 1.124$
Power-Law	$n = 2000$, $m = 10$, $k = 0.2$	$n = 1000000$, $m = 10$, $k = 0.2$ Vanilla's $\alpha = 1.123$ Circular First's $\alpha = 1.124$

pieces, sequence generation (solving Lozi map and Rössler system or generate a random values sequence) and graph exploration. The first part can be prepared beforehand (offline) and does not affect the exploration part, which is considered to be an online process. Therefore, only the CPU time required for the graph exploration is considered. This metric is dependent on the hardware platform, but it can be used as a guideline for what to expect from each algorithm. The Mean Time Coverage is defined as follows:

$$\text{Mean Time Coverage (nodes/second)} = \frac{\text{Number of Discovered Nodes}}{\text{Graph Exploration CPU Time}} . \tag{4.13}$$

4.4 Experimental Results: Rössler CHAT

In this section, the experimental setup and the results are presented along with their discussions. Experimental results are divided in two parts. The first one contains the result of CHAT on a small graph in order to study the impact of α in the Rössler system. The second part presents the results on 1,000,000 nodes graphs to evaluate the performance of CHAT with Rössler dynamics on large graphs.

4.4.1 Experimental Setup

The test platform is implemented in Python (2.7) and NetworkX [6]. The platform employs one agent in order to study the chaotic behaviour. All tests are run on the High Performance Computing platform of the University of Luxembourg [96]. The algorithms are tested on five graph topologies as listed in Tab. 4.1. All tested graphs are undirected and unweighted graphs. For Ring, Small World and Random graphs, are generated using Watt-Strogatz Generator. For Grid graphs, 2D-Grid Generator is used. Lastly, Power-Law Cluster Graph Generator is used to generate the Power-Law graphs. All the parameters being used are presented in Tab. 4.1.

All the graphs in the experiment are either small (2,000 nodes and 2,500 nodes) or large graphs (1,000,000 nodes). The 2,000 nodes graphs are used to show the impact of α on the Coverage Percentage of CHAT and for studying the optimal parameters for the chaotic systems. The large graphs aare used to confirm the finding in the small graph experiment. The sparse graph is generated by giving the generator a low node degree. For each topology, there are 10 different graphs and each one are tested 30 times with Rössler system based CHAT

for both small and large graphs with different starting points in each run. In case of grid topology, an agent is deployed on several shapes, not just a square one. In addition, to obtain the final α, each value of α and its effect on the performance on each topology are extensively studied to retrieve the ones that yield the best performance to test them on large graphs. Finally, the algorithms are compared against Random Walk (see Alg. 13) where the random values sequence can also be pre-computed offline like for chaotic systems. The reason for choosing Random Walk for comparison is because Vanilla CHAT and Random Walk are memoryless. In addition, the Circular CHAT, on the other hand, requires only marginal constant memory, while, other algorithms in the survey require memory to operate.

Algorithm 13 Random Walk of an agent.

Algorithm: Random Walk
Data: Graph
Result: Discovery nodes List
Initialise an Agent with a starting point
Generate random values sequence from a uniform random function
while *termination condition is not met* **do**
  Retrieve a value from a sequence
  Pick a destination based on a random value
  Move to the chosen destination
Return out the result

This section is organised into three main parts. The first part presents the result of the small graph experiment. The second part show the result of the large graph experiment and lastly, the study of α effect is presented in the last part.

4.4.2 Results on Small Graph Instances

This section shows the result of our algorithm comparing to Random Walk and also effects of varying the value of α in the Rössler system to analyse the influence of the dynamical properties on the behaviour of our algorithms. From the results in this section, the coverage differences of each approach are able to be observed and that aids us in choosing which combination of approach to be tested on larger graphs. For all figures in this section, the y-axis is the Coverage Percentage and the x-axis is the value of α supplement to the Rössler system. The black solid line is the result of the Circular First CHAT and the solid grey line is the result of Vanilla CHAT. Only Circular First method (named as "Circular" in all presented figures) is shown here due to its superior performance comparing to the Circular Last method. Lastly, the dashed line represents the result of the Random Walk. The shaded areas contain the result of the periodic solutions (in Rössler system) that does not exhibit chaotic dynamic behaviour. Hence, they are not taken into consideration since the sole focus of our work is on addressing graph traversal problem with chaotic dynamics. Please also note that it is not possible to shade all the periodic regions due to the fact that some periodic parts are small and located in between chaotic regions. In this work, the values of α in the range $[-0.5, 1.2]$ are considered because the dynamical analysis is already performed for each attractor solution and from Fig. 4.4, values of α that are less than -0.5 mostly yield periodic attractors. Thus, the chaotic dynamics properties in this range of values can be compared.

4.4.2.1 Ring Topology

The nature of the ring topology is a structured graph. Referring to Fig. 4.13, it can be seen that when the system gives out the periodic solution, there is a sharp rise in the Coverage Percentage for the Circular version of CHAT. In fact, all the sharp peaks (above 20%) for the Circular version are periodic solutions, e.g., at

$\alpha = -0.41$ and $\alpha = 1.0$. These periodic period are too small to shade. The average Coverage Percentage for the Circular version is about 10%. The highest Coverage Percentage (about 17%) for the Circular version is obtained with $\alpha = 0.78$. The Vanilla CHAT cover about 60%. While the Circular CHAT also has a good Coverage Percentage varied by the α. All in all, both chaos-based algorithms outperform Random Walk which only cover about 10% of the whole graph.

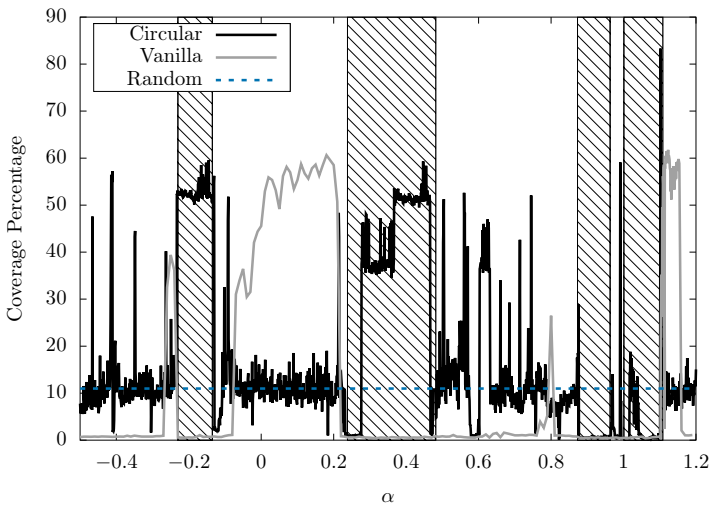


FIGURE 4.13: Performance of Rössler CHATs in comparison to the Random Walk on the ring topology. The black solid line is the result of the Circular CHAT and the greosslery line is the result of Vanilla CHAT. The dashed line represents the result of the Random Walk. The shaded areas highlight periodic solutions of the Rössler system.

4.4.2.2 Small World Topology

This topology exhibits the theory of six degree of separation which is one of the most important aspects of the social network [141]. In the small world topology, Vanilla CHAT is able to outperform in Coverage Percentage of the Random Walk by 8% at $\alpha = 1.123$ with its 50% Coverage Percentage. The Circular version, even though, has lower Coverage Percentage than the former, is still able to produce a better performance than the Random Walk as well. However, when comparing Fig. 4.13 and Fig. 4.14, it seems like the effect of periodic solutions and chaotic solutions (in Rössler dynamics) become less distinctive as the graphs are less structured.

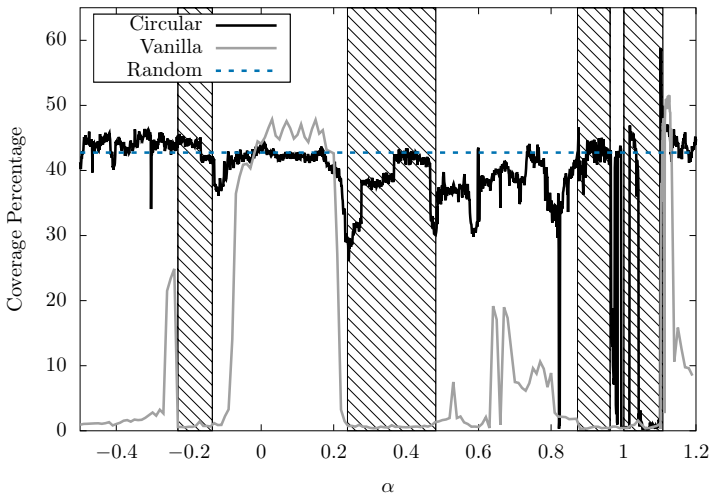


FIGURE 4.14: Performance of Rössler CHATs in comparison to the Random Walk on the small world topology (see Fig. 4.13 definitions).

4.4.2.3 Random Topology

Random Graph is one of the most used topology to capture the essence of the real-world graphs [122]. While, this topology should be the most well suited for the Random Walk due to its randomness, the increase of Coverage Percentage from the Circular CHAT can still be seen over the Random Walk in Fig. 4.15. This is due to the circulation strategy that reduce the number of times an agent revisits already discovered nodes. An improvement of 4% in coverage over Random Walk is reported at $\alpha = 1.123$ with Circular version. The Coverage Percentage around $\alpha = 0.2$ can be disregarded since it falls in the periodic category. As for the reason why the Circular CHAT can obtain such a high Coverage Percentage, it is due to the understanding of the behaviour of the system. By understanding the Rössler system, it is possible to strategically place neighbours to avoid repetition. With a uniform random function, this is not possible due to its randomness.

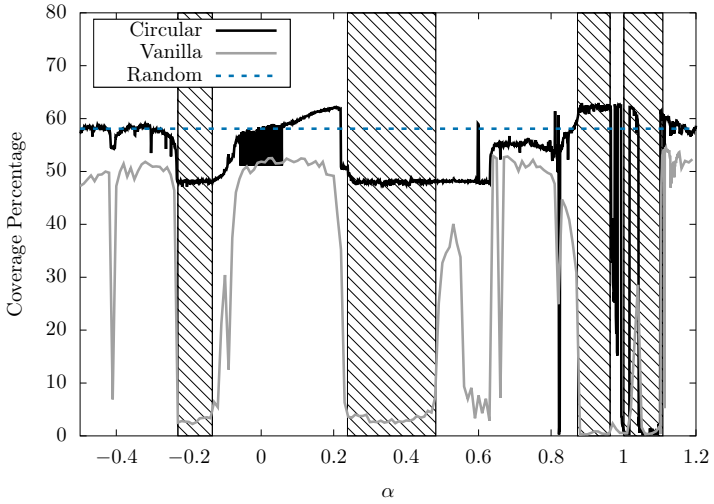


FIGURE 4.15: Performance of Rössler CHATs in comparison to the Random Walk on the random topology (see Fig. 4.13 definitions).

4.4.2.4 Grid Topology

Even though, the grid topology is also a structured topology like ring topology, the achieved result is quite different from the ring graphs. The Vanilla CHAT Coverage Percentage is lower in grid topology compared to the ring topology. It is suspected that this is due to the effect of the locality of neighbour. Since the nodes are tightly connected together and share same neighbours, it is easier to send an agent into a loop traversal. Several peaks (e.g., at $\alpha = -0.41$, 0.6, and 0.87) are spotted from the Circular version. They can be disregarded as they fall in the periodic period. However, at $\alpha = 1.124$, not only the Vanilla CHAT obtain the highest Coverage Percentage, the Circular CHAT also yields the Coverage Percentage of 27%which is higher than the Random Walk as shown in Fig. 4.16 due to the ability to avoid the revisiting of the previous nodes. At this alpha, Circular CHAT shows 10% improvement in Coverage Percentage over Random Walk and over 18% compared to the Vanilla version.This piece of result emphasises the need of an ability to study the system to enhance the performance of the algorithm even further which is not available on a uniform random function.

4.4.2.5 Power-Law Topology

Referring to Fig. 4.17, the Random Walk, and Vanilla version can cover a similar amount of nodes in the graph. On the other hand, the Circular CHAT shows a higher Coverage Percentage than other algorithms in this topology. A 2% improvement in Coverage Percentage at $\alpha = 1.124$. the Coverage Percentage around $\alpha = 0.2$

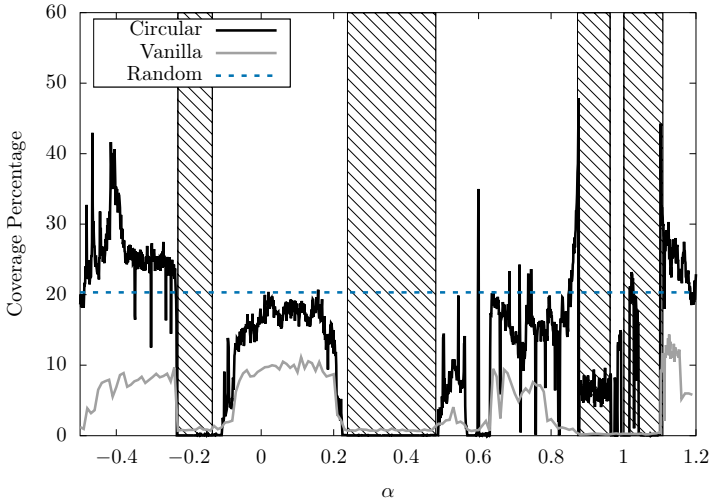


FIGURE 4.16: Performance of Rössler CHATs in comparison to the Random Walk on the grid topology (see Fig. 4.13 definitions).

are not taken into account as the gaps in the solution make it behave like a periodic solution. It is worth to note also that the result observed from Fig. 4.17 is similar to the result of CHATs shown in Fig. 4.15. This is because both have similar randomness properties, but with different rules of constructing the graphs.

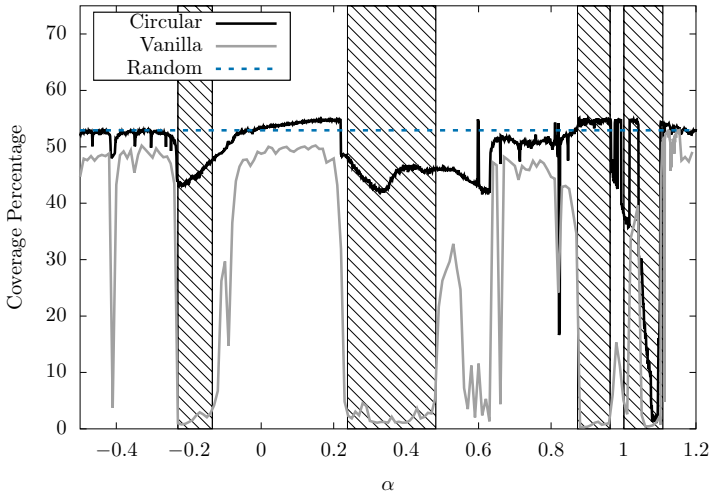


FIGURE 4.17: Performance of Rössler CHATs in comparison to the Random Walk on power-law topology (see Fig. 4.13 definitions).

From all presented figures, it is clear that with a combination of periodic solutions and Vanilla version from the chaotic system, the Coverage Percentage is low. Yet, the Circular version is able to get a good Coverage Percentage on all topologies. This initial result on small graphs enables us to have a rough estimation of what α should be used in a larger experiment. From the algorithm point of view, the result should not change much since an agent in CHAT algorithm only takes a local information from the node in the graph and does not consider a big picture of the graph at all. Therefore, those values of α which give out chaotic solutions that yield a high Coverage Percentage are tested in the larger graph experiment.

4.4.3 Results on Large Graph Instances

In this section, the best results for each algorithm on large graphs are presented. The α being used in this experiment are taken from the top three α that yield the best Coverage Percentage for each topology and

algorithm. Tab. 4.2 compare the results between CHAT (both versions) and Random Walk using Kruskal-Wallis test [111]. In this table, the result is expressed in a form of Coverage Percentage. Furthermore, the comparison between each algorithm on each topology is conducted using Wilcoxon test [112] in Tab. 4.3. Both Kruskal-Wallis and Wilcoxon tests are known for testing the difference in means of targeted populations. Kruskal-Wallis test is normally used when there are more than two populations to test, while Wilcoxon mainly test between two populations. For both experiments, a confidence interval of 95% is used. P-Value is used to decide whether accept or reject the null hypothesis as it reflect the probability of obtaining the observed results of a test, assuming that the null hypothesis is correct. Hence, by this definition and the confidence interval of 95%, if the observed p-value is less than 0.05, the null-hypothesis is rejected where the null hypothesis is there is no difference in the average performance among all observed algorithms.

Table 4.2: Coverage Percentage of CHAT and Random Walk. The first number is the Coverage Percentage. The second number behind ± is the standard deviation.

Topology	Random Walk	CHAT (Rössler Vanilla)	CHAT (Rössler Circular)	$p-$value
Ring	0.521 (0.155)	**60.471 (0.001)**	0.5645 (0.002)	$<$ 2.2e-16
Small World	42.219 (0.101)	**50.504 (0.087)**	45.956 (0.112)	$<$ 2.2e-16
Random	58.101 (0.003)	52.761 (0.061)	**60.218 (0.001)**	$<$ 2.2e-16
Grid	14.419 (3.734)	0.690 (0.606)	**18.052 (4.108)**	$<$ 2.2e-16
Power-Law	51.236 (0.032)	51.195 (0.035)	**54.045 (0.011)**	$<$ 2.2e-16

From Tab. 4.2, the result show that CHATs can perform significantly better than traditional Random Walk in all tested topologies. In ring topology, an improvement of 60% is obtained from Vanilla CHAT with the Rössler dynamic over Random Walk, while Circular version yields relatively the same Coverage Percentage as Random Walk. The next topology is small world topology where CHAT (Vanilla version with Rössler dynamic) having the highest Coverage Percentage, outperforms Random Walk by 8% in coverage and approximately, 2% of improvement in coverage over Random Walk is reported from Circular CHAT. Even with random topology, the significant improvement in coverage of 2% can be found and here Circular CHAT demonstrates that with a negligible constant memory, it yields the best Coverage Percentage. As in grid topology, the Circular version outperforms the Random Walk in term of Coverage Percentage significantly by 4%. As for power-law topology, CHAT (Circular version) still displays a significant improvement of 3% in coverage over Random Walk, while the Vanilla version shows a similar result to the random Walk.

It is clearly shown in the table that CHAT dominates the Random Walk in terms of Coverage Percentage in all tested topologies. This result also shows that they still follow the same trend even if there are some discrepancies of the Coverage Percentage which can be accounted by the random starting points, a difference in size and dimensions, plus the randomness during graphs construction. For example, from Fig. 4.13, Circular CHAT performs better than the Random Walk, and in large graphs, Circular CHAT still perform better. Even though the improvement is small, but it is still proved to be significant. Moreover, in the grid topology, the presented result in Tab. 4.2 and Fig. 4.16 are also similar, although, there is a drop in Coverage Percentage from a small experiment in a large graphs. However, that is to be expected given that more different grid dimensions are used to test the algorithm.

Table 4.3: Wilcoxon test between each algorithm where ▲, ▼ mean that the result is better, respectively worse, significantly The five symbols in a column represent a performance for each topology: Ring, Small World, Random, Grid and Power-Law respectively

Algorithm	Random Walk	CHAT (Rössler Vanilla)	CHAT (Circular)
Random Walk		▼▼▲▲▲	▼▼▼▼▼
CHAT (Vanilla)	▲▲▼▼▼		▲▲▼▼▼
CHAT (Circular)	▲▲▲▲▲	▼▼▲▲▲	

To further analyse the result, the matrix of Wilcoxon test is shown in Tab. 4.3. The Wilcoxon test indicates the significant difference in the means of two populations, so it can be determined which algorithm performs better by the statistical significance and their differences in Coverage Percentage. The upward triangles mean that the result is better significantly and the downward ones, otherwise. The circle represents that there is no statistical significance in the difference between each algorithm for a certain topology, hence two algorithms yield similar Coverage Percentage for a certain topology. The five symbols in a column represent a performance for each topology, ring, small world, random, grid and power-law respectively. From the Tab. 4.3, it can be observed that the Circular version of CHAT is better than Random Walk in all tested topologies and perform significantly better than the Vanilla version in random, grid and power-law topologies. From these results, Vanilla Rössler CHAT should be used for ring and small world topologies, while Circular version should be used for grid, random and power-law topologies. From both small and large graph experiments, all the results are collected conduct an in-depth analysis on α. The analysis concerns both the effect of α and algorithm optimisation.

4.4.4 Study of Chaotic Dynamics Impact

In this section, the analysis of performance for both periodic and chaotic solutions are presented. First return maps are used as an analysis tools to explain the system behaviours and how they affect CHAT performance. From the values obtained from the previous figures, the first return maps associated to each value of α can be obtained. The example of the first return maps for periodic solutions is shown in Fig. 4.18 which is the solution given by the system when $\alpha = 0.33$. From this first return map, it is self-explanatory as why it is periodic since there are only a few points on this map. In contrast, Fig. 4.19 and Fig. 4.22 display first return maps of the chaotic solution. The difference between the three figures is quite obvious since in the first return maps of chaotic solution, there are more points than the periodic one which means more possible values can be obtained from the system unlike, in Fig. 4.18 where only three points are presented. In the following sections, the effect of periodic and chaotic solutions on CHAT is discussed.

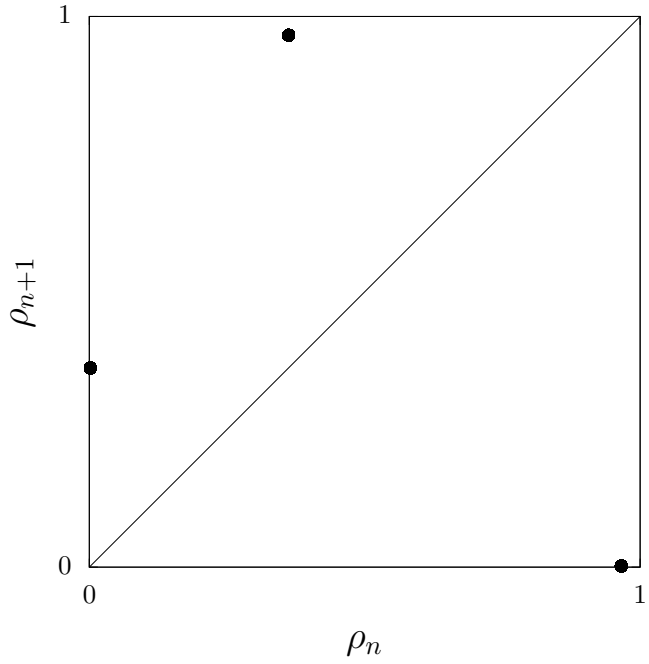


FIGURE 4.18: First Return Map to the Poincaré section for $\alpha = 0.33$.

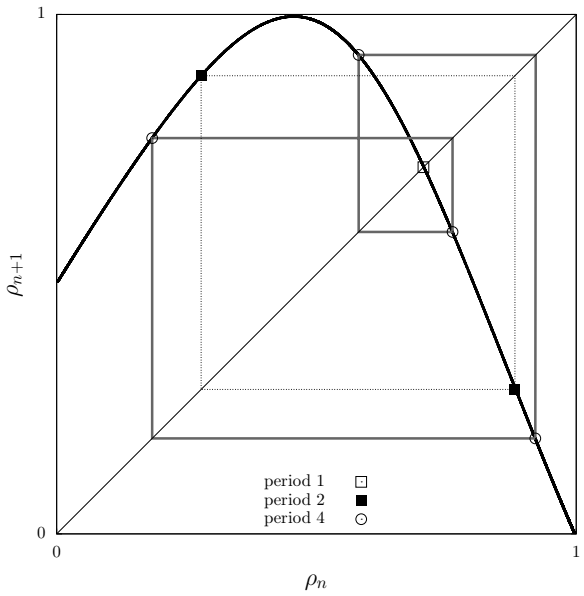


FIGURE 4.19: First Return Map to the Poincaré section for $\alpha = -0.26$. Periodic points are indicated to underline the structure of the first return map. These periodic points corresponds to orbits in the attractor. Only periodic points associated to orbits with a period lower than five are indicated.

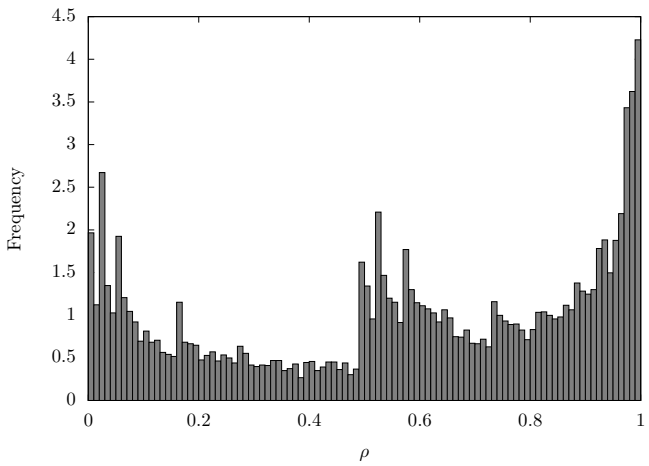


FIGURE 4.20: The density of iterates at $\alpha = -0.26$.

4.4.4.1 Effect of Periodic Solutions on CHAT

It can be seen in Fig. 4.13 and Fig. 4.16 when the system gives out periodic solutions, Circular CHAT performs really well and even better than the Random Walk. In fact, for a structured graph, there is not a need for randomness at all. The structured graphs can be traversed efficiently with simple systematic rules and a path history to prevent it from traveling in a circle. Therefore, this is not in the essence of our work since the focus of the research is not in the chaotic behaviour and the real world graphs are unstructured. Despite that, it is still important to describe this event to explain the cause of sharp rise in those result figures. As shown in Fig. 4.18, given only a few possible choices to move should not result in a high Coverage Percentage and that is true as shown in every tested topology with the Vanilla CHAT. From the analysis, the agent moves in circle due to limited choices in destination as shown in Fig. 4.21 and this can be seen in all small experiment figures. However, in Circular CHAT, by circulating the neighbours list, the algorithm becomes more like a port selection technique.

A port selection technique [170] is a traversal technique in a network. In a network graph, a server is considered to be a node and each connected port on a server is an edge in the graph. From the starting point in a graph,

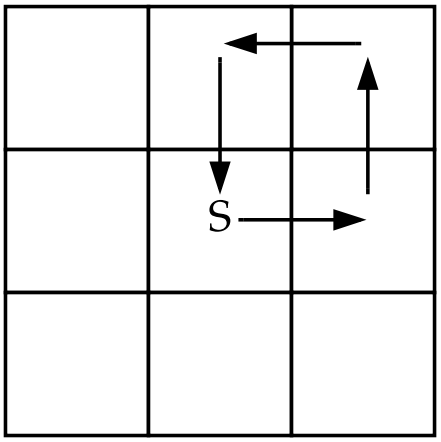


FIGURE 4.21: Periodic solution and traversal of an agent in a grid from the starting point (S).

the agent chooses to traverse on the first edge (the first connected port on the list) and that edge (or port) is then marked. On every node, the agent chooses the next unmarked edge (the next connected port on the list) to be traversed on and once the agent visits a node that has all edges marked, the marks are removed, and the same process starts over. The Circular CHAT and a periodic solution is similar to this technique. The circulation of neighbours in the list is similar to marking the edge and with few possible values from the periodic solution, the previous nodes are moved to the position that will not be selected. For example, there exists a period-1 solution. In that case, the previous node can be strategically moved to a position that will not be selected and promote the unvisited node to be chosen. Thus, the agent has less chance to visit the already visited nodes and higher chance to explore the graph. As in Fig. 4.18 which is a period-3 solution, it produces a higher Coverage Percentage in Grid and Ring topologies because these two topologies are structured and this port selection technique happens to be very efficient in a structured graph traversal. With this, the cause of high Coverage Percentage in periodic regions presented in figures in Section 4.4.2 is explained. Next is the discussion on how chaotic dynamics affect CHAT performance.

4.4.4.2 Effect of Chaotic Solutions on CHAT

To study the effect of chaotic dynamics thoroughly, the focus is on the Vanilla CHAT. This is because there is no modification being done on the Vanilla CHAT, hence it exhibits a pure chaotic behaviour. In all tested topologies, the periodic solutions fail to cover the graph efficiently compared to the chaotic solutions. To elaborate the cause of this result, considering a maze walking problem, there is a right hand rule where the agent always walk on the right side of the wall. With this rule, one is able to exit the maze. However, exploration is completely different. If only one rule is to be followed without any regards of the previous path. There is a chance that an agent will be trapped at some point. Hence, if only one rule is applied to traverse the graph, it would be the same as Breadth First Search (BFS) or Depth First Search (DFS). Coupling with the issue of no memory of the traversing path, an agent ends up being caught in a circle at some point which is the reason why, the relocation function is needed in BFS and DFS for traversing an unknown graph [126]. In Vanilla version, the neighbours list is deterministic. The order of neighbours of each node does not change over time. Using Fig. 4.18 as an example, there are only three points on the map. This is similar to having only three rules. Thus, at the fourth iteration, if the agent happens to revisit the starting node again, the same route will be repeated over. The chaotic solutions are different. In Fig. 4.19 and Fig. 4.22, there are more points on those two maps compared to Fig. 4.18. Each point on the map can be considered as a rule for the agent. With the chaotic solution, for a specific amount of time, the agent is given a set of rule (in this case, a pattern

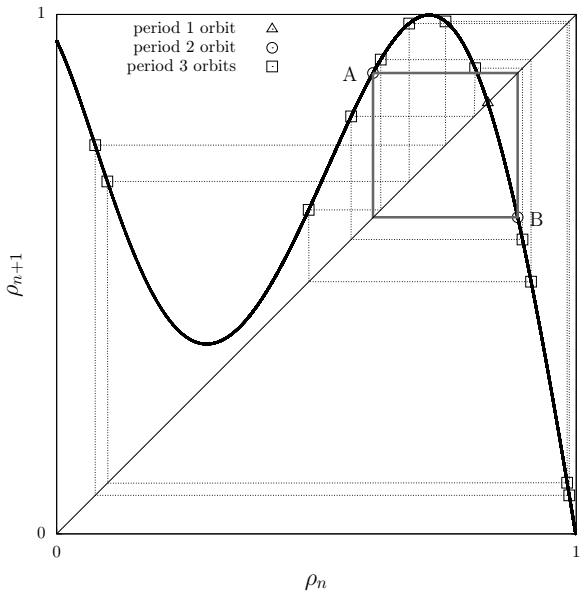


FIGURE 4.22: First Return Map to the Poincaré section for $\alpha = 1.124$. Periodic points represent periodic orbits in the three-dimensional space. Period 2 orbit is thus represented by the points $A = (x_a, y_a)$ and $B = (y_a, x_a)$ in the first return map. Only periodic points associated to orbits with a period lower than four are indicated.

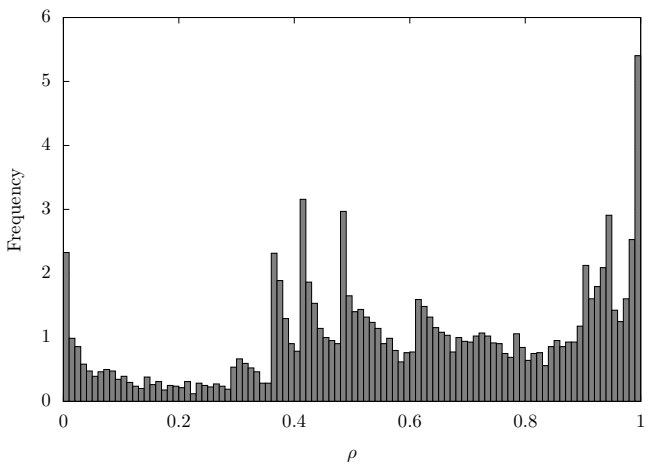


FIGURE 4.23: The density of iterates at $\alpha = 1.124$.

from the system) to traverse the area, but this rule changes over time to offer a breakthrough from potential circle without a need for a memory of the path. It can be further elaborated with the orbits in Fig. 4.19 and Fig. 4.22. There are periodic points in these two first return map that are temporarily reached and corresponds to orbits. These orbits are unstable, thus the periodic solution do not last. Only periodic points for low period orbit to detail the first return map dynamic are provided. Low period orbits are more often visited. But it can be observed that there are more points in the orbits in Fig. 4.22 which means more rules are available. In addition, the density of iterates (further read in [185]) are shown in Fig. 4.20 and Fig. 4.23. These two figures depict the density of values in the first return maps. It can be observed that there are more distinguished peaks in Fig. 4.23 than in Fig. 4.20. This difference shows that there are around four prominent choices (frequency above 1) if $\alpha = -0.26$ is used in CHAT. On the other hand, there are seven peaks in the first return map of $\alpha = 1.124$. This is the reason why a better Coverage Percentage is shown from the chaotic solution over the periodic solution in Vanilla CHAT on all tested topologies and also explain why a more chaotic behaviour can perform better than a standard one.

As the results are shown Figs. 4.13 – 4.17, it is clear that chaotic behaviour is indeed the key to higher Coverage

Percentage on the graphs for CHAT. However, even chaotic solutions are different as it can be observed that for different values of α. This is due to the complexity of the chaotic behaviours. In Fig. 4.19, there are two branches. From $\rho_n = [0, 0.45]$, this is the first branch and it is increasing. The second branch is from $\rho_n = (0.45, 1]$. While in Fig. 4.22, there are three branches. The first branch is a decreasing branch starting from $\rho_n = [0, 0.3]$. The second branch is from $\rho_n = (0.3, 0.7]$ and it is an increasing branch. The last branch is a decreasing branch and starting from $\rho_n = (0.7, 1]$. The higher number of branches in the first return map, the more complex the chaotic behaviour. Considering Fig. 4.19 and its given value at each iteration, the consisted patterns do not change abruptly. Some patterns even continue for a number of iterations as shown in Fig. 4.24. This is the same problem with the periodic solution. If the agent arrives at the node where the pattern first starts, it is bounded to continuously move in a circle until the next pattern comes. Therefore, with a more complex chaotic behaviour in the banded chaos area where the first return maps contain more than two branches, more patterns exist and can be changed abruptly leading to a wider coverage on a graph as can be seen in the result figures of each topology where $\alpha = 1.123$ and $\alpha = 1.124$ dominate other values of α in Coverage Percentage. When this complex chaotic behaviour is combined with the Circular strategy, it leads to even better Coverage Percentage since repetitions that come with patterns in chaotic dynamics are efficiently avoided.

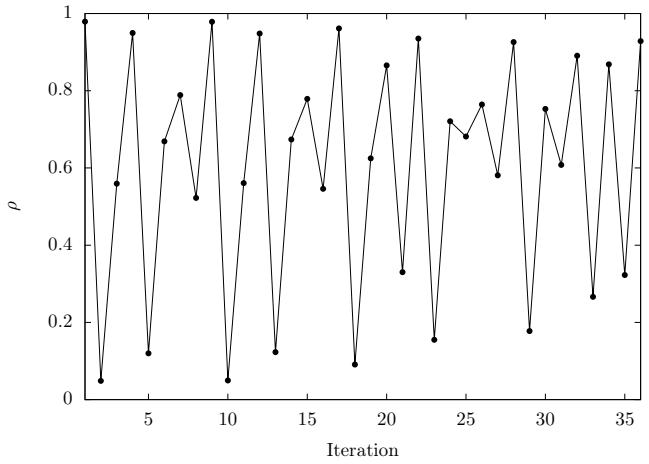


FIGURE 4.24: A plot of ρ against iteration.

4.5 Experimental Results: Lozi CHAT

This section presents the results of CHAT with the Lozi map and its comparison to CHAT with the Rössler system which showed superior performance than Random Walk in the previous section. This section contains three main parts. The first one presents the experimental setup. Then, the preliminary results on small graphs are presented in the second part. In the last part, the results on large graphs are presented together with their discussion.

4.5.1 Experimental Setup

The same test platform and libraries have been used to test CHAT with the Lozi map to ensure a fair comparison (cf. Section 4.4.1). The Lozi map is based on two parameters, a and b. These two parameters allow many variations, however it is not the main goal of in this work to explore them all. Therefore, two sets of parameters are used in these experiments, $a = 1.5, b = [-0.5, 0.47]$ and $a = [1.1 - 1.8], b = 0.1$ which were previously studied

TABLE 4.4: Experiments formulation. The best values of a and b obtained from the small graph experiments are used to explore large graphs. Selected combinations of a, b, and method are tested for 30 runs on each graph.

Topology	Small Graph Experiment	Large Graph Experiment	
Ring	$n = 2000$, $k = 10$, $p = 0$	$n = 1000000$, $k = 10$, $p = 0$	
		$a = 1.5$, $b = -0.44$	Vanilla
Small World	$n = 2000$, $k = 10$, $p = 0.01$	$n = 1000000$, $k = 10$, $p = 0.01$	
		$a = 1.5$, $b = -0.42$	Vanilla
Random	$n = 2000$, $k = 10$, $p = 1$	$n = 1000000$, $k = 10$, $p = 1$	
		$a = 1.78$, $b = 0.1$	Circular Last
Grid	dimensions: 5×500, 10×250, 20×125, 25×100 and 50×50	dimensions: $50 \times 20000, 100 \times 10000, 160 \times 6250, 200 \times 5000, 320 \times 3125, 400 \times 2500, 500 \times 2000, 625 \times 1600, 800 \times 1250$ and 1000×1000	
		$a = 1.43$, $b = 0.1$	Circular Last
Power-Law	$n = 2000$, $m = 10$, $k = 0.2$	$n = 1000000$, $m = 10$, $k = 0.2$	
		$a = 1.5$, $b = 0.41$	Circular Last

in [181]. The values of x_n scaled to $[0, 1]$ are used which is the same approach as [133]. Each algorithm is tested on each graph for 30 runs with different starting points. The experimental configuration is shown in Table 4.4 where the parameters for the large graph experiments are chosen from the configurations that yield the highest Coverage Percentage in the small graph experiment.

4.5.2 Results on Small Graphs Instances

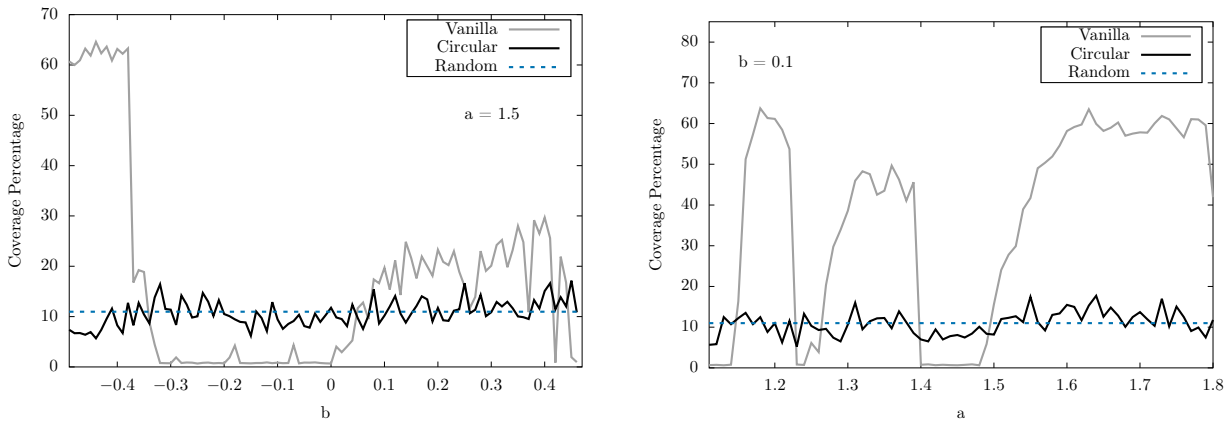
The results in this section serve as a screening process and help us in obtaining the best combination of parameters and algorithms for the large graphs experiments. All the result figures in this section present the Coverage Percentage on the $y - axis$. The x-axis represents either a or b. The obtained solutions from the Lozi map with the selected parameters are all chaotic, hence there is no shaded area in the figures. In all presented figures in this section, the word "Circular" refers to the Circular Last method. For simplicity, the Circular First method is not shown due to its inferior results.

4.5.2.1 Ring Topology

Both versions of Lozi CHAT are reported to have a superior Coverage Percentage than the Random Walk as shown in Fig. 4.25. The Vanilla version can achieve around 60% Coverage Percentage which is 50% higher than Random Walk. This result is also similar to the Rössler CHAT where the Vanilla version performs best on the ring topology. From the results, the Vanilla method with parameters $a = 1.5$ and $b = -0.44$ is chosen to be used in the large graphs experiment since it provided the highest Coverage Percentage with almost 63%.

4.5.2.2 Small World Topology

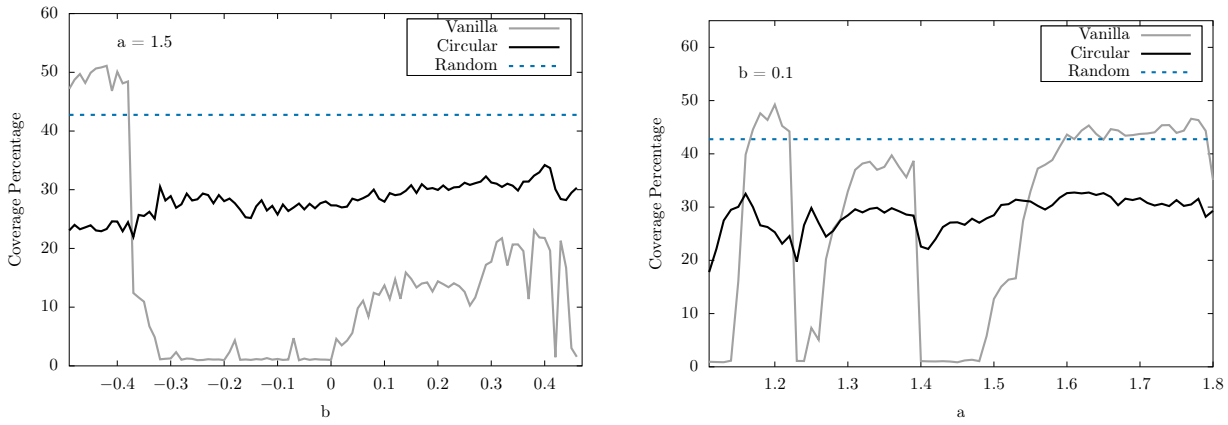
On the small world topology, the Vanilla version also yields the highest Coverage Percentage at around 50% as shown in Fig. 4.26. The Circular version, on the other hand, yields lower Coverage Percentage than the Vanilla version and the Random Walk. From this result, the Lozi map also shares the same trait with the Rössler system in that the Coverage Percentage decreases when the graph becomes less structured. The combination of $a = 1.5$ and $b = -0.42$ is selected to be used in the large graph experiment due to its superior performance.



(a) $a = 1.5$ and $b = [-0.5, 0.47]$.

(b) $a = [1.1, 1.8]$ and $b = 0.1$.

FIGURE 4.25: Performance of Lozi CHAT: both are compared to the Random Walk on the ring topology. The black solid line represents the results of the Circular (Last) CHAT and the grey line the result of Vanilla CHAT. The dashed line represents the result of the Random Walk. The Circular First method is not shown due to its inferior results compared to the Circular Last method.



(a) $a = 1.5$ and $b = [-0.5, 0.47]$.

(b) $a = [1.1, 1.8]$ and $b = 0.1$.

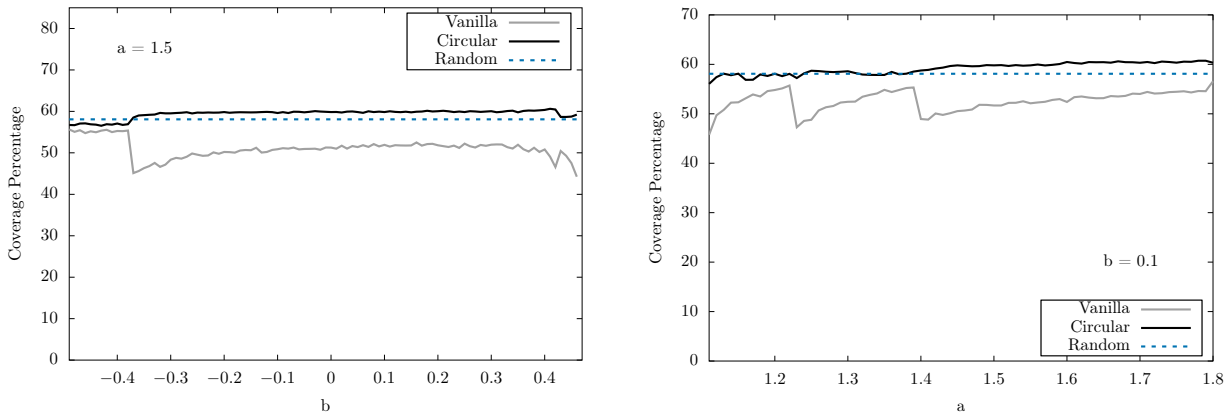
FIGURE 4.26: Performance of Lozi CHAT in comparison to the Random Walk on small world topology (see Fig. 4.25 definitions).

4.5.2.3 Random Topology

Fig. 4.27 shows that the Circular version of Lozi CHAT can achieve a higher Coverage Percentage than the Random Walk with a 2% improvement. In contrast, the Vanilla version performs worse than both with the peak Coverage Percentage at 56% only. With this result, $a = 1.78$ and $b = 0.1$ are selected for the large graph experiment for the random topology.

4.5.2.4 Grid Topology

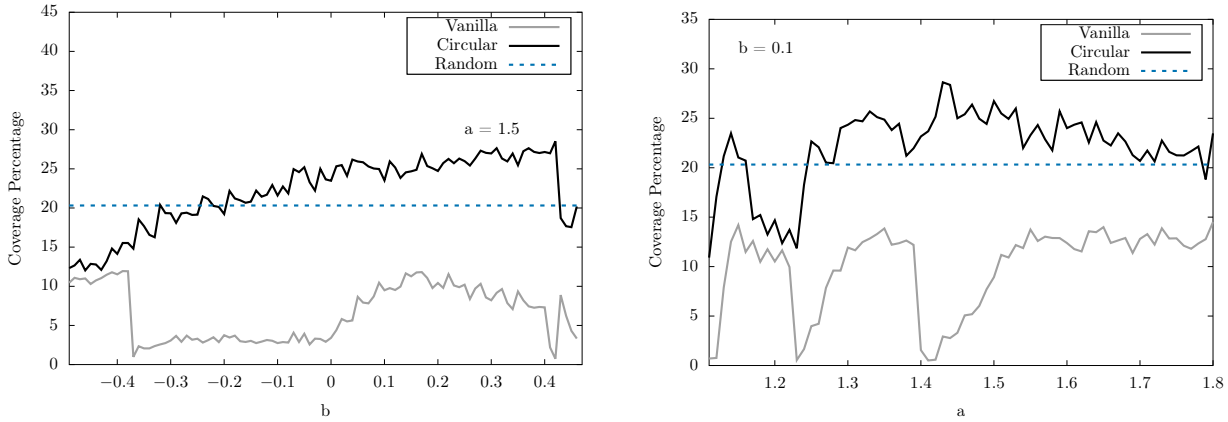
In the grid topology, the Random Walk obtains 20% coverage on average which is better than the Vanilla Lozi CHAT (refer to Fig. 4.28) that yields around 14% at most. On the other hand, the Circular version outperforms the Random Walk by over 8% in coverage at $a = 1.43$ and $b = 0.1$, hence these parameters are selected.



(a) $a = 1.5$ and $b = [-0.5, 0.47]$.

(b) $a = [1.1, 1.8]$ and $b = 0.1$.

FIGURE 4.27: Performance of Lozi CHATs in comparison to the Random Walk on random topology (see Fig. 4.25 definitions).



(a) $a = 1.5$ and $b = [-0.5, 0.47]$.

(b) $a = [1.1, 1.8]$ and $b = 0.1$.

FIGURE 4.28: Performance of Lozi CHATs in comparison to the Random Walk on grid topology (see Fig. 4.25 definitions).

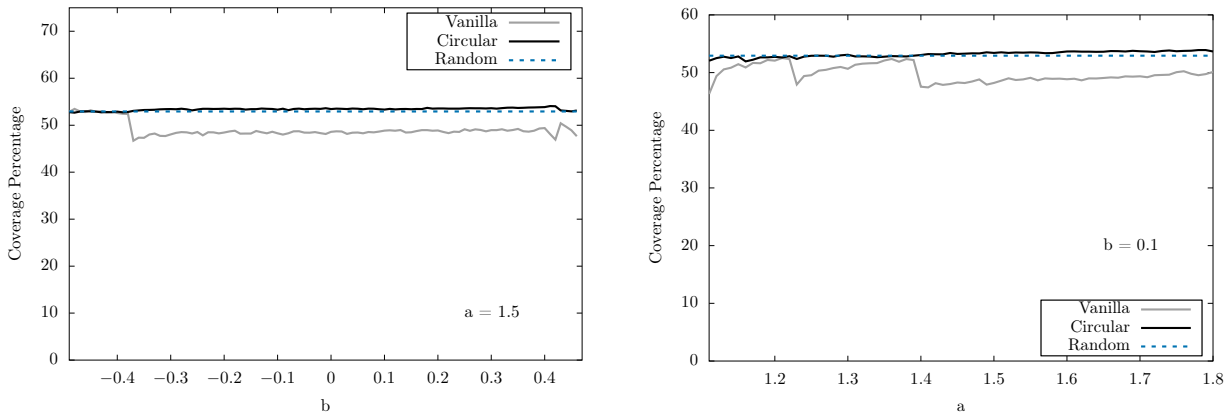
4.5.2.5 Power-Law Topology

The performance trend in Fig. 4.29 is similar to the trend in the random topology (Fig. 4.27). Even though, the Coverage Percentage of the Random Walk is quite comparable to the Circular version, at $a = 1.5$ and $b = 0.41$, a 2% improvement in Coverage Percentage from the Circular version can be observed. From all these results, it seems that once the graph becomes structureless, the importance of Vanilla version for Lozi CHAT is diminished as it cannot be compared with the Random Walk.

4.5.3 Results on Large Graphs Instances

In this section, the selected methods and parameters that yield the highest Coverage Percentage from each topology are utilised and tested on a $1,000,000$ nodes graph. This graphs are the same ones that are used to test the Rössler CHAT in Section 4.4.3. In that section, it has also been concluded that Rössler CHAT yields a higher Coverage Percentage than the Random Walk for all topologies, therefore, Only the Coverage Percentage between Rössler and Lozi CHAT as shown in Tab. 4.5 using Wilcoxon test [112] are compared.

From Tab. 4.5, Lozi CHAT is significantly better than Rössler CHAT on the ring and random topologies. Lozi CHAT yields 2% more Coverage Percentage on the ring topology, while an improvement on the random



(a) $a = 1.5$ and $b = [-0.5, 0.47]$.

(b) $a = [1.1, 1.8]$ and $b = 0.1$.

FIGURE 4.29: Performance of Lozi CHATs in comparison to the Random Walk on power-law topology (see Fig. 4.25 definitions).

TABLE 4.5: Best performance of Rössler CHAT and Lozi CHAT for each graph topology. The first number is the Coverage Percentage and the number behind ± is the standard deviation. For Rössler and Lozi CHAT configurations, please refer to Tab. 4.1 and Tab. 4.4 respectively.

Topology	Best Rössler CHAT	Best Lozi CHAT	$p-$value
Ring	60.471 (0.001)	**62.475 (0.001)**	< 2.2e-16
Small World	**50.504 (0.087)**	49.739 (0.134)	< 2.2e-16
Random	60.218 (0.001)	**60.558 (0.034)**	< 2.2e-16
Grid	18.052 (4.108)	18.892 (6.548)	0.8704
Power-Law	**54.045 (0.011)**	53.344 (0.036)	< 2.2e-16

topology is minor. As for Rössler CHAT, it outperforms Lozi CHAT on small world and power-law topologies, by almost 1% in Coverage Percentage. On the random topology, the difference in Coverage Percentage is marginal. Lastly, on the grid topology, both Rössler and Lozi CHAT are comparable in terms of Coverage Percentage. The standard deviation might be large, but this trend also persists in the Random Walk as well, due to the variation of the shape of the tested grids.

4.5.4 Study of Chaotic Dynamics Impact

From the study of the Rössler system in Section. 4.4.4.2, the effect of density of iterates in the Coverage Percentage is observed. The solutions that have high fluctuation in ρ tend to yield higher Coverage Percentage than the ones that do not. Therefore, the density of iterates for $a = 1.5$, $b = -0.44$ and $a = 1.78$, $b = 0.1$ are studied as they yield the highest Coverage Percentage for different versions of CHAT. The x-axis in the plot is ρ or normalised solution from the map and the y-axis is the frequency of the ρ. In Fig 4.30, the density of iterates for $a = 1.5$ and $b = -0.44$ is shown. This set of parameters yield a high Coverage Percentage in ring and small world topology. From the plot, there are many peaks above one (in frequency) which means an agent has more variations for choosing the destination. However, these peaks are close to each other and sometimes represent the same choice which is different from the Rössler system with $\alpha = 1.124$ in Fig .4.23 that has fewer peaks and these peaks are not clustered together leading to the higher variations in choices.

Another example is the density plot from the Lozi map with $a = 1.78$ and $b = 0.1$. This set of parameter works really well with Circular Last method. The reason becomes clear with Fig. 4.31. The majority of ρ is in $[0, 0.8]$ as many peaks can be observed in that area. This means when the previous node is shifted to the end of the neighbour list, it has less chance to be visited again. With this result, the importance of the Circular approach

is emphasised and it demonstrates that chaotic systems provide more promising results for graph exploration than the Random Walk.

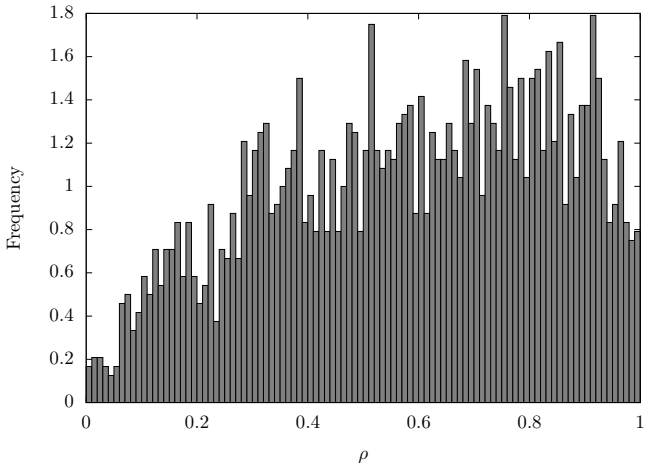


FIGURE 4.30: The density of iterates of Lozi map for parameter values $a = 1.5$ and $b = -0.44$.

From all the CHAT results, it shows that CHAT can outperform the Random Walk in terms of node discovery (Coverage Percentage). Based on the results in this section and the previous section, Lozi map should be used when traversing the ring and random graphs. While Rössler system should be used to traverse small world and power-law graphs. As for grid, both can be used, but Rössler system might be a better choice due to the lower standard deviation. The next section proposes to explore further the experimental results in terms of Mean Time Coverage.

4.6 Experimental Results: Mean Time Coverage

This section reports the results in terms of Mean Time Coverage (Eq. (4.13)) which is the ratio of number of discovered nodes over the algorithm's execution time. In CHAT and Random Walk, there are fundamentally two processes. The first one is to prepare the sequence of values for graph traversal (ρ in CHAT and random number in a range of [0,1] in Random Walk). The second part is to execute the traversal algorithm. For the Mean Time Coverage metric in this work, the focus is on the second part where the algorithm is executed instead of the first part due to the fact that the sequences can be pre-computed (i.e., generated offline) and have thus no effect on the second part. However, the sequence generation time is also reported in this section to complete the whole study.

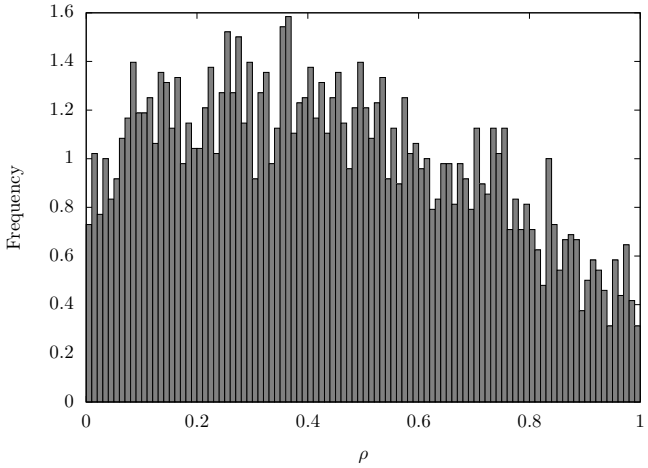


FIGURE 4.31: The density of iterates of Lozi map for parameters values $a = 1.78$ and $b = 0.1$.

4.6.1 Test Platform

Our implementation is in Python 2.7. The essential libraries are NetworkX 1.11, numpy 1.11.1 (for pseudo-random number generator (PRNG) with Marsenne Twister algorithm [186]) and scipy 0.17.1. The test is performed on a single core of an Intel Xeon L5640 (2.26 GHz) with 4 GB memory available from the High Performance Computing (HPC) platform of the University of Luxembourg [96]. Every algorithm (with reported parameters that yield the best result) is tested on five topologies with the small graphs as shown in the small graph experiment setup in Tab.4.4 for 30 runs on each graph. The measured CPU time for online exploration and number of discovered nodes is the average of 30 runs. To accommodate with 2,500 node graph, the CPU time for generating a 2,500-value and 1,000,000-values sequences for Rösssler system, Lozi map and uniform random function (using the numpy library) are measured. The average CPU time for a sequence generation process is calculated from 30 runs as well. The $\alpha = 1.124$ and $a = 1.5$, $b = -0.42$ are selected as representatives for generating the ρ sequences for Rössler system and Lozi map respectively since the computation time for these parameters is the highest (upper bound) in their respective systems.

4.6.2 Results on Algorithm Computational Performance

The Mean Time Coverage for each algorithm is shown in Tab. 4.6. These numbers reflect the number of nodes that can be discovered in one second by the algorithms. The Vanilla Lozi CHAT obtains the best result on the ring topology. On the small world topology, Vanilla Rössler CHAT yields the highest value with 8388 nodes in one second. The Circular First Rössler then dominates other algorithms on all other topologies. These values are compliant with the previous results (Tab. 4.3) since those algorithms provide a higher Coverage Percentage. However the outcomes are different on the random and grid topologies. It is shown in Tab. 4.5 that Circular Last Lozi CHAT yields a higher Coverage Percentage on the random topology and there is no significance in the difference between the latter and the Circular First Rössler CHAT on the grid topology. Yet, the Mean Time Coverage suggests otherwise. This is due to the execution time of the Circular Last methods which is longer than its counterpart leading to a lower efficiency. However, in any case, the efficiency of CHAT is higher than Random Walk on all tested topologies.

TABLE 4.6: Mean Time Coverage for each Algorithm in node/second. The CPU time and number of discovered nodes are calculated from 30 runs. The best results are in bold.

Algorithm	Ring	Small World	Random	Grid	Power-Law
Vanilla Rössler	10304	**8388**	8881	2447	6038
Circular First Rössler	2833	7565	**9677**	**5319**	**6625**
Circular Last Rössler	2343	7120	6645	2430	5306
Vanilla Lozi	**10641**	8299	9265	2534	6050
Circular First Lozi	8500	7000	9193	3191	6500
Circular Last Lozi	2656	5379	9375	4861	5510
Random Walk	1854	7077	9604	3602	6025

Considering CPU time, the exploration of the graph with our algorithm is quite comparable, however the bottleneck lies in the sequence generation time. The CPU time for generating the value sequences is shown

TABLE 4.7: CPU time for generating the value sequence for Rössler system ($\alpha = 1.124$), Lozi map ($a = 1.5$, b = -0.44), and Marsenne Twister algorithm (in second). These CPU times are averaged from 30 runs.

Chaotic dynamics or PRNG algorithm	2,500 values (second)	1,000,000 values (second)
Rössler System	20.325	N/A
Lozi Map	5.9e-5	0.0132
Marsenne Twister Algorithm	9.084e-5	0.0201

in Tab. 4.7. It can be observed that the Rössler system takes the longest time to compute with around 20 seconds due to the solving of "Ordinary Differential Equation" (ODE) and "Runge-Kutta method (fourth order)" (RK4). The Lozi map is much faster than the Rössler system and is comparable to Mersenne Twister algorithm. As for the chaotic dynamics computation of Rössler system, one possibility to optimise further the computation time is to consider an FPGA implementation [187]. However this is not the objective of this paper and will be considered for future work. Regardless of the computing time for each system, the sequence can always be pre-computed, and hence, can be disregarded in the real scenarios. The reported results in this work indicate that CHAT can perform better than the Random Walk in all tested topologies in terms of Coverage Percentage and Mean Time Coverage. The Lozi map is also more suitable to use with CHAT than the Rössler system due to the significantly lower computation time to obtain discrete chaotic dynamics (number sequence generation).

4.7 Experimental Results: Summary

Considering only the dynamical system performance without computing time, CHAT with Lozi map can cover the highest number of nodes in the ring and random topologies. On the other hand, Rössler CHAT shows exceptional results in small world and power-law topologies. Both chaotic systems yield comparable Coverage Percentage on the grid topology. Apart from the Coverage Percentage which defines the number of discovered nodes of an algorithm, the Mean Time Coverage metric is also proposed to measure the efficiency (against time) of an algorithm. When both metrics are considered in conjunction with the computation time of the Lozi map, Lozi CHAT is even better than the Rössler CHAT due to its comparable Coverage Percentage and significantly lower computation time. Furthermore, this research utilised a bifurcation diagram of a Rössler system to further improve the performance of CHAT. The bifurcation diagram is divided into parts to capture the essence of the Rössler system including the banded chaos attractors. It is also found that the complexity of the banded chaos contributes to the increased performance of CHAT. The proposed methodology using this bifurcation diagram enables us to test nonequivalent chaotic mechanisms for a given optimisation algorithm and other optimisation problems as well. This partition method can also be used with the Lozi map to avoid the gap in the value sequence which is unwanted in our use case. However, the topological analysis cannot be performed to compare the chaotic mechanisms of the Lozi map. According to the previous statement, Tab. 4.8 summarises the best parameters and CHAT algorithms on each topology to use with the Lozi map.

Table 4.8: Best parameters for Lozi CHAT for each tested graph topology

Topology	Lozi map parameters	Algorithm
Ring	$a = 1.5$, $b = -0.44$	Vanilla
Small World	$a = 1.5$, $b = -0.42$	Vanilla
Random	$a = 1.78$, $b = 0.1$	Circular Last
Grid	$a = 1.43$, $b = 0.1$	Circular Last
Power-Law	$a = 1.5$, $b = 0.41$	Circular Last

Chapter 5

Conclusions

This chapter summarises all essential content in this research in order to gain a complete big picture which encompasses our intentions and contributions to the industrial and academic communities. The chapter is separated into two sections. The first section iterates the findings and restates the contributions of this research. The second section discusses the future research directions that can be explored from this research.

5.1 Summary

The main focus of this research is the graph algorithms. The research accomplishes this goal by demonstrating two applications of the algorithms in two different real-world problems, one being in smart mobility planning and another one being in large graph discovery.

In smart mobility, this research contributes substantially in fleet placement planning in round-trip carsharing service. Traditionally, this planning process is performed manually by experts. This research proposes a system which automatises this process, and at the same time, enhances the accuracy and feasibility of the obtained results. There are three main contributions. First, two optimisation models aiming at different phases of planning according to the carsharing company needs are proposed. Second, a hybrid metaheuristic algorithm is proposed. The algorithm is a combination of NSGA-III and Pareto Local Search (PLS) that utilises our novel local search operator, Extensible Neighbourhood Search (ENS).. And thirdly, a benchmark suite is proposed to facilitate the future research. This application is separated into three phases, (1) Fleet placement problem and general algorithms screening, (2) Vehicle placement problem and metaheuristic algorithms screening and (3) Hybridisation and the utilisation of local knowledge.

The first phase presents the optimisation model of the fleet placement problem. The problem focuses on finding the best location of carsharing stations in a city. There are two objectives in this phase, (1) maximise user coverage and (2) minimise the global walking distance. Optimizing those two objectives is proven to be NP-Hard. Along with the optimisation model, three solvers are evaluated, each representing three categories, (1) PolySCIP (exact solver), (2) heuristics from [17] and (3) NSGA-II (metaheuristic). Three indicators are used to measure the quality of the solutions, IGD, spread and hypervolume. The findings show that PolySCIP cannot solve a relatively small instance (1000×1000) in an acceptable time (more than 18 days and no result is found). On the other hand, heuristic yields good solutions, but at a high computational cost (17 hours on an instance that contains about 17,000 potential locations). If an approximated Pareto front is needed, the process can take even longer depending on the granularity of the weight vector. Finally, NSGA-II is shown to be the most

efficient. The execution time is the fastest (only 26 minutes) and in just one execution, an approximated Pareto front can be obtained. In addition, solutions from heuristics can be used as seeds (or initial population) to obtain even higher quality solution through the search process of NSGA-II. The comparison between manual allocation, heuristics and metaheuristics is also presented. It is proven that heuristic and metaheuristics yield higher quality solutions and are able to provide interesting prospects for the decision maker. Lastly, in this phase, the effectiveness of the optimisation objectives is clarified. While the user coverage maximisation is proven to be useful, the global walking distance shows otherwise. In fact, the global walking distance objective provides little to no benefit contradicting the first assumption from the carsharing company. The minimised distance can only be translated to two to three minutes less walking time, while the user coverage suffers greatly. Therefore, we recommended the carsharing company to use metaheuristic algorithms for its efficiency and suggested ignoring the global walking distance objective as it is not worthwhile sacrificing the user coverage.

In the second phase, a novel vehicle placement problem (VPP) is introduced. VPP has three optimisation objectives. The first objective remains user coverage maximisation. However, the global walking distance objective from FPP has been discarded and is replaced by the vehicle number minimisation. This second objective is designed to help the company estimate the number of vehicles in the fleet at the initial deployment of a business in a new city. The third objective is the maximisation of public transport coverage. The model is also proven to be NP-Hard. Therefore, based on the results of the first phase, different state-of-the-art metaheuristic algorithms to are evaluated on VPP. In the experiment, three algorithms are considered, SPEA-II, NSGA-II and NSGA-III. Each algorithm is tested on 10 synthetic (50000-node) and two real-world instances (Munich and Hamburg). Through the statistical and convergence analyses, the difference in term diversity is that SPEA-II and NSGA-III dominate NSGA-II in all instances. However, with IGD and hypervolume indicators, SPEA-II has significantly worse performance than the other two algorithms in these two departments. Meanwhile, NSGA-II and NSGA-III have comparable performance in IGD and hypervolume. And since NSGA-III dominates NSGA-II in term of diversity, NSGA-III is selected as the base algorithm for hybridisation which is the next step in the third phase.

The third phase explains the proposed hybrid metaheuristic algorithm. The hybrid algorithm, NGAP, is the combination of NSGA-III and Pareto Local Search (PLS). They work co-operatively using a low-level teamwork hybrid strategy. Problem specific knowledge is used to design a local search operator called Extensible Neighbourhood Search (ENS). ENS relies on the neighbourhood concept in graph theory for the search. The main idea is to search locally for non-dominating solutions instead of searching blindly in the solution space without taking any advantage from the nature of a graph. Two strategies for NGAP are proposed, the first strategy is to stop the local search operator once the first non-dominating solution is found, while the second strategy continues the local search operator until a round limit is reached. The two variations of NGAP are compared against the original NSGA-III on 24 instances. Under the same execution time, NGAP outperforms the original NSGA-III in IGD and hypervolume in almost all instances. Second strategy is also found to be more efficient than the first strategy on larger instances as well. Therefore, from the series of experiments, NGAP with second local search strategy strategy is recommended for the application since it is more efficient (in term of IGD and hypervolume) than the original NSGA-III and for a similar execution time.

After various experiments in smart mobility using graphs, the next step is very large real-world graphs. These graphs are dynamic and practically unknown (not enough or almost no information given). These graphs are also ones that contain valuable insights, but those insights need to be extracted. However, the analysis cannot be performed if the graph is not yet discovered (or traversed). Conventional state-of-the-art graph traversal algorithms cannot be used on these graphs without complex modification and memory usage. Therefore, a novel memoryless graph traversal algorithm based on chaos theory called Chaotic Traversal (CHAT) is proposed. This algorithm marks the first application of chaos theory in the graph traversal field.

There are two chaotic attractors involved in the experiment, Lozi and Rössler attractors. Five graph topologies, namely, ring, small world, random, grid and power-law are used as experimental instances. The obtained results illustrate that CHAT outperforms the state-of-the-art memoryless Random Walk on all experimental instances, whether in coverage percentage or mean time coverage. The suitable attractor for CHAT to be used on each topology is also suggested.

In summary, this thesis proposes two novel graph algorithms in real-world scenarios. The theoretical contributions consist in the optimisation models for smart mobility planning, the hybrid algorithm and problem specific local search operator, the benchmark suite for further research, and finally the novel memoryless graph traversal algorithm based on chaos theory. In addition, the research also contributes automating a process and providing optimised solutions for a problem that is tackled manually in the industrial sector that is normally performed manually. These contributions are significant to graph utilisation in real-world problems and research to industry conversion which benefit both sectors.

5.2 Future Work

Practiced deployment, utilisation of technology, and betterment of society as a whole are the setting goals since the beginning of this research work. With this lead, there are several possibilities to extend this research. This chapter discusses future directions in smart mobility first and then the large graph discovery.

In smart mobility, the intention is to add other realistic features to the model. These possible enhancements will not only make the optimisation models even more realistic, but also will extend the focus of the model from initial deployment to fleet monitoring and management system. To achieve these goals, three possible enhancements concerning problem instance, optimisation model and solving algorithm are discussed here.

Currently, we distribute users in residential buildings. The process follows a uniform distribution on a district basis. The method works well as we are able to estimate roughly where the residential areas are. However one enhancement might improve the accuracy to pinpoint potential user locations even further is to distribute users according to the size and type of the building [188, 189]. For example, there can be more inhabitants in a single apartment complex than a single household. With the resident estimation model that take the size and type of the residence into consideration, the problem instance will be even more accurate to the real city.

The next enhancements concern the optimisation model. The current FPP and VPP deal with the initial fleet planning where the estimated demand is static. However, in the real-world, dynamicity is its nature. Time and days can affect the demand of car booking [81], while, a fleet expansion is also another factor for fleet planning. The extended model can be formulated to consider both varying demands, and the growing numbers of vehicles in the fleet. It is also possible that an even more robust planning that is less sensitive to changes and maintain its peak performance at all time can be formulated [190]. Another enhancement is the introduction of new objectives and discarding old ones as the service progresses into the later phase with more constraints and considerations. Machine learning might be used for simulation (from collected usage data) to aid the vehicle usage maximisation instead of the user coverage.

The third direction is about algorithms. As already mentioned in Chapter 3, the variable length encoding is still an open problem [98]. It is very interesting to look deeper into this area and evaluate how encoding can affect the results in our problem. Not to mention that a new encoding that involves more unique data from the problem might be able to lead to better solutions or affect the convergence of the algorithm. It might also be possible to come up with a problem specific recombination operator to improve the quality of the solutions even further.

Next is the future direction for the large graph discovery. This topic is ever changing due to the rate that the data are produced and their dynamicity. One aspect that can be addressed is, indeed, to extend CHAT to be able to adapt to the dynamicity of the real-world graph and perform the online discovery. Another aspect which merits further exploration is to study more chaotic attractors since each system has different characteristics. Therefore, the research aims to further improve the current algorithm into a multi-agent system which can employ several chaotic systems at once and analyse the result through templates and subtemplates [136]. With the reported performance of the Lozi map being comparable to the Rössler system and taking less time to solve the system, it may be beneficial to further study attractors similar to Lozi in order to extract similar continuous chaotic system behaviours out of the discrete systems. The work presented in [191] is a good place to consider in this regard. Moreover, some recent tools developed to obtain chaotic flow with suspension of the map (including the template of the chaotic dynamics) can be used to explore the capabilities of the Lozi map [183, 184]. All of these findings and prospects demonstrate the capability to shorten the computation time of the chaotic dynamics and preserve the same coverage percentage in the algorithm. These challenges also open many possibilities in experimenting with various existing chaotic systems. The ultimate goal would be the classification on selecting suitable attractors for each graph topology to perform an efficient traversal on the large unknown graphs. This might even lead to the determination of the topology of the unknown graph based on the chaotic attractors.

Appendix A

Preliminary Results for State of the Art Algorithms

In this section, we present the statistical and convergence analyses of three state-of-the-art algorithms in the rest of the instances that are not shown in the main content of the thesis. These algorithms are Strength Pareto Evolutionary Algorithm II (SPEA-II), Non-dominated Sorting Genetic Algorithm II (NSGA-II) and Reference Point based Non-dominated Sorting Genetic Algorithm (NSGA-III). There are three performance indicators, namely, inverted generational distance (IGD), spread and hypervolume.

A.1 Statistical Results

In the following, the results reflecting each indicator are compiled into tables. As IGD and spread need a reference solution set, we build it by consolidating the approximated fronts from 30 runs of all algorithms into one reference set for each instance. Then, we employ Kruskal-Wallis test and Wilcoxon test for statistical analysis with the confidence interval of 95%.

A.1.1 Results for 5000-node Instances

The followings present the statistical analysis for 5000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

TABLE A.1: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
5000_1	0.156 (0.018)	0.221 (0.028)	0.227 (0.028)	2.43e-12
5000_2	0.179 (0.025)	0.190 (0.025)	0.207 (0.024)	3.68e-5
5000_3	0.174 (0.021)	0.202 (0.023)	0.188 (0.021)	5.06e-05
5000_4	0.174 (0.020)	0.212 (0.024)	0.186 (0.022)	5.13e-08
5000_5	0.192 (0.020)	0.210 (0.023)	0.181 (0.019)	2.12e-05
5000_6	0.166 (0.015)	0.203 (0.020)	0.179 (0.017)	2.45e-09
5000_7	0.188 (0.030)	0.194 (0.024)	0.184 (0.029)	3.50e-01
5000_8	0.178 (0.020)	0.189 (0.020)	0.177 (0.025)	3.00e-02
5000_9	0.169 (0.018)	0.204 (0.025)	0.183 (0.024)	4.67e-07
5000_10	0.156 (0.017)	0.200 (0.022)	0.176 (0.024)	3.31e-09

TABLE A.2: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
5000_1	0.555 (0.042)	0.605 (0.030)	0.591 (0.046)	4.27e-05
5000_2	0.572 (0.035)	0.630 (0.043)	0.601 (0.051)	1.40e-05
5000_3	0.560 (0.038)	0.620 (0.041)	0.606 (0.044)	2.38e-06
5000_4	0.568 (0.044)	0.626 (0.049)	0.617 (0.050)	4.48e-05
5000_5	0.584 (0.038)	0.623 (0.035)	0.605 (0.033)	5.59e-04
5000_6	0.571 (0.039)	0.619 (0.049)	0.595 (0.038)	6.63e-4
5000_7	0.573 (0.045)	0.621 (0.032)	0.601 (0.032)	1.05e-04
5000_8	0.580 (0.026)	0.616 (0.049)	0.601 (0.048)	4.18e-03
5000_9	0.586 (0.037)	0.611 (0.041)	0.595 (0.051)	8.66e-03
5000_10	0.568 (0.040)	0.623 (0.041)	0.610 (0.049)	1.80e-05

TABLE A.3: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
5000_1	0.397 (0.011)	0.377 (0.014)	0.366 (0.015)	1.21e-10
5000_2	0.368 (0.013)	0.371 (0.013)	0.355 (0.011)	5.85e-05
5000_3	0.412 (0.011)	0.405 (0.012)	0.388 (0.020)	1.68e-06
5000_4	0.381 (0.012)	0.371 (0.013)	0.373 (0.015)	4.95e-03
5000_5	0.376 (0.010)	0.378 (0.014)	0.381 (0.012)	3.50e-01
5000_6	0.375 (0.009)	0.368 (0.013)	0.365 (0.011)	3.81e-03
5000_7	0.407 (0.013)	0.412 (0.012)	0.407 (0.017)	5.30e-01
5000_8	0.370 (0.011)	0.372 (0.011)	0.370 (0.016)	7.10e-01
5000_9	0.383 (0.010)	0.376 (0.010)	0.373 (0.017)	2.18e-02
5000_10	0.380 (0.010)	0.370 (0.010)	0.371 (0.014)	1.39e-03

A.1.2 Results for 10000-node Instances

The followings present the statistical analysis for 10000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

TABLE A.4: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
10000_1	0.173 (0.016)	0.190 (0.018)	0.188 (0.025)	3.66e-03
10000_2	0.196 (0.019)	0.180 (0.024)	0.189 (0.034)	7.13e-03
10000_3	0.174 (0.017)	0.189 (0.017)	0.181 (0.032)	8.16e-03
10000_4	0.227 (0.031)	0.196 (0.020)	0.185 (0.023)	3.07e-07
10000_5	0.204 (0.029)	0.191 (0.027)	0.170 (0.021)	1.09e-05
10000_6	0.198 (0.025)	0.177 (0.021)	0.171 (0.019)	2.98e-05
10000_7	0.177 (0.022)	0.178 (0.023)	0.160 (0.023)	3.59e-03
10000_8	0.175 (0.018)	0.205 (0.021)	0.171 (0.024)	8.78e-08
10000_9	0.197 (0.022)	0.178 (0.019)	0.162 (0.019)	1.38e-07
10000_10	0.179 (0.022)	0.169 (0.022)	0.164 (0.021)	3.96e-02

TABLE A.5: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
10000_1	0.548 (0.039)	0.630 (0.047)	0.582 (0.043)	3.23e-08
10000_2	0.570 (0.041)	0.624 (0.036)	0.597 (0.050)	1.05e-05
10000_3	0.564 (0.037)	0.595 (0.040)	0.577 (0.039)	1.82e-02
10000_4	0.602 (0.037)	0.643 (0.050)	0.628 (0.054)	3.95e-03
10000_5	0.578 (0.035)	0.621 (0.043)	0.603 (0.042)	5.82e-04
10000_6	0.580 (0.035)	0.641 (0.048)	0.600 (0.043)	1.43e-06
10000_7	0.572 (0.034)	0.610 (0.049)	0.583 (0.043)	1.66e-03
10000_8	0.575 (0.034)	0.621 (0.043)	0.590 (0.045)	2.13e-04
10000_9	0.592 (0.044)	0.627 (0.046)	0.610 (0.048)	2.70e-02
10000_10	0.568 (0.027)	0.632 (0.049)	0.599 (0.044)	1.47e-06

TABLE A.6: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
10000_1	0.361 (0.009)	0.356 (0.011)	0.354 (0.014)	4.01e-02
10000_2	0.367 (0.010)	0.371 (0.013)	0.368 (0.015)	3.69e-01
10000_3	0.347 (0.012)	0.350 (0.010)	0.347 (0.013)	3.21e-01
10000_4	0.366 (0.013)	0.383 (0.014)	0.380 (0.014)	2.43e-05
10000_5	0.349 (0.014)	0.363 (0.015)	0.360 (0.015)	3.23e-04
10000_6	0.344 (0.013)	0.356 (0.013)	0.353 (0.012)	2.16e-03
10000_7	0.355 (0.011)	0.358 (0.013)	0.362 (0.017)	2.13e-01
10000_8	0.347 (0.012)	0.341 (0.010)	0.346 (0.015)	2.69e-01
10000_9	0.350 (0.010)	0.367 (0.013)	0.364 (0.016)	3.98e-06
10000_10	0.333 (0.012)	0.341 (0.012)	0.334 (0.014)	1.45e-02

A.1.3 Results for 15000-node Instances

The followings present the statistical analysis for 10000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

TABLE A.7: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
15000_1	0.188 (0.021)	0.167 (0.019)	0.159 (0.016)	3.23e-07
15000_2	0.198 (0.021)	0.182 (0.027)	0.169 (0.021)	1.89e-05
15000_3	0.162 (0.021)	0.183 (0.013)	0.159 (0.025)	1.78e-05
15000_4	0.182 (0.025)	0.165 (0.019)	0.156 (0.018)	1.55e-04
15000_5	0.170 (0.024)	0.172 (0.018)	0.162 (0.025)	2.16e-01
15000_6	0.159 (0.017)	0.177 (0.024)	0.159 (0.019)	4.62e-03
15000_7	0.175 (0.024)	0.174 (0.020)	0.165 (0.013)	6.93e-02
15000_8	0.174 (0.020)	0.185 (0.016)	0.151 (0.021)	9.68e-08
15000_9	0.205 (0.023)	0.183 (0.016)	0.179 (0.018)	8.89e-06
15000_10	0.169 (0.017)	0.193 (0.024)	0.178 (0.020)	3.79e-04

TABLE A.8: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
15000_1	0.589 (0.040)	0.627 (0.043)	0.600 (0.044)	4.63e-03
15000_2	0.566 (0.036)	0.612 (0.041)	0.606 (0.047)	1.48e-04
15000_3	0.561 (0.032)	0.606 (0.039)	0.590 (0.035)	8.18e-05
15000_4	0.585 (0.037)	0.621 (0.036)	0.600 (0.041)	1.69e-03
15000_5	0.571 (0.036)	0.625 (0.048)	0.584 (0.049)	3.96e-05
15000_6	0.562 (0.039)	0.618 (0.037)	0.590 (0.040)	1.09e-05
15000_7	0.574 (0.037)	0.619 (0.037)	0.590 (0.036)	1.13e-04
15000_8	0.561 (0.043)	0.611 (0.038)	0.592 (0.052)	1.35e-04
15000_9	0.579 (0.039)	0.637 (0.037)	0.615 (0.053)	1.36E-05
15000_10	0.559 (0.037)	0.628 (0.044)	0.590 (0.030)	1.02E-07

TABLE A.9: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
15000_1	0.332 (0.011)	0.349 (0.011)	0.346 (0.012)	7.76e-07
15000_2	0.330 (0.011)	0.341 (0.012)	0.339 (0.015)	1.87e-03
15000_3	0.329 (0.012)	0.324 (0.008)	0.332 (0.017)	4.63e-02
15000_4	0.321 (0.012)	0.334 (0.014)	0.334 (0.015)	5.44e-04
15000_5	0.337 (0.014)	0.345 (0.011)	0.340 (0.020)	1.28e-01
15000_6	0.359 (0.011)	0.354 (0.012)	0.356 (0.016)	4.08e-01
15000_7	0.352 (0.013)	0.357 (0.012)	0.355 (0.011)	3.02e-01
15000_8	0.340 (0.010)	0.349 (0.010)	0.351 (0.019)	1.08e-02
15000_9	0.365 (0.012)	0.387 (0.015)	0.377 (0.012)	4.60e-07
15000_10	0.345 (0.010)	0.338 (0.012)	0.339 (0.014)	3.3e-02

A.1.4 Results for 20000-node Instances

The followings present the statistical analysis for 10000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

TABLE A.10: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
20000_1	0.199 (0.031)	0.176 (0.015)	0.171 (0.020)	4.37e-04
20000_2	0.180 (0.025)	0.190 (0.025)	0.171 (0.019)	6.42e-03
20000_3	0.177 (0.024)	0.222 (0.021)	0.190 (0.024)	4.15e-09
20000_4	0.190 (0.021)	0.183 (0.021)	0.168 (0.020)	7.59e-04
20000_5	0.175 (0.021)	0.184 (0.021)	0.163 (0.023)	1.16e-03
20000_6	0.183 (0.024)	0.175 (0.017)	0.164 (0.022)	9.42e-03
20000_7	0.169 (0.017)	0.174 (0.018)	0.160 (0.018)	3.18e-03
20000_8	0.183 (0.017)	0.203 (0.021)	0.172 (0.022)	1.58e-06
20000_9	0.170 (0.022)	0.188 (0.024)	0.176 (0.032)	8.39e-03
20000_10	0.192 (0.028)	0.163 (0.016)	0.161 (0.023)	2.52e-06

TABLE A.11: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
20000_1	0.581 (0.038)	0.640 (0.038)	0.617 (0.037)	1.11e-06
20000_2	0.587 (0.036)	0.639 (0.043)	0.599 (0.032)	1.37e-05
20000_3	0.553 (0.039)	0.591 (0.035)	0.583 (0.040)	6.14e-04
20000_4	0.558 (0.039)	0.628 (0.034)	0.574 (0.046)	5.00e-08
20000_5	0.579 (0.041)	0.620 (0.033)	0.597 (0.030)	1.64e-04
20000_6	0.588 (0.035)	0.624 (0.042)	0.597 (0.040)	1.64e-03
20000_7	0.562 (0.039)	0.617 (0.035)	0.584 (0.035)	5.87e-06
20000_8	0.565 (0.035)	0.609 (0.047)	0.586 (0.048)	2.08e-03
20000_9	0.559 (0.030)	0.607 (0.033)	0.593 (0.043)	7.48e-06
20000_10	0.578 (0.037)	0.635 (0.052)	0.612 (0.047)	8.35e-05

TABLE A.12: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
20000_1	0.367 (0.014)	0.389 (0.012)	0.382 (0.017)	1.11e-06
20000_2	0.357 (0.014)	0.357 (0.013)	0.362 (0.013)	1.73e-01
20000_3	0.340 (0.015)	0.324 (0.010)	0.329 (0.015)	1.81e-04
20000_4	0.332 (0.011)	0.336 (0.013)	0.338 (0.015)	1.89e-01
20000_5	0.334 (0.014)	0.337 (0.013)	0.341 (0.019)	3.05e-01
20000_6	0.331 (0.012)	0.341 (0.014)	0.340 (0.016)	2.19e-02
20000_7	0.343 (0.011)	0.343 (0.014)	0.343 (0.010)	9.83e-01
20000_8	0.337 (0.010)	0.337 (0.013)	0.342 (0.014)	1.79e-01
20000_9	0.354 (0.013)	0.349 (0.017)	0.347 (0.018)	3.83e-01
20000_10	0.346 (0.013)	0.369 (0.013)	0.364 (0.016)	7.63e-08

A.1.5 Results for 25000-node Instances

The followings present the statistical analysis for 10000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

TABLE A.13: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
25000_1	0.226 (0.028)	0.184 (0.031)	0.189 (0.027)	1.31e-06
25000_2	0.181 (0.025)	0.164 (0.020)	0.157 (0.014)	1.26e-04
25000_3	0.170 (0.019)	0.174 (0.015)	0.173 (0.029)	6.32e-01
25000_4	0.188 (0.021)	0.182 (0.023)	0.165 (0.021)	4.21e-04
25000_5	0.183 (0.022)	0.183 (0.024)	0.169 (0.025)	2.14e-02
25000_6	0.210 (0.025)	0.201 (0.023)	0.185 (0.028)	2.43e-04
25000_7	0.214 (0.030)	0.194 (0.026)	0.179 (0.026)	1.86e-04
25000_8	0.184 (0.026)	0.182 (0.013)	0.170 (0.020)	3.02e-02
25000_9	0.178 (0.021)	0.158 (0.017)	0.153 (0.020)	8.71e-05
25000_10	0.200 (0.021)	0.179 (0.019)	0.171 (0.021)	8.66e-06

TABLE A.14: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
25000_1	0.579 (0.039)	0.620 (0.041)	0.581 (0.036)	1.06e-04
25000_2	0.585 (0.040)	0.631 (0.037)	0.598 (0.036)	6.89e-05
25000_3	0.577 (0.041)	0.602 (0.040)	0.585 (0.038)	5.82e-02
25000_4	0.573 (0.043)	0.609 (0.038)	0.599 (0.033)	8.87e-03
25000_5	0.562 (0.037)	0.624 (0.037)	0.598 (0.044)	2.12e-06
25000_6	0.592 (0.054)	0.639 (0.043)	0.606 (0.054)	1.23e-03
25000_7	0.581 (0.038)	0.630 (0.052)	0.601 (0.038)	1.03e-03
25000_8	0.551 (0.043)	0.629 (0.044)	0.580 (0.038)	5.26e-08
25000_9	0.584 (0.035)	0.643 (0.044)	0.600 (0.041)	1.60e-06
25000_10	0.568 (0.056)	0.624 (0.038)	0.590 (0.049)	1.46e-04

TABLE A.15: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
25000_1	0.326 (0.012)	0.346 (0.014)	0.338 (0.018)	6.50e-06
25000_2	0.335 (0.013)	0.348 (0.012)	0.347 (0.013)	1.14e-04
25000_3	0.316 (0.012)	0.319 (0.009)	0.314 (0.015)	3.04e-01
25000_4	0.308 (0.011)	0.318 (0.013)	0.320 (0.015)	8.13e-04
25000_5	0.335 (0.011)	0.340 (0.013)	0.343 (0.020)	5.26e-02
25000_6	0.338 (0.013)	0.352 (0.018)	0.351 (0.018)	1.20e-03
25000_7	0.333 (0.012)	0.344 (0.016)	0.346 (0.017)	4.39e-03
25000_8	0.341 (0.013)	0.349 (0.012)	0.347 (0.016)	6.89e-02
25000_9	0.332 (0.010)	0.344 (0.013)	0.343 (0.015)	6.62e-04
25000_10	0.331 (0.011)	0.344 (0.013)	0.343 (0.016)	2.64e-04

A.1.6 Results for 30000-node Instances

The followings present the statistical analysis for 10000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

TABLE A.16: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
30000_1	0.329 (0.022)	0.234 (0.027)	0.276 (0.025)	6.60e-15
30000_2	0.326 (0.030)	0.191 (0.024)	0.230 (0.022)	3.24e-16
30000_3	0.279 (0.027)	0.176 (0.018)	0.194 (0.023)	2.38e-14
30000_4	0.326 (0.033)	0.188 (0.017)	0.241 (0.025)	2.20e-16
30000_5	0.244 (0.023)	0.174 (0.018)	0.175 (0.016)	1.99e-13
30000_6	0.287 (0.025)	0.191 (0.028)	0.234 (0.023)	6.16e-15
30000_7	0.242 (0.019)	0.173 (0.014)	0.188 (0.019)	7.14e-14
30000_8	0.273 (0.022)	0.196 (0.029)	0.222 (0.020)	1.55e-13
30000_9	0.254 (0.026)	0.201 (0.022)	0.210 (0.022)	5.57e-10
30000_10	0.266 (0.025)	0.195 (0.021)	0.196 (0.020)	5.01e-13

TABLE A.17: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
30000_1	0.591 (0.047)	0.643 (0.045)	0.582 (0.045)	2.48e-06
30000_2	0.595 (0.044)	0.631 (0.046)	0.606 (0.046)	2.90e-02
30000_3	0.610 (0.041)	0.644 (0.033)	0.614 (0.043)	3.66e-03
30000_4	0.594 (0.038)	0.646 (0.041)	0.607 (0.035)	5.52e-06
30000_5	0.595 (0.036)	0.627 (0.046)	0.606 (0.049)	3.49e-02
30000_6	0.593 (0.042)	0.634 (0.036)	0.610 (0.043)	5.41e-04
30000_7	0.589 (0.034)	0.642 (0.046)	0.586 (0.046)	1.16e-05
30000_8	0.584 (0.045)	0.639 (0.046)	0.599 (0.041)	6.60e-05
30000_9	0.583 (0.045)	0.651 (0.041)	0.583 (0.047)	2.24e-07
30000_10	0.589 (0.039)	0.634 (0.040)	0.599 (0.038)	2.43e-04

TABLE A.18: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
30000_1	0.273 (0.008)	0.299 (0.012)	0.292 (0.013)	7.10e-11
30000_2	0.295 (0.010)	0.348 (0.017)	0.332 (0.014)	2.88e-14
30000_3	0.286 (0.010)	0.333 (0.013)	0.325 (0.016)	1.96E-13
30000_4	0.275 (0.013)	0.324 (0.010)	0.307 (0.015)	3.36e-14
30000_5	0.287 (0.011)	0.326 (0.014)	0.328 (0.014)	5.54e-13
30000_6	0.281 (0.011)	0.317 (0.015)	0.302 (0.015)	1.35e-11
30000_7	0.282 (0.010)	0.314 (0.012)	0.307 (0.013)	3.54e-12
30000_8	0.275 (0.010)	0.304 (0.016)	0.295 (0.014)	1.05e-09
30000_9	0.296 (0.010)	0.314 (0.015)	0.315 (0.013)	2.12e-07
30000_10	0.307 (0.013)	0.338 (0.013)	0.340 (0.014)	1.20e-11

A.1.7 Results for 35000-node Instances

The followings present the statistical analysis for 10000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

TABLE A.19: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
35000_1	0.268 (0.022)	0.165 (0.015)	0.179 (0.020)	2.03e-14
35000_2	0.324 (0.023)	0.195 (0.030)	0.244 (0.021)	2.44e-16
35000_3	0.210 (0.032)	0.167 (0.012)	0.172 (0.023)	5.46e-08
35000_4	0.323 (0.025)	0.236 (0.019)	0.258 (0.030)	4.96e-13
35000_5	0.313 (0.031)	0.218 (0.024)	0.240 (0.024)	1.03e-13
35000_6	0.309 (0.019)	0.260 (0.024)	0.275 (0.028)	6.94e-09
35000_7	0.358 (0.024)	0.248 (0.032)	0.281 (0.029)	2.27e-14
35000_8	0.336 (0.040)	0.193 (0.028)	0.243 (0.019)	5.13e-16
35000_9	0.304 (0.026)	0.188 (0.020)	0.237 (0.025)	2.33e-16
35000_10	0.329 (0.026)	0.190 (0.030)	0.237 (0.025)	3.11e-16

TABLE A.20: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
35000_1	0.600 (0.035)	0.652 (0.042)	0.599 (0.038)	2.38e-06
35000_2	0.583 (0.037)	0.637 (0.040)	0.588 (0.034)	1.19e-06
35000_3	0.574 (0.036)	0.624 (0.045)	0.586 (0.034)	2.55e-05
35000_4	0.594 (0.044)	0.651 (0.037)	0.611 (0.041)	6.01e-06
35000_5	0.585 (0.044)	0.640 (0.042)	0.593 (0.042)	2.09e-05
35000_6	0.561 (0.034)	0.638 (0.047)	0.573 (0.030)	2.23e-08
35000_7	0.588 (0.040)	0.644 (0.052)	0.595 (0.040)	4.07e-05
35000_8	0.584 (0.034)	0.653 (0.042)	0.609 (0.040)	6.42e-08
35000_9	0.578 (0.037)	0.640 (0.033)	0.580 (0.035)	6.83e-09
35000_10	0.599 (0.041)	0.644 (0.039)	0.606 (0.028)	5.29e-05

TABLE A.21: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
35000_1	0.275 (0.011)	0.320 (0.012)	0.316 (0.015)	2.04e-13
35000_2	0.273 (0.010)	0.319 (0.017)	0.301 (0.013)	9.58e-14
35000_3	0.332 (0.014)	0.356 (0.014)	0.348 (0.018)	1.50e-06
35000_4	0.278 (0.011)	0.304 (0.012)	0.299 (0.013)	2.57e-10
35000_5	0.274 (0.012)	0.300 (0.011)	0.297 (0.014)	1.02e-09
35000_6	0.277 (0.009)	0.283 (0.011)	0.283 (0.015)	2.90e-02
35000_7	0.274 (0.010)	0.302 (0.016)	0.300 (0.016)	4.17e-10
35000_8	0.289 (0.015)	0.335 (0.014)	0.317 (0.010)	3.79e-14
35000_9	0.290 (0.012)	0.333 (0.013)	0.316 (0.012)	2.51e-14
35000_10	0.291 (0.008)	0.335 (0.014)	0.322 (0.012)	4.11e-14

A.1.8 Results for 40000-node Instances

The followings present the statistical analysis for 10000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

TABLE A.22: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
40000_1	0.272 (0.023)	0.210 (0.027)	0.214 (0.019)	4.07e-12
40000_2	0.252 (0.026)	0.179 (0.014)	0.168 (0.018)	3.62e-14
40000_3	0.275 (0.020)	0.193 (0.020)	0.201 (0.022)	2.22e-13
40000_4	0.302 (0.029)	0.204 (0.030)	0.234 (0.023)	2.49e-14
40000_5	0.347 (0.033)	0.209 (0.030)	0.267 (0.024)	2.20e-16
40000_6	0.276 (0.028)	0.210 (0.023)	0.206 (0.022)	8.65e-12
40000_7	0.352 (0.032)	0.176 (0.030)	0.257 (0.023)	2.20e-16
40000_8	0.314 (0.029)	0.217 (0.021)	0.250 (0.027)	7.35e-14
40000_9	0.328 (0.019)	0.204 (0.028)	0.247 (0.028)	7.92e-16
40000_10	0.298 (0.024)	0.190 (0.026)	0.220 (0.022)	7.29e-15

TABLE A.23: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
40000_1	0.578 (0.042)	0.637 (0.043)	0.575 (0.037)	6.94e-07
40000_2	0.576 (0.038)	0.644 (0.042)	0.609 (0.044)	3.32e-07
40000_3	0.577 (0.040)	0.635 (0.035)	0.591 (0.031)	2.05e-07
40000_4	0.589 (0.038)	0.639 (0.035)	0.589 (0.043)	3.92e-06
40000_5	0.600 (0.049)	0.633 (0.049)	0.603 (0.042)	3.45e-02
40000_6	0.585 (0.044)	0.627 (0.051)	0.605 (0.036)	3.65e-03
40000_7	0.610 (0.040)	0.651 (0.041)	0.608 (0.050)	7.11e-04
40000_8	0.593 (0.045)	0.650 (0.047)	0.588 (0.041)	2.91e-06
40000_9	0.588 (0.036)	0.642 (0.041)	0.605 (0.042)	1.91e-05
40000_10	0.604 (0.038)	0.663 (0.035)	0.607 (0.040)	1.15e-07

TABLE A.24: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
40000_1	0.286 (0.011)	0.311 (0.017)	0.313 (0.015)	1.53e-09
40000_2	0.269 (0.013)	0.309 (0.011)	0.319 (0.015)	4.61e-14
40000_3	0.286 (0.011)	0.320 (0.013)	0.319 (0.013)	2.34e-12
40000_4	0.272 (0.011)	0.308 (0.017)	0.302 (0.012)	5.17e-12
40000_5	0.273 (0.014)	0.314 (0.015)	0.300 (0.013)	5.45e-13
40000_6	0.265 (0.011)	0.293 (0.014)	0.295 (0.015)	1.56e-10
40000_7	0.290 (0.011)	0.351 (0.017)	0.323 (0.013)	4.13e-16
40000_8	0.279 (0.012)	0.318 (0.012)	0.310 (0.015)	1.44e-12
40000_9	0.276 (0.008)	0.320 (0.015)	0.306 (0.014)	1.70e-13
40000_10	0.280 (0.010)	0.320 (0.016)	0.313 (0.013)	3.60e-13

A.1.9 Results for 45000-node Instances

The followings present the statistical analysis for 10000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

TABLE A.25: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
45000_1	0.302 (0.033)	0.159 (0.015)	0.212 (0.022)	2.20e-16
45000_2	0.309 (0.029)	0.175 (0.021)	0.228 (0.027)	2.20e-16
45000_3	0.354 (0.024	0.233 (0.031)	0.289 (0.025)	4.19e-16
45000_4	0.314 (0.028)	0.188 (0.018)	0.230 (0.020)	2.20e-16
45000_5	0.297 (0.026)	0.208 (0.023)	0.229 (0.025)	4.05e-14
45000_6	0.345 (0.025)	0.225 (0.030)	0.288 (0.027)	5.11e-16
45000_7	0.355 (0.029)	0.227 (0.032)	0.278 (0.028)	1.70e-15
45000_8	0.411 (0.027)	0.248 (0.038)	0.326 (0.025)	2.20e-16
45000_9	0.336 (0.033)	0.200 (0.025)	0.257 (0.022)	2.20e-16
45000_10	0.350 (0.029)	0.170 (0.025)	0.259 (0.021)	2.20e-16

TABLE A.26: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
45000_1	0.597 (0.050)	0.641 (0.038)	0.606 (0.039)	5.37e-04
45000_2	0.587 (0.042)	0.663 (0.048)	0.601 (0.041)	1.24e-07
45000_3	0.569 (0.032)	0.639 (0.044)	0.599 (0.041)	1.51e-07
45000_4	0.587 (0.038)	0.657 (0.052)	0.608 (0.042)	3.30e-06
45000_5	0.592 (0.034)	0.651 (0.044)	0.599 (0.034)	1.19e-06
45000_6	0.569 (0.045)	0.642 (0.051)	0.590 (0.046)	2.35e-06
45000_7	0.581 (0.034)	0.638 (0.032)	0.602 (0.052)	3.45e-06
45000_8	0.589 (0.046)	0.638 (0.046)	0.601 (0.047)	3.62e-04
45000_9	0.595 (0.046)	0.652 (0.038)	0.615 (0.047)	2.01e-05
45000_10	0.578 (0.037)	0.643 (0.054)	0.606 (0.029)	7.67e-06

TABLE A.27: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
45000_1	0.279 (0.014)	0.340 (0.012)	0.315 (0.013)	5.92e-16
45000_2	0.283 (0.012)	0.335 (0.015)	0.314 (0.018)	9.75e-14
45000_3	0.282 (0.009)	0.321 (0.017)	0.303 (0.010)	1.09e-13
45000_4	0.288 (0.012)	0.335 (0.012)	0.322 (0.014)	5.78e-14
45000_5	0.275 (0.011)	0.306 (0.013)	0.305 (0.017)	1.83e-11
45000_6	0.269 (0.008)	0.304 (0.014)	0.286 (0.012)	2.10e-13
45000_7	0.274 (0.011)	0.312 (0.014)	0.296 (0.014)	8.99e-13
45000_8	0.256 (0.009)	0.302 (0.015)	0.283 (0.012)	8.33e-15
45000_9	0.284 (0.012)	0.327 (0.014)	0.311 (0.010)	4.16e-14
45000_10	0.288 (0.011)	0.351 (0.015)	0.324 (0.011)	2.20e-16

A.1.10 Results for Berlin and Dusseldorf Instances

The followings present the statistical analysis for Berlin and Dusseldorf instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

TABLE A.28: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
Berlin	0.270 (0.027)	0.236 (0.015)	0.330 (0.087)	3.48e-07
Dusseldorf	0.256 (0.024)	0.180 (0.016)	0.188 (0.052)	1.22e-11

Table A.29: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
Berlin	0.606 (0.043)	0.635 (0.042)	0.662 (0.205)	9.16e-03
Dusseldorf	0.542 (0.035)	0.619 (0.041)	0.557 (0.045)	8.48e-09

Table A.30: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
Berlin	0.346 (0.021)	0.348 (0.010)	0.336 (0.020)	6.66e-03
Dusseldorf	0.396 (0.014)	0.425 (0.012)	0.431 (0.015)	2.05e-11

All the statistical results for all instances has been presented. We have seen the trend that persists throughout all experiments and also a new find on SPEA-II. In the next section, we present the convergence plot of all performance indicators mentioned in this thesis, IGD, spread, and hypervolume.

A.2 Convergence Results for Real World Instances

In this section, we present the convergence plots of the real world instances only, due to the excessive amount of synthetic instances. The convergence plots in this section are used to show the behaviour of each algorithm over time. This can help us determining the termination point of the algorithms when more time yield only small improvement.

A.2.1 Convergence Plots of Berlin

There are three convergence plots for IGD, spread and hypervolume respectively. The y-axis represents the performance indicator. The x-axis represents the generation of the population.

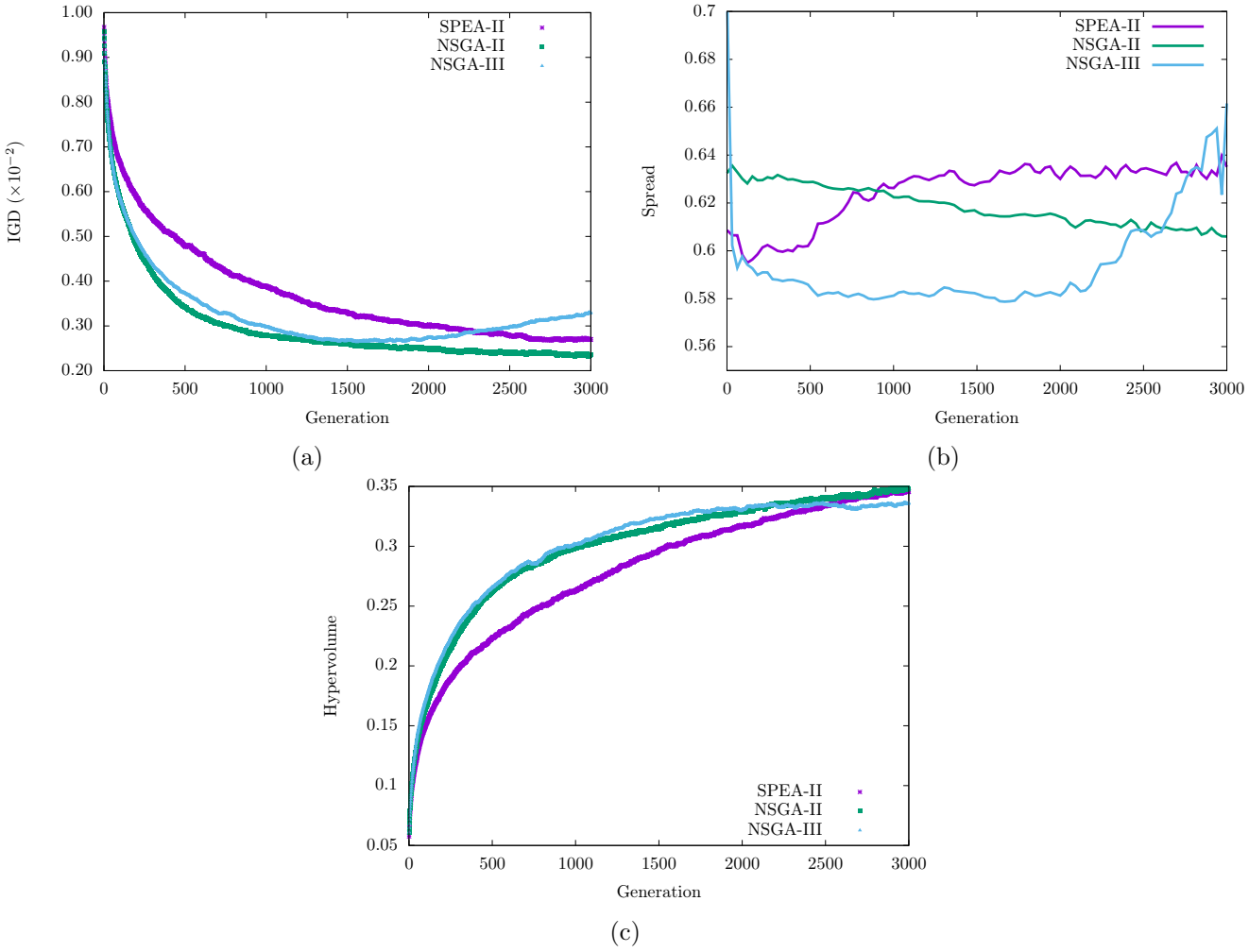


FIGURE A.1: Generational plots. (a) A generational plot of IGD (b) A generational plot of spread (c) A generational plot of hypervolume.

A.2.2 Convergence Plots of Dusseldorf

There are three convergence plots for IGD, spread and hypervolume respectively. The y-axis represents the performance indicator. The x-axis represents the generation of the population.

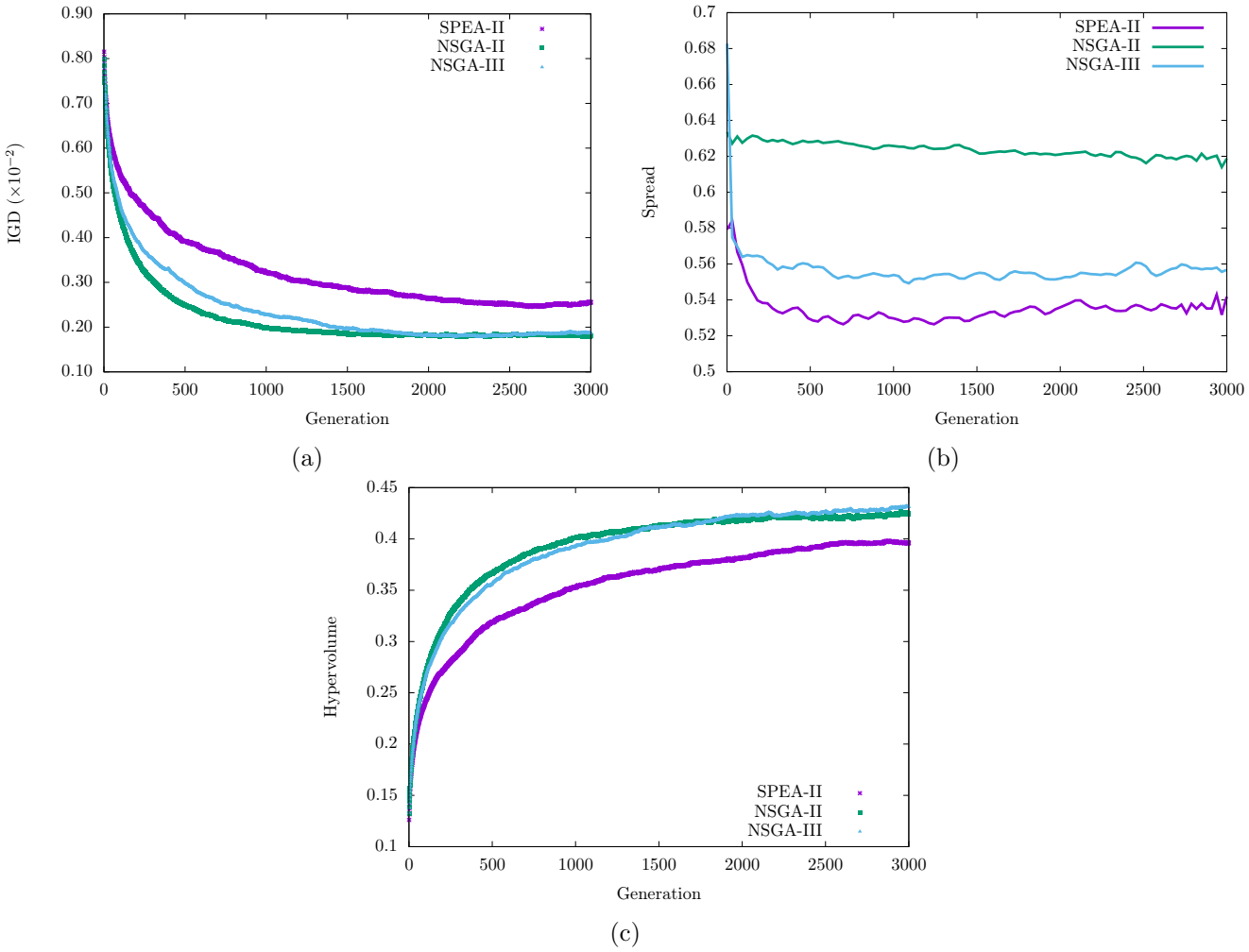


FIGURE A.2: Generational plots. (a) A generational plot of IGD (b) A generational plot of spread (c) A generational plot of hypervolume.

Appendix B

Hybrid Algorithm Results

We present the statistical analysis of the instances for the proposed hybrid algorithm, NGAP. There are three performance indicators, namely, inverted generational distance (IGD), spread and hypervolume. The discussion of the results are also provided.

B.1 Statistical Results

In the following, the results reflecting each indicator are compiled into tables. As IGD and spread need a reference solution set, we build it by consolidating the approximated fronts from 30 runs of all algorithms. Then, we employ Kruskal-Wallis test and Wilcoxon test for statistical analysis. The confidence interval for the statistical analyses is 95%

B.1.1 Statistical Results for 10000-node Instances

The followings present the statistical analysis for 10000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

Table B.1: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
10000_1	0.382 (0.067)	0.187 (0.025)	0.171 (0.020)	2.03E-14
10000_2	0.268 (0.056)	0.182 (0.019)	0.182 (0.021)	8.03e-12
10000_3	0.294 (0.035)	0.223 (0.024)	0.224 (0.023)	1.98e-11
10000_4	0.308 (0.076)	0.185 (0.018)	0.180 (0.016)	9.52e-14
10000_5	0.282 (0.052)	0.194 (0.025)	0.191 (0.020)	1.62e-11
10000_6	0.280 (0.054)	0.173 (0.022)	0.166 (0.015)	4.83e-13
10000_7	0.279 (0.051)	0.191 (0.029)	0.179 (0.018)	2.14e-12
10000_8	0.331 (0.048)	0.220 (0.028)	0.207 (0.019)	9.71e-13
10000_9	0.329 (0.052)	0.215 (0.034)	0.215 (0.026)	2.66e-12
10000_10	0.344 (0.049)	0.224 (0.029)	0.224 (0.030)	6.60e-13

TABLE B.2: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
10000_1	0.613 (0.043)	0.645 (0.049)	0.650 (0.051)	2.90e-03
10000_2	0.644 (0.044)	0.644 (0.043)	0.660 (0.046)	1.83e-01
10000_3	0.627 (0.041)	0.634 (0.052)	0.632 (0.042)	9.34e-01
10000_4	0.645 (0.053)	0.648 (0.044)	0.665 (0.046)	1.78e-01
10000_5	0.646 (0.053)	0.657 (0.047)	0.658 (0.036)	6.87e-01
10000_6	0.635 (0.032)	0.644 (0.049)	0.641 (0.055)	7.33e-01
10000_7	0.641 (0.041)	0.649 (0.055)	0.647 (0.043)	7.37e-01
10000_8	0.643 (0.052)	0.644 (0.043)	0.648 (0.025)	8.27e-01
10000_9	0.640 (0.060)	0.647 (0.047)	0.642 (0.055)	9.35e-01
10000_10	0.636 (0.041)	0.639 (0.054)	0.628 (0.034)	5.84e-01

TABLE B.3: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
10000_1	0.299 (0.017)	0.357 (0.013)	0.371 (0.014)	5.77e-15
10000_2	0.315 (0.015)	0.357 (0.014)	0.360 (0.013)	2.76e-13
10000_3	0.289 (0.011)	0.320 (0.013)	0.324 (0.012)	2.96e-13
10000_4	0.307 (0.014)	0.355 (0.011)	0.360 (0.011)	8.32E-14
10000_5	0.310 (0.016)	0.352 (0.013)	0.354 (0.013)	4.56e-13
10000_6	0.310 (0.015)	0.352 (0.015)	0.357 (0.009)	2.72e-13
10000_7	0.309 (0.014)	0.345 (0.014)	0.350 (0.013)	4.77e-13
10000_8	0.304 (0.014)	0.338 (0.012)	0.345 (0.012)	2.25e-13
10000_9	0.307 (0.014)	0.357 (0.017)	0.356 (0.013)	3.25e-13
10000_10	0.277 (0.011)	0.315 (0.012)	0.321 (0.011)	1.48e-13

B.1.2 Statistical Results for 15000-node Instances

The followings present the statistical analysis for 10000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

TABLE B.4: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
15000_1	0.323 (0.056)	0.183 (0.018)	0.179 (0.019)	1.26e-13
15000_2	0.314 (0.038)	0.221 (0.026)	0.219 (0.029)	5.54e-12
15000_3	0.342 (0.043)	0.200 (0.025)	0.191 (0.019)	9.59e-14
15000_4	0.303 (0.035)	0.188 (0.024)	0.188 (0.022)	1.93e-13
15000_5	0.330 (0.046)	0.203 (0.023)	0.202 (0.025)	2.10e-13
15000_6	0.370 (0.074)	0.183 (0.027)	0.177 (0.016)	1.57e-13
15000_7	0.328 (0.071)	0.186 (0.026)	0.178 (0.018)	1.93e-13
15000_8	0.341 (0.050)	0.247 (0.032)	0.241 (0.028)	5.83e-12
15000_9	0.278 (0.060)	0.174 (0.018)	0.170 (0.016)	2.06e-13
15000_10	0.351 (0.047)	0.176 (0.014)	0.177 (0.018)	1.28e-13

TABLE B.5: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
15000_1	0.634 (0.038)	0.644 (0.051)	0.647 (0.043)	5.52e-01
15000_2	0.628 (0.035)	0.616 (0.049)	0.630 (0.035)	3.49e-01
15000_3	0.605 (0.044)	0.646 (0.043)	0.633 (0.045)	9.38e-04
15000_4	0.628 (0.048)	0.640 (0.052)	0.646 (0.041)	3.18e-01
15000_5	0.629 (0.036)	0.655 (0.042)	0.621 (0.052)	1.87e-02
15000_6	0.612 (0.042)	0.647 (0.046)	0.644 (0.053)	1.13e-02
15000_7	0.625 (0.044)	0.652 (0.052)	0.629 (0.049)	8.87e-02
15000_8	0.626 (0.038)	0.621 (0.054)	0.636 (0.047)	4.57e-01
15000_9	0.657 (0.038)	0.655 (0.051)	0.664 (0.047)	6.33e-01
15000_10	0.610 (0.045)	0.633 (0.054)	0.633 (0.045)	1.02e-01

TABLE B.6: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
15000_1	0.282 (0.016)	0.338 (0.014)	0.343 (0.013)	7.33e-14
15000_2	0.281 (0.012)	0.311 (0.012)	0.317 (0.012)	1.40e-12
15000_3	0.277 (0.010)	0.328 (0.014)	0.338 (0.014)	2.15e-14
15000_4	0.263 (0.011)	0.310 (0.014)	0.315 (0.013)	8.99e-14
15000_5	0.287 (0.014)	0.336 (0.014)	0.337 (0.013)	2.26e-13
15000_6	0.285 (0.014)	0.347 (0.015)	0.352 (0.011)	1.29e-13
15000_7	0.308 (0.016)	0.361 (0.017)	0.367 (0.013)	1.21e-13
15000_8	0.287 (0.014)	0.317 (0.013)	0.323 (0.012)	2.27e-12
15000_9	0.321 (0.014)	0.372 (0.012)	0.376 (0.009)	6.59e-14
15000_10	0.287 (0.012)	0.340 (0.011)	0.345 (0.011)	8.45e-14

B.1.3 Statistical Results for 20000-node Instances

The followings present the statistical analysis for 10000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

TABLE B.7: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
20000_1	0.320 (0.064)	0.164 (0.022)	0.161 (0.017)	1.25e-13
20000_2	0.307 (0.058)	0.181 (0.024)	0.173 (0.018)	2.45e-13
20000_3	0.323 (0.048)	0.223 (0.029)	0.211 (0.029)	4.77e-12
20000_4	0.364 (0.064)	0.172 (0.018)	0.166 (0.020)	8.14e-14
20000_5	0.320 (0.053)	0.198 (0.024)	0.204 (0.026)	3.55e-12
20000_6	0.363 (0.050)	0.204 (0.027)	0.189 (0.027)	6.25e-14
20000_7	0.333 (0.082)	0.173 (0.014)	0.164 (0.016)	1.31e-13
20000_8	0.361 (0.050)	0.224 (0.023)	0.207 (0.024)	1.19e-13
20000_9	0.362 (0.053)	0.185 (0.026)	0.172 (0.017)	4.34e-14
20000_10	0.308 (0.051)	0.191 (0.022)	0.183 (0.021)	1.07e-13

TABLE B.8: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
20000_1	0.641 (0.056)	0.653 (0.043)	0.641 (0.052)	5.78e-01
20000_2	0.637 (0.045)	0.665 (0.048)	0.661 (0.057)	7.92e-02
20000_3	0.627 (0.050)	0.629 (0.040)	0.639 (0.045)	7.20e-01
20000_4	0.615 (0.044)	0.653 (0.056)	0.644 (0.051)	1.33e-02
20000_5	0.639 (0.060)	0.638 (0.045)	0.644 (0.041)	6.77e-01
20000_6	0.630 (0.040)	0.632 (0.046)	0.623 (0.045)	9.37e-01
20000_7	0.650 (0.054)	0.646 (0.045)	0.637 (0.044)	6.03e-01
20000_8	0.614 (0.036)	0.612 (0.054)	0.624 (0.054)	7.44e-01
20000_9	0.613 (0.058)	0.632 (0.038)	0.633 (0.045)	3.03e-01
20000_10	0.641 (0.037)	0.656 (0.053)	0.663 (0.050)	3.59e-01

TABLE B.9: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
20000_1	0.310 (0.019)	0.378 (0.017)	0.379 (0.012)	1.27e-13
20000_2	0.307 (0.018)	0.359 (0.014)	0.366 (0.015)	1.67e-13
20000_3	0.280 (0.015)	0.317 (0.016)	0.327 (0.017)	9.31e-13
20000_4	0.277 (0.015)	0.341 (0.016)	0.349 (0.016)	5.23e-14
20000_5	0.280 (0.012)	0.325 (0.013)	0.324 (0.013)	2.45e-13
20000_6	0.277 (0.014)	0.328 (0.014)	0.340 (0.017)	3.90e-14
20000_7	0.280 (0.019)	0.340 (0.009)	0.350 (0.014)	2.17e-14
20000_8	0.273 (0.012)	0.314 (0.010)	0.327 (0.014)	1.18e-14
20000_9	0.286 (0.014)	0.343 (0.017)	0.353 (0.010)	2.11e-14
20000_10	0.292 (0.014)	0.347 (0.012)	0.358 (0.016)	2.63e-14

B.1.4 Statistical Results for 25000-node Instances

The followings present the statistical analysis for 10000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

TABLE B.10: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
25000_1	0.325 (0.055)	0.172 (0.020)	0.160 (0.018)	4.66e-14
25000_2	0.335 (0.046)	0.171 (0.021)	0.159 (0.021)	3.86e-14
25000_3	0.317 (0.030)	0.219 (0.026)	0.222 (0.028)	2.38e-13
25000_4	0.318 (0.037)	0.229 (0.030)	0.227 (0.028)	2.17e-13
25000_5	0.368 (0.059)	0.191 (0.020)	0.185 (0.012)	1.08e-13
25000_6	0.331 (0.054)	0.179 (0.024)	0.170 (0.018)	7.52e-14
25000_7	0.353 (0.053)	0.225 (0.030)	0.205 (0.032)	8.06e-14
25000_8	0.344 (0.037)	0.191 (0.026)	0.180 (0.021)	7.05e-14
25000_9	0.328 (0.049)	0.181 (0.020)	0.172 (0.017)	6.92e-14
25000_10	0.367 (0.066)	0.159 (0.016)	0.153 (0.017)	8.30e-14

TABLE B.11: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
25000_1	0.624 (0.048)	0.633 (0.048)	0.649 (0.055)	1.13e-01
25000_2	0.638 (0.043)	0.651 (0.050)	0.657 (0.052)	2.67e-01
25000_3	0.613 (0.056)	0.616 (0.037)	0.625 (0.052)	6.51e-01
25000_4	0.607 (0.043)	0.626 (0.043)	0.629 (0.046)	2.46e-01
25000_5	0.624 (0.054)	0.638 (0.041)	0.643 (0.042)	1.49e-01
25000_6	0.633 (0.050)	0.648 (0.036)	0.646 (0.033)	2.59e-01
25000_7	0.623 (0.042)	0.647 (0.051)	0.647 (0.040)	6.56e-02
25000_8	0.617 (0.047)	0.651 (0.048)	0.621 (0.046)	2.07e-02
25000_9	0.645 (0.047)	0.644 (0.046)	0.637 (0.051)	8.06e-01
25000_10	0.632 (0.040)	0.650 (0.057)	0.626 (0.040)	2.32e-01

TABLE B.12: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
25000_1	0.278 (0.013)	0.335 (0.015)	0.346 (0.015)	3.95e-14
25000_2	0.287 (0.016)	0.351 (0.014)	0.363 (0.015)	1.94e-14
25000_3	0.256 (0.009)	0.296 (0.013)	0.300 (0.014)	9.07e-14
25000_4	0.262 (0.013)	0.298 (0.013)	0.300 (0.012)	3.85e-13
25000_5	0.267 (0.015)	0.336 (0.014)	0.341 (0.010)	7.91e-14
25000_6	0.285 (0.018)	0.346 (0.014)	0.353 (0.013)	7.52e-14
25000_7	0.267 (0.013)	0.312 (0.013)	0.328 (0.016)	7.27e-15
25000_8	0.275 (0.011)	0.329 (0.015)	0.337 (0.014)	7.21e-14
25000_9	0.275 (0.015)	0.328 (0.013)	0.336 (0.013)	3.73e-14
25000_10	0.273 (0.016)	0.351 (0.013)	0.358 (0.012)	4.66e-14

B.1.5 Statistical Results for 30000-node Instances

The followings present the statistical analysis for 10000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

TABLE B.13: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
30000_1	0.407 (0.060)	0.157 (0.019)	0.139 (0.016)	1.01e-14
30000_2	0.355 (0.070)	0.168 (0.017)	0.174 (0.024)	1.29e-13
30000_3	0.327 (0.040)	0.173 (0.018)	0.173 (0.019)	1.30e-13
30000_4	0.358 (0.068)	0.134 (0.012)	0.129 (0.013)	5.92e-14
30000_5	0.297 (0.029)	0.185 (0.024)	0.181 (0.023)	1.32e-13
30000_6	0.348 (0.077)	0.130 (0.016)	0.126 (0.013)	1.03e-13
30000_7	0.295 (0.050)	0.163 (0.022)	0.165 (0.023)	2.01e-13
30000_8	0.336 (0.065)	0.127 (0.016)	0.116 (0.012)	2.49e-14
30000_9	0.341 (0.044)	0.170 (0.028)	0.156 (0.014)	6.36e-14
30000_10	0.317 (0.035)	0.180 (0.025)	0.176 (0.024)	1.48e-13

TABLE B.14: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
30000_1	0.631 (0.050)	0.625 (0.060)	0.647 (0.062)	3.58e-01
30000_2	0.637 (0.049)	0.629 (0.049)	0.645 (0.059)	5.78e-01
30000_3	0.615 (0.042)	0.643 (0.043)	0.621 (0.046)	4.31e-02
30000_4	0.628 (0.053)	0.641 (0.062)	0.631 (0.051)	5.44e-01
30000_5	0.638 (0.049)	0.639 (0.043)	0.618 (0.046)	1.63e-01
30000_6	0.620 (0.049)	0.632 (0.037)	0.642 (0.047)	1.61e-01
30000_7	0.620 (0.047)	0.601 (0.047)	0.648 (0.045)	1.70e-03
30000_8	0.639 (0.047)	0.638 (0.052)	0.620 (0.044)	2.77e-01
30000_9	0.599 (0.044)	0.631 (0.059)	0.621 (0.051)	6.83e-02
30000_10	0.638 (0.042)	0.636 (0.048)	0.629 (0.058)	4.39e-01

TABLE B.15: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
30000_1	0.244 (0.013)	0.335 (0.011)	0.349 (0.014)	5.71e-15
30000_2	0.280 (0.017)	0.362 (0.009)	0.365 (0.015)	1.07e-13
30000_3	0.269 (0.012)	0.342 (0.013)	0.347 (0.015)	1.01e-13
30000_4	0.261 (0.012)	0.345 (0.011)	0.354 (0.011)	1.20e-14
30000_5	0.268 (0.011)	0.325 (0.015)	0.330 (0.014)	9.78e-14
30000_6	0.263 (0.014)	0.348 (0.012)	0.354 (0.011)	5.27e-14
30000_7	0.262 (0.013)	0.323 (0.013)	0.328 (0.015)	1.01e-13
30000_8	0.255 (0.017)	0.336 (0.014)	0.349 (0.013)	1.20e-14
30000_9	0.266 (0.014)	0.333 (0.016)	0.348 (0.011)	8.70e-15
30000_10	0.287 (0.013)	0.347 (0.017)	0.353 (0.016)	1.05e-13

B.1.6 Statistical Results for 35000-node Instances

The followings present the statistical analysis for 10000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

TABLE B.16: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
35000_1	0.337 (0.033)	0.170 (0.022)	0.170 (0.022)	1.29e-13
35000_2	0.362 (0.079)	0.143 (0.019)	0.136 (0.014)	8.84e-14
35000_3	0.338 (0.084)	0.183 (0.019)	0.190 (0.015)	1.14e-10
35000_4	0.397 (0.037)	0.160 (0.018)	0.156 (0.014)	1.15e-13
35000_5	0.372 (0.049)	0.165 (0.018)	0.162 (0.019)	1.11e-13
35000_6	0.384 (0.038)	0.151 (0.009)	0.139 (0.013)	4.67e-15
35000_7	0.442 (0.056)	0.147 (0.016)	0.143 (0.020)	1.04e-13
35000_8	0.422 (0.066)	0.152 (0.019)	0.147 (0.014)	1.24e-13
35000_9	0.400 (0.058)	0.159 (0.018)	0.157 (0.016)	1.20e-13
35000_10	0.402 (0.076)	0.161 (0.019)	0.141 (0.019)	6.77e-15

TABLE B.17: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
35000_1	0.614 (0.039)	0.629 (0.043)	0.632 (0.041)	2.63e-01
35000_2	0.623 (0.047)	0.623 (0.054)	0.625 (0.047)	9.30e-01
35000_3	0.623 (0.057)	0.625 (0.038)	0.634 (0.049)	3.30e-01
35000_4	0.616 (0.046)	0.643 (0.046)	0.634 (0.042)	6.54e-02
35000_5	0.626 (0.045)	0.622 (0.041)	0.632 (0.048)	7.15e-01
35000_6	0.622 (0.045)	0.630 (0.039)	0.623 (0.038)	9.20e-01
35000_7	0.611 (0.049)	0.649 (0.052)	0.628 (0.039)	1.79e-02
35000_8	0.620 (0.048)	0.637 (0.053)	0.626 (0.041)	4.04e-01
35000_9	0.630 (0.042)	0.645 (0.043)	0.618 (0.047)	5.13e-02
35000_10	0.640 (0.039)	0.645 (0.050)	0.623 (0.051)	4.70e-01

Table B.18: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
35000_1	0.256 (0.011)	0.327 (0.013)	0.334 (0.013)	5.41E-14
35000_2	0.261 (0.015)	0.340 (0.014)	0.346 (0.013)	8.22e-14
35000_3	0.299 (0.020)	0.423 (0.015)	0.423 (0.014)	1.30e-13
35000_4	0.255 (0.010)	0.341 (0.015)	0.347 (0.013)	6.29e-14
35000_5	0.253 (0.010)	0.331 (0.011)	0.337 (0.011)	5.92e-14
35000_6	0.250 (0.011)	0.330 (0.009)	0.343 (0.012)	1.91e-15
35000_7	0.256 (0.015)	0.350 (0.015)	0.353 (0.015)	1.18e-13
35000_8	0.261 (0.015)	0.351 (0.011)	0.354 (0.011)	1.05e-13
35000_9	0.270 (0.015)	0.349 (0.014)	0.355 (0.012)	7.91e-14
35000_10	0.266 (0.015)	0.347 (0.012)	0.363 (0.012)	2.99e-15

B.1.7 Statistical Results for 40000-node Instances

The followings present the statistical analysis for 10000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

Table B.19: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
40000_1	0.328 (0.036)	0.170 (0.021)	0.161 (0.019)	7.44e-14
40000_2	0.308 (0.032)	0.191 (0.025)	0.188 (0.020)	1.17e-13
40000_3	0.362 (0.044)	0.162 (0.020)	0.159 (0.016)	1.17e-13
40000_4	0.360 (0.056)	0.148 (0.014)	0.142 (0.022)	8.30e-14
40000_5	0.429 (0.081)	0.141 (0.014)	0.129 (0.013)	2.07e-14
40000_6	0.333 (0.044)	0.167 (0.016)	0.176 (0.022)	7.44e-14
40000_7	0.365 (0.074)	0.166 (0.016)	0.162 (0.021)	1.16e-13
40000_8	0.367 (0.043)	0.173 (0.018)	0.158 (0.018)	2.40e-14
40000_9	0.372 (0.068)	0.138 (0.017)	0.138 (0.011)	1.27e-13
40000_10	0.362 (0.040)	0.144 (0.014)	0.136 (0.019)	6.44e-14

TABLE B.20: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
40000_1	0.619 (0.044)	0.624 (0.050)	0.632 (0.045)	5.46e-01
40000_2	0.616 (0.045)	0.630 (0.036)	0.615 (0.048)	3.68e-01
40000_3	0.618 (0.034)	0.632 (0.042)	0.633 (0.054)	2.90e-01
40000_4	0.632 (0.041)	0.632 (0.044)	0.636 (0.049)	8.40e-01
40000_5	0.625 (0.050)	0.637 (0.040)	0.634 (0.041)	3.43e-01
40000_6	0.612 (0.033)	0.630 (0.037)	0.625 (0.051)	1.19e-01
40000_7	0.637 (0.047)	0.642 (0.057)	0.652 (0.051)	4.82e-01
40000_8	0.643 (0.042)	0.629 (0.056)	0.643 (0.052)	5.85e-01
40000_9	0.627 (0.046)	0.627 (0.064)	0.615 (0.053)	7.18e-01
40000_10	0.634 (0.035)	0.610 (0.046)	0.624 (0.034)	5.96e-02

TABLE B.21: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
40000_1	0.268 (0.012)	0.336 (0.014)	0.347 (0.016)	2.54e-14
40000_2	0.256 (0.013)	0.309 (0.016)	0.317 (0.015)	5.13e-14
40000_3	0.259 (0.014)	0.334 (0.017)	0.348 (0.011)	1.20e-14
40000_4	0.252 (0.012)	0.330 (0.011)	0.339 (0.017)	4.40e-14
40000_5	0.254 (0.018)	0.343 (0.012)	0.357 (0.014)	8.77e-15
40000_6	0.248 (0.012)	0.321 (0.011)	0.325 (0.014)	9.67e-14
40000_7	0.278 (0.018)	0.351 (0.013)	0.357 (0.013)	6.07e-14
40000_8	0.262 (0.013)	0.338 (0.012)	0.356 (0.015)	2.60e-15
40000_9	0.263 (0.013)	0.349 (0.015)	0.350 (0.011)	1.29e-13
40000_10	0.257 (0.011)	0.343 (0.013)	0.356 (0.016)	1.61e-14

B.1.8 Statistical Results for 45000-node Instances

The followings present the statistical analysis for 10000-node synthetic instances. The first table is for IGD. The second table presents spread indicator and the last table shows the hypervolume indicator for each algorithm.

TABLE B.22: Average results obtained for the IGD metric.

Instance	IGD ($\times 10^{-2}$)			
	SPEA-II	NSGA-II	NSGA-III	p-value
45000_1	0.390 (0.065)	0.160 (0.020)	0.137 (0.021)	6.30e-15
45000_2	0.399 (0.065)	0.155 (0.017)	0.148 (0.016)	8.61e-14
45000_3	0.462 (0.055)	0.161 (0.021)	0.152 (0.017)	6.37e-14
45000_4	0.400 (0.055)	0.170 (0.021)	0.162 (0.022)	8.53e-14
45000_5	0.371 (0.032)	0.175 (0.019)	0.155 (0.023)	7.63e-15
45000_6	0.441 (0.066)	0.131 (0.016)	0.131 (0.014)	1.29e-13
45000_7	0.431 (0.078)	0.151 (0.018)	0.141 (0.015)	4.79e-14
45000_8	0.471 (0.085)	0.156 (0.019)	0.134 (0.018)	2.83e-15
45000_9	0.449 (0.044)	0.154 (0.016)	0.151 (0.022)	1.17e-13
45000_10	0.438 (0.079)	0.172 (0.021)	0.172 (0.027)	1.28e-13

TABLE B.23: Average results obtained for the spread metric.

Instance	Spread			
	SPEA-II	NSGA-II	NSGA-III	p-value
45000_1	0.623 (0.054)	0.627 (0.048)	0.632 (0.053)	6.89e-01
45000_2	0.631 (0.051)	0.628 (0.044)	0.643 (0.045)	3.04e-01
45000_3	0.617 (0.045)	0.624 (0.049)	0.630 (0.037)	4.58e-01
45000_4	0.604 (0.041)	0.644 (0.040)	0.634 (0.042)	1.20e-03
45000_5	0.602 (0.044)	0.614 (0.048)	0.615 (0.049)	6.82e-01
45000_6	0.606 (0.047)	0.639 (0.050)	0.619 (0.047)	5.67e-02
45000_7	0.600 (0.057)	0.631 (0.040)	0.647 (0.055)	3.59e-03
45000_8	0.641 (0.036)	0.639 (0.043)	0.632 (0.045)	6.49e-01
45000_9	0.621 (0.055)	0.647 (0.048)	0.629 (0.059)	1.37e-01
45000_10	0.630 (0.045)	0.619 (0.048)	0.635 (0.052)	3.02e-01

TABLE B.24: Average results obtained for the hypervolume metric.

Instance	Hypervolume			
	SPEA-II	NSGA-II	NSGA-III	p-value
45000_1	0.253 (0.015)	0.331 (0.013)	0.355 (0.014)	4.30e-16
45000_2	0.262 (0.013)	0.337 (0.012)	0.347 (0.014)	3.96e-14
45000_3	0.256 (0.016)	0.346 (0.015)	0.353 (0.014)	5.63e-14
45000_4	0.263 (0.012)	0.343 (0.016)	0.354 (0.015)	2.54e-14
45000_5	0.256 (0.010)	0.321 (0.014)	0.343 (0.016)	4.91e-16
45000_6	0.245 (0.013)	0.345 (0.014)	0.347 (0.012)	1.28e-13
45000_7	0.253 (0.013)	0.340 (0.017)	0.349 (0.014)	3.79e-14
45000_8	0.245 (0.016)	0.335 (0.011)	0.350 (0.012)	3.07e-15
45000_9	0.253 (0.014)	0.345 (0.012)	0.348 (0.016)	1.09e-13
45000_10	0.259 (0.016)	0.345 (0.014)	0.352 (0.017)	7.28e-14

..

Publications

[1] B. Changaival, G. Danoy, D. Kliazovich, F. Guinand, M. Brust, J. Musial, K. Lavangnananda and P. Bouvry, "Toward real-world vehicle placement optimization in round-trip carsharing", *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1138–1146, Prague, Czech Republic, 2019.

[2] B. Changaival and M. Rosalie, "Exploring chaotic dynamics by partition of bifurcation diagram", *Proceedings of WANCSA 2017, Workshop on Advance in NonLinear Complex Systems and Applications*, pp. 7–8, Le Havre, France, 2017.

[3] B. Changaival, M. Rosalie, G. Danoy, K. Lavangnananda and P. Bouvry, "Chaotic Traversal (CHAT): Very Large Graphs Traversal Using Chaotic Dynamics", *International Journal of Bifurcation and Chaos*, vol. 27, no. 1, pp. 175–215, World Scientific, 2017.

[4] B. Changaival, G. Danoy, M. Ostaszewski, K. Lavangnananda and P. Bouvry, "Metaheuristic Based Clustering Algorithms for Biological Hypergraphs", *Proceedings of META'2016, 6th International Conference on Metaheuristics and Nature Inspired computing*, pp. 364–366, Marrakech, Morocco, 2016.

References

[1] Bernard Marr, "How much data do we create every day? the mind-blowing stats everyone should read," 2018. [Online; accessed 11-06-2019].
One citation in page 1.

[2] M. Newman, *Networks: an introduction*. Oxford university press, 2010.
One citation in page 1.

[3] A.-L. Barabási *et al.*, *Network science*. Cambridge university press, 2016.
2 citations in pages 1 and 2.

[4] R. Angles and C. Gutierrez, "Survey of graph database models," *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, p. 1, 2008.
One citation in page 1.

[5] Neo4j, "Why graph databases?," 2018. [Online; accessed 13-06-2019].
One citation in page 1.

[6] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pp. 11–15, Aug. 2008.
2 citations in pages 1 and 78.

[7] OpenStreetMap contributors, "Cities network dump retrieved from https://planet.osm.org ." `https://www.openstreetmap.org`, 2017.
4 citations in pages 1, 17, 18, and 26.

[8] C. C. Aggarwal and C. Zhai, *Mining text data*. Springer Science & Business Media, 2012.
One citation in page 1.

[9] R. A. Hill and R. I. Dunbar, "Social network size in humans," *Human nature*, vol. 14, no. 1, pp. 53–72, 2003.
One citation in page 2.

[10] C. Benevolo, R. P. Dameri, and B. D'Auria, "Smart mobility in smart city," in *Empowering Organizations*, pp. 13–28, Springer, 2016.
One citation in page 5.

[11] E. Martin, S. Shaheen, and J. Lidicker, "Impact of carsharing on household vehicle holdings: Results from north american shared-use vehicle survey," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2143, pp. 150–158, 2010.
One citation in page 5.

[12] S. A. Shaheen and A. P. Cohen, "Growth in worldwide carsharing: An international comparison," *Transportation Research Record*, vol. 1992, no. 1, pp. 81–89, 2007.
One citation in page 5.

[13] Frost&Sullivan, "Future of carsharing market to 2025," 2016. [Online; accessed 19-October-2018].
2 citations in pages 5 and 26.

[14] Deloitte, "Car sharing in europe: Business models, national variations and upcoming disruptions," 2017. [Online; accessed 29-August-2017].
One citation in page 5.

[15] CARSHARING-NEWS, "Carsharing anbieter," 2019. [Online; accessed 3-April-2019].
One citation in page 5.

[16] G. Laporte, S. Nickel, and F. S. da Gama, *Location science*, vol. 528. Springer, 2015.
2 citations in pages 6 and 7.

[17] R. Church and C. ReVelle, "The maximal covering location problem," in *Papers of the Regional Science Association*, vol. 32, pp. 101–118, Springer, 1974.
4 citations in pages 6, 9, 23, and 96.

[18] N. Megiddo and K. J. Supowit, "On the complexity of some common geometric location problems," *SIAM journal on computing*, vol. 13, no. 1, pp. 182–196, 1984.
One citation in page 6.

[19] D. B. Seargeant, *The Maximal Covering Location Problem: An Application in Reproductive Health Services*. PhD thesis, UCLA, 2012.
2 citations in pages 6 and 8.

[20] V. Schmid and K. F. Doerner, "Ambulance location and relocation problems with time-dependent travel times," *European journal of operational research*, vol. 207, no. 3, pp. 1293–1303, 2010.
2 citations in pages 6 and 8.

[21] N. Ghaffarinasab and A. Motallebzadeh, "Hub interdiction problem variants: Models and metaheuristic solution algorithms," *European Journal of Operational Research*, vol. 267, no. 2, pp. 496–512, 2018.
One citation in page 6.

[22] N. Xiao, D. A. Bennett, and M. P. Armstrong, "Using evolutionary algorithms to generate alternatives for multiobjective site-search problems," *Environment and Planning A*, vol. 34, no. 4, pp. 639–656, 2002.
2 citations in pages 6 and 11.

[23] K. Kim and A. T. Murray, "Enhancing spatial representation in primary and secondary coverage location modeling," *Journal of Regional Science*, vol. 48, no. 4, pp. 745–768, 2008.
2 citations in pages 6 and 11.

[24] H. Malekpoor, K. Chalvatzis, N. Mishra, and A. Ramudhin, "A hybrid approach of vikor and bi-objective integer linear programming for electrification planning in a disaster relief camp," *Annals of Operations Research*, pp. 1–27, 2018.
2 citations in pages 7 and 8.

[25] N. Megiddo and A. Tamir, "On the complexity of locating linear facilities in the plane," *Operations research letters*, vol. 1, no. 5, pp. 194–197, 1982.
One citation in page 7.

[26] L. Raisanen and R. M. Whitaker, "Comparison and evaluation of multiple objective genetic algorithms for the antenna placement problem," *Mobile Networks and Applications*, vol. 10, no. 1-2, pp. 79–88, 2005.
2 citations in pages 7 and 23.

[27] M. Vasquez and J.-K. Hao, "A heuristic approach for antenna positioning in cellular networks," *Journal of Heuristics*, vol. 7, no. 5, pp. 443–472, 2001.
One citation in page 7.

[28] T. Sakai, K. Kawamura, and T. Hyodo, "The relationship between commodity types, spatial characteristics, and distance optimality of logistics facilities," *Journal of Transport and Land Use*, vol. 11, no. 1, 2018.
One citation in page 7.

[29] N. Razi and M. Karatas, "A multi-objective model for locating search and rescue boats," *European Journal of Operational Research*, vol. 254, no. 1, pp. 279–293, 2016.
2 citations in pages 7 and 8.

[30] P. Kumar and M. Bierlaire, "Optimizing locations for a vehicle sharing system," in *Swiss Transport Research Conference*, no. EPFL-CONF-195890, 2012.
4 citations in pages 7, 8, 9, and 16.

[31] A. Awasthi, D. Breuil, S. S. Chauhan, M. Parent, and T. Reveillere, "A multicriteria decision making approach for carsharing stations selection," *Journal of decision systems*, vol. 16, no. 1, pp. 57–78, 2007.
One citation in page 7.

[32] B. Boyacı, K. G. Zografos, and N. Geroliminis, "An optimization framework for the development of efficient one-way car-sharing systems," *European Journal of Operational Research*, vol. 240, no. 3, pp. 718–733, 2015.
3 citations in pages 7, 8, and 9.

[33] M. Lage, C. Machado, F. Berssaneti, and J. Quintanilha, "A method to define the spatial stations location in a carsharing system in são paulo–brazil," *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci*, pp. 27–32, 2018.
One citation in page 7.

[34] J. Schwer and S. Timpf, "Local-level site-selection model for integrated carsharing services," *GI_Forum 2016,*, vol. 4, pp. 243–249, 2016.
One citation in page 7.

[35] H. Calık and B. Fortz, "Location of stations in a one-way electric car sharing system," in *2017 IEEE Symposium on Computers and Communications (ISCC)*, pp. 134–139, IEEE, 2017.
One citation in page 8.

[36] Z. Jiao, L. Ran, J. Chen, H. Meng, and C. Li, "Data-driven approach to operation and location considering range anxiety of one-way electric vehicles sharing system," *Energy Procedia*, vol. 105, pp. 2287–2294, 2017.
One citation in page 8.

[37] G. Brandstätter, M. Kahr, and M. Leitner, "Determining optimal locations for charging stations of electric car-sharing systems under stochastic demand," *Transportation Research Part B: Methodological*, vol. 104, pp. 17–35, 2017.
One citation in page 8.

[38] J. P. Romero, A. Ibeas, J. L. Moura, J. Benavente, and B. Alonso, "A simulation-optimization approach to design efficient systems of bike-sharing," *Procedia-Social and Behavioral Sciences*, vol. 54, pp. 646–655, 2012.
One citation in page 8.

[39] T. Raviv, M. Tzur, and I. A. Forma, "Static repositioning in a bike-sharing system: models and solution approaches," *EURO Journal on Transportation and Logistics*, vol. 2, no. 3, pp. 187–229, 2013.
One citation in page 8.

[40] J. C. García-Palomares, J. Gutiérrez, and M. Latorre, "Optimizing the location of stations in bike-sharing programs: a gis approach," *Applied Geography*, vol. 35, no. 1-2, pp. 235–246, 2012.
One citation in page 8.

[41] J. Liu, Q. Li, M. Qu, W. Chen, J. Yang, H. Xiong, H. Zhong, and Y. Fu, "Station site optimization in bike sharing systems," in *Data Mining (ICDM), 2015 IEEE International Conference on*, pp. 883–888, IEEE, 2015.
One citation in page 8.

[42] S. Weikl and K. Bogenberger, "Relocation strategies and algorithms for free-floating car sharing systems," *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 4, pp. 100–111, 2013.
One citation in page 8.

[43] M. Barth and M. Todd, "Simulation model performance analysis of a multiple station shared vehicle system," *Transportation Research Part C: Emerging Technologies*, vol. 7, no. 4, pp. 237–259, 1999.
One citation in page 8.

[44] I. I. CPLEX, "V12. 1: User's manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.
One citation in page 8.

[45] R. Fourer, D. M. Gay, and B. Kernighan, *Ampl*, vol. 117. Boyd & Fraser Danvers, MA, 1993.
One citation in page 8.

[46] M. Efroymson and T. Ray, "A branch-bound algorithm for plant location," *Operations Research*, vol. 14, no. 3, pp. 361–368, 1966.
One citation in page 8.

[47] M. F. Zarandi, S. Davari, and S. H. Sisakht, "The large scale maximal covering location problem," *Scientia Iranica*, vol. 18, no. 6, pp. 1564–1570, 2011.
3 citations in pages 8, 10, and 36.

[48] W. Manopiniwes and T. Irohara, "Stochastic optimisation model for integrated decisions on relief supply chains: preparedness for disaster response," *International Journal of Production Research*, vol. 55, no. 4, pp. 979–996, 2017.
One citation in page 8.

[49] G. H. de Almeida Correia and A. P. Antunes, "Optimization approach to depot location and trip selection in one-way carsharing systems," *Transportation Research Part E: Logistics and Transportation Review*, vol. 48, no. 1, pp. 233–247, 2012.
One citation in page 8.

[50] J. Asamer, M. Reinthaler, M. Ruthmair, M. Straub, and J. Puchinger, "Optimizing charging station locations for urban taxi providers," *Transportation Research Part A: Policy and Practice*, vol. 85, pp. 233–246, 2016.
One citation in page 8.

[51] R. Borndörfer, S. Schenker, M. Skutella, and T. Strunk, "Polyscip," in *International Congress on Mathematical Software*, pp. 259–264, Springer, 2016.
3 citations in pages 8, 9, and 23.

[52] R. D. Galvão and C. ReVelle, "A lagrangean heuristic for the maximal covering location problem," *European Journal of Operational Research*, vol. 88, no. 1, pp. 114–123, 1996.
One citation in page 9.

[53] B. Gendron, P.-V. Khuong, and F. Semet, "A lagrangian-based branch-and-bound algorithm for the two-level uncapacitated facility location problem with single-assignment constraints," *Transportation Science*, vol. 50, no. 4, pp. 1286–1299, 2016.
One citation in page 9.

[54] E.-G. Talbi, *Metaheuristics: from design to implementation*, vol. 74. John Wiley & Sons, 2009.
5 citations in pages 10, 14, 42, 45, and 61.

[55] M. Gendreau, G. Laporte, and F. Semet, "Solving an ambulance location model by tabu search," *Location science*, vol. 5, no. 2, pp. 75–88, 1997.
One citation in page 10.

[56] A. T. Murray and R. L. Church, "Applying simulated annealing to location-planning models," *Journal of Heuristics*, vol. 2, no. 1, pp. 31–53, 1996.
One citation in page 10.

[57] S. C. Ho, "An iterated tabu search heuristic for the single source capacitated facility location problem," *Applied Soft Computing*, vol. 27, pp. 169–178, 2015.
One citation in page 10.

[58] F. Ye, Q. Zhao, M. Xi, and M. Dessouky, "Chinese national emergency warehouse location research based on vns algorithm," *Electronic Notes in Discrete Mathematics*, vol. 47, pp. 61–68, 2015.
One citation in page 10.

[59] A. Liefooghe, J. Humeau, S. Mesmoudi, L. Jourdan, and E.-G. Talbi, "On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems," *Journal of Heuristics*, vol. 18, no. 2, pp. 317–352, 2012.
5 citations in pages 10, 46, 48, 57, and 61.

[60] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.
6 citations in pages 11, 12, 14, 23, 25, and 42.

[61] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.
One citation in page 11.

[62] E. Karasakal and A. Silav, "A multi-objective genetic algorithm for a bi-objective facility location problem with partial coverage," *Top*, vol. 24, no. 1, pp. 206–232, 2016.
One citation in page 11.

[63] F. RanjbarTezenji, M. Mohammadi, and S. H. R. Pasandideh, "Bi-objective location-allocation-inventory-network design in a two-echelon supply chain using de novo programming, nsga-ii and nrga," *International Journal of Logistics Systems and Management*, vol. 28, no. 3, pp. 308–337, 2017.
One citation in page 11.

[64] R. Z. Farahani, A. Hassani, S. M. Mousavi, and M. B. Baygi, "A hybrid artificial bee colony for disruption in a hierarchical maximal covering location problem," *Computers & Industrial Engineering*, vol. 75, pp. 129–141, 2014.
2 citations in pages 11 and 45.

[65] B. Jayalakshmi and A. Singh, "A hybrid artificial bee colony algorithm for the cooperative maximum covering location problem," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 2, pp. 691–697, 2017.
2 citations in pages 11 and 45.

[66] S. Davari, M. H. F. Zarandi, and A. Hemmati, "Maximal covering location problem (mclp) with fuzzy travel times," *Expert Systems with Applications*, vol. 38, no. 12, pp. 14535–14541, 2011.
2 citations in pages 11 and 45.

[67] S. Davari, M. H. F. Zarandi, and I. B. Turksen, "A greedy variable neighborhood search heuristic for the maximal covering location problem with fuzzy coverage radii," *Knowledge-Based Systems*, vol. 41, pp. 68–76, 2013.
2 citations in pages 11 and 45.

[68] D. E. Golberg, "Genetic algorithms in search, optimization, and machine learning," *Addion wesley*, vol. 1989, no. 102, p. 36, 1989.
One citation in page 11.

[69] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 28, no. 3, pp. 392–403, 1998.
2 citations in pages 11 and 45.

[70] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," *TIK-report*, vol. 103, 2001.
2 citations in pages 11 and 23.

[71] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.
One citation in page 12.

[72] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints.," *IEEE Trans. Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.
3 citations in pages 13, 23, and 41.

[73] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712–731, 2007.
One citation in page 13.

[74] H. Ishibuchi, R. Imada, Y. Setoguchi, and Y. Nojima, "Performance comparison of nsga-ii and nsga-iii on various many-objective test problems," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pp. 3045–3052, IEEE, 2016.
One citation in page 13.

[75] H. Seada and K. Deb, "U-nsga-iii: A unified evolutionary optimization procedure for single, multiple, and many objectives: Proof-of-principle results," in *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 34–49, Springer, 2015.
One citation in page 13.

[76] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 631–657, 1998.
One citation in page 13.

[77] D. A. Van Veldhuizen, *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations.* PhD thesis, Wright Patterson AFB, OH, USA, 1999. Adviser-Lamont, Gary B.
2 citations in pages 14 and 42.

[78] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.
One citation in page 15.

[79] R. Daniels and C. Mulley, "Explaining walking distance to public transport: The dominance of public transport supply," *Journal of Transport and Land Use*, vol. 6, no. 2, pp. 5–20, 2013.
2 citations in pages 16 and 26.

[80] S. Schmöller, S. Weikl, J. Müller, and K. Bogenberger, "Empirical analysis of free-floating carsharing usage: The munich and berlin case," *Transportation Research Part C: Emerging Technologies*, vol. 56, pp. 34–51, 2015.
One citation in page 16.

[81] C. Willing, K. Klemmer, T. Brandt, and D. Neumann, "Moving in time and space–location intelligence for carsharing decision support," *Decision Support Systems*, vol. 99, pp. 75–85, 2017.
2 citations in pages 17 and 98.

[82] Y. I. Parish and P. Müller, "Procedural modeling of cities," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 301–308, ACM, 2001.
One citation in page 17.

[83] H. R. Anderson and J. P. McGeehan, "Optimizing microcell base station locations using simulated annealing techniques," in *Vehicular Technology Conference, 1994 IEEE 44th*, pp. 858–862, IEEE, 1994.
One citation in page 18.

[84] M. H. Azizan, C. S. Lim, W. L. W. Hatta, and L. C. Gan, "Application of openstreetmap data in ambulance location problem," in *Computational Intelligence, Communication Systems and Networks (CICSyN), 2012 Fourth International Conference on*, pp. 321–325, IEEE, 2012.
One citation in page 18.

[85] G. Boeing, "OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Computers, Environment and Urban Systems*, vol. 65, pp. 126–139, sep 2017.
One citation in page 18.

[86] D. Luxen and C. Vetter, "Real-time routing with openstreetmap data," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '11, pp. 513–516, 2011.
One citation in page 18.

[87] M. Juschten, T. Ohnmacht, V. T. Thao, R. Gerike, and R. Hössinger, "Carsharing in switzerland: Identifying new markets by predicting membership based on data on supply and demand," *Transportation*, pp. 1–24, 2017.
One citation in page 19.

[88] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.
2 citations in pages 21 and 35.

[89] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*, pp. 85–103, Springer, 1972.
One citation in page 21.

[90] R. Raz and S. Safra, "A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np," in *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, (New York, NY, USA), pp. 475–484, ACM, 1997.
2 citations in pages 22 and 36.

[91] C. Spearman, "The proof and measurement of association between two things," *American journal of Psychology*, vol. 15, no. 1, pp. 72–101, 1904.
One citation in page 22.

[92] M. Laumanns, L. Thiele, and E. Zitzler, "Running time analysis of evolutionary algorithms on a simplified multiobjective knapsack problem," *Natural Computing*, vol. 3, no. 1, pp. 37–51, 2004.
One citation in page 23.

[93] D. W. Corne, J. D. Knowles, and M. J. Oates, "The pareto envelope-based selection algorithm for multi-objective optimization," in *International conference on parallel problem solving from nature*, pp. 839–848, Springer, 2000.
One citation in page 23.

[94] T. Koch, *Rapid Mathematical Prototyping*. PhD thesis, Technische Universität Berlin, 2004.
One citation in page 23.

[95] La Ville de Luxembourg, "Etat de la population 2016," 2016. [Online; accessed 15-August-2017].
One citation in page 26.

[96] S. Varrette, P. Bouvry, H. Cartiaux, and F. Georgatos, "Management of an academic hpc cluster: The ul experience," in *Proc. of the 2014 Intl. Conf. on High Performance Computing & Simulation (HPCS 2014)*, (Bologna, Italy), pp. 959–967, IEEE, July 2014.
5 citations in pages 26, 41, 56, 78, and 94.

[97] V. V. Vazirani, *Approximation algorithms*. Springer Science & Business Media, 2013.
One citation in page 36.

[98] H. Li and K. Deb, "Challenges for evolutionary multiobjective optimization algorithms in solving variable-length problems," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2217–2224, IEEE, 2017.
2 citations in pages 37 and 98.

[99] D. E. Goldberg, B. Korb, K. Deb, *et al.*, "Messy genetic algorithms: Motivation, analysis, and first results," *Complex systems*, vol. 3, no. 5, pp. 493–530, 1989.
One citation in page 37.

[100] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective optimization with messy genetic algorithms," in *Symposium on Applied Computing: Proceedings of the 2000 ACM symposium on Applied computing.*, vol. 1, pp. 470–476, Citeseer, 2000.
One citation in page 37.

[101] S. Zhu and L. Xu, "Many-objective fuzzy centroids clustering algorithm for categorical data," *Expert Systems with Applications*, vol. 96, pp. 230–248, 2018.
One citation in page 37.

[102] Z. Zhu, F. Wang, S. He, and Y. Sun, "Global path planning of mobile robots using a memetic algorithm," *International Journal of Systems Science*, vol. 46, no. 11, pp. 1982–1993, 2015.
One citation in page 37.

[103] A. Gad and O. Abdelkhalik, "Hidden genes genetic algorithm for multi-gravity-assist trajectories optimization," *Journal of Spacecraft and Rockets*, vol. 48, no. 4, pp. 629–641, 2011.
One citation in page 37.

[104] R. S. Zebulum, M. Vellasco, and M. A. Pacheco, "Variable length representation in evolutionary electronics," *Evolutionary Computation*, vol. 8, no. 1, pp. 93–120, 2000.
One citation in page 37.

[105] T.-M. Chan, K.-F. Man, K.-S. Tang, and S. Kwong, "A jumping-genes paradigm for optimizing factory wlan network," *IEEE Transactions on Industrial Informatics*, vol. 3, no. 1, pp. 33–43, 2007.
One citation in page 37.

[106] S.-M. Pan and K.-S. Cheng, "Evolution-based tabu search approach to automatic clustering," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 5, pp. 827–838, 2007.
One citation in page 37.

[107] S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, vol. 38, no. 1, pp. 218–237, 2007.
One citation in page 37.

[108] S. Das and S. Sil, "Kernel-induced fuzzy clustering of image pixels with an improved differential evolution algorithm," *Information Sciences*, vol. 180, no. 8, pp. 1237–1256, 2010.
One citation in page 37.

[109] S. Bandyopadhyay and U. Maulik, "Genetic clustering for automatic evolution of clusters and application to image classification," *Pattern recognition*, vol. 35, no. 6, pp. 1197–1208, 2002.
One citation in page 37.

[110] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, jul 2012.
2 citations in pages 38 and 41.

[111] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *Journal of the American Statistical Association*, vol. 47, no. 260, pp. 583–621, 1952.
4 citations in pages 38, 41, 56, and 83.

[112] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.
5 citations in pages 38, 41, 56, 83, and 91.

[113] Y. Yuan, H. Xu, B. Wang, and X. Yao, "A new dominance relation-based evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 16–37, 2015.
One citation in page 41.

[114] L. Paquete, M. Chiarandini, and T. Stützle, "Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study," in *Metaheuristics for multiobjective optimisation*, pp. 177–199, Springer, 2004.
2 citations in pages 45 and 46.

[115] M. M. Paydar and M. Saidi-Mehrabad, "A hybrid genetic-variable neighborhood search algorithm for the cell formation problem based on grouping efficacy," *Computers & Operations Research*, vol. 40, no. 4, pp. 980–990, 2013.
One citation in page 45.

[116] A. Jaszkiewicz, "Genetic local search for multi-objective combinatorial optimization," *European journal of operational research*, vol. 137, no. 1, pp. 50–71, 2002.
One citation in page 45.

[117] K. C. Tan, Y. H. Chew, and L. Lee, "A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows," *Computational Optimization and Applications*, vol. 34, no. 1, p. 115, 2006.
One citation in page 45.

[118] P. Hansen and N. Mladenović, "Variable neighborhood search," in *Handbook of metaheuristics*, pp. 145–184, Springer, 2003.
2 citations in pages 46 and 60.

[119] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973.
One citation in page 47.

[120] E. Taillard, "Some efficient heuristic methods for the flow shop sequencing problem," *European journal of Operational research*, vol. 47, no. 1, pp. 65–74, 1990.
One citation in page 47.

[121] A. Khan and S. Elnikety, "Systems for big-graphs," *Proceedings of the VLDB Endowment*, vol. 7, pp. 1709–1710, aug 2014.
One citation in page 63.

[122] L. Lovász, *Large networks and graph limits*, vol. 60. Providence: American Mathematical Society, 2012.
3 citations in pages 63, 65, and 81.

[123] R. Murphy, J. Berry, W. McLendon, B. Hendrickson, D. Gregor, and A. Lumsdaine, "DFS: A simple to write yet difficult to execute benchmark," in *Proc. IEEE Int. Symp. Workload Characterization*, pp. 175–177, Oct 2006.
One citation in page 63.

[124] B. Kalyanasundaram. and K. R. Pruhs, "Constructing competitive tours from local information," *Theor. Comput. Sci.*, vol. 130, pp. 125–138, aug 1994.
2 citations in pages 63 and 64.

[125] P. Panaite and A. Pelc, "Exploring unknown undirected graphs," *J. Algorithms*, vol. 33, pp. 281–295, nov 1999.
2 citations in pages 64 and 65.

[126] S. Albers and M. R. Henzinger, "Exploring unknown environments," *SIAM J. Comput.*, vol. 29, pp. 1164–1188, jan 2000.
4 citations in pages 64, 65, 66, and 86.

[127] D. Coppersmith, P. Doyle, P. Raghavan, and M. Snir, "Random walks on weighted graphs and applications to on-line algorithms," *J. ACM*, vol. 40, no. 3, pp. 421–453, 1993.
2 citations in pages 64 and 65.

[128] R. Fleischer and G. Trippen, "Experimental studies of graph traversal algorithms," in *Exp. Efficient Algorithms*, pp. 120–133, 2003.
3 citations in pages 64, 65, and 66.

[129] M. Bucolo, R. Caponetto, L. Fortuna, M. Frasca, and A. Rizzo, "Does chaos work better than noise?," *IEEE Circuits Syst. Mag.*, vol. 2, no. 3, pp. 4–19, 2002.
2 citations in pages 64 and 67.

[130] L. Li, Y. Yang, H. Peng, and X. Wang, "An optimization method inspired by "chaotic" ant behavior," *Int. J. Bifurcation Chaos*, vol. 16, pp. 2351–2364, aug 2006.
One citation in page 64.

[131] B. Liu, L. Wang, Y.-H. Jin, F. Tang, and D.-X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons & Fractals*, vol. 25, pp. 1261–1271, sep 2005.
One citation in page 64.

[132] T. Xiang, X. Liao, and K.-W. Wong, "An improved particle swarm optimization algorithm combined with piecewise linear chaotic map," *Appl. Math. Comput.*, vol. 190, no. 2, pp. 1637–1645, 2007.
One citation in page 64.

[133] E. Araujo and L. d. S. Coelho, "Particle swarm approaches using Lozi map chaotic sequences to fuzzy modelling of an experimental thermal-vacuum system," *Appl. Soft Comput.*, vol. 8, no. 4, pp. 1354–1364, 2008.
3 citations in pages 64, 67, and 89.

[134] A. H. Gandomi, G. J. Yun, X.-S. Yang, and S. Talatahari, "Chaos-enhanced accelerated particle swarm optimization," vol. 18, no. 2, pp. 327–340, 2013.
2 citations in pages 64 and 67.

[135] M. Pluhacek, R. Senkerik, and I. Zelinka, "PSO algorithm enhanced with Lozi chaotic map-tuning experiment," in *AIP Conf. Proc.*, vol. 1648, p. 550022, 2015.
2 citations in pages 64 and 67.

[136] M. Rosalie, "Templates and subtemplates of Rössler attractors from a bifurcation diagram," *J. Phys. A: Math. Theor.*, vol. 49, p. 315101, jun 2016.
6 citations in pages 64, 69, 70, 71, 72, and 99.

[137] K. V. Shrikhande, I. M. White, D.-R. Wonglumsom, S. M. Gemelos, M. S. Rogge, Y. Fukashiro, M. Avenarius, and L. G. Kazovsky, "HORNET: A packet-over-WDM multiple access metropolitan area ring network," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, pp. 2004–2016, 2000.
One citation in page 64.

[138] R. Gaudino, A. Carena, V. Ferrero, A. Pozzi, V. De Feo, P. Gigante, F. Neri, and P. Poggiolini, "RINGO: A WDM ring optical packet network demonstrator," in *Eur. Conf. Optical Com. (ECOC'01)*, vol. 4,

pp. 620–621, 2001.
One citation in page 64.

[139] A. Richter, H. Bock, W. Fischler, P. Leisching, P. Krummrich, A. Mayer, R. Neuhauser, J. Elbers, and C. Glingener, "Germany-wide DWDM field trial: transparent connection of a long haul link and a multiclient metro network," in *Optical Fiber Com. Conf.*, p. ML3, 2001.
One citation in page 64.

[140] E. Horvitz and J. Leskovec, "Planetary-scale views on an instant-messaging network," *MSR-TR*, 2007.
One citation in page 64.

[141] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.
2 citations in pages 64 and 80.

[142] L. A. Adamic, "The small world web," in *ECDL*, vol. 99, pp. 443–452, 1999.
One citation in page 64.

[143] S.-H. Yook, Z. N. Oltvai, and A.-L. Barabási, "Functional and topological characterization of protein interaction networks," *Proteomics*, vol. 4, no. 4, pp. 928–942, 2004.
2 citations in pages 64 and 65.

[144] A. Wagner and D. A. Fell, "The small world inside large metabolic networks," *Proc. R. Soc. London, Ser. B*, vol. 268, no. 1478, pp. 1803–1810, 2001.
One citation in page 64.

[145] K. A. Seaton and L. M. Hackett, "Stations, trains and small-world networks," *Physica A*, vol. 339, no. 3, pp. 635–644, 2004.
One citation in page 64.

[146] E. J. Sanz-Arigita, M. M. Schoonheim, J. S. Damoiseaux, S. A. Rombouts, E. Maris, F. Barkhof, P. Scheltens, and C. J. Stam, "Loss of 'small-world' networks in alzheimer's disease: graph analysis of fmri resting-state functional connectivity," *PloS one*, vol. 5, no. 11, p. e13788, 2010.
One citation in page 64.

[147] P. Erdős and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.
One citation in page 65.

[148] S. D. Servetto and G. Barrenechea, "Constrained random walks on random graphs: routing algorithms for large scale wireless sensor networks," in *Proc. ACM Int. Workshop Wireless sensor networks and applications*, pp. 12–21, 2002.
One citation in page 65.

[149] M. Haenggi, J. G. Andrews, F. Baccelli, O. Dousse, and M. Franceschetti, "Stochastic geometry and random graphs for the analysis and design of wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 7, 2009.
One citation in page 65.

[150] M. E. Newman, S. H. Strogatz, and D. J. Watts, "Random graphs with arbitrary degree distributions and their applications," *Phys. Rev. E*, vol. 64, no. 2, p. 026118, 2001.
One citation in page 65.

[151] A. K. Wong and M. You, "Entropy and distance of random graphs with application to structural pattern recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 5, pp. 599–609, 1985.
One citation in page 65.

[152] N. Alon, J. Bruck, J. Naor, M. Naor, and R. M. Roth, "Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs," *IEEE Trans. Inf. Theory*, vol. 38, no. 2, pp. 509–516, 1992.
One citation in page 65.

[153] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proc. IEEE Int. Conf. Rob. Automation*, vol. 2, pp. 116–121, 1985.
One citation in page 65.

[154] S. Koenig, B. Szymanski, and Y. Liu, "Efficient and inefficient ant coverage methods," *Ann. Math. Artif. Intell.*, vol. 31, no. 1-4, pp. 41–76, 2001.
2 citations in pages 65 and 66.

[155] V. Yanovski, I. A. Wagner, and A. M. Bruckstein, "Vertex-ant-walk–a robust method for efficient exploration of faulty graphs," *Ann. Math. Artif. Intell.*, vol. 31, no. 1-4, pp. 99–112, 2001.
2 citations in pages 65 and 66.

[156] M. Rosalie, G. Danoy, S. Chaumette, and P. Bouvry, "From random process to chaotic behavior in swarms of UAVs," in *Proc. ACM Int. Symp. on Design and Analysis of Intelligent Vehicular Networks and Applications (DIVANet 2016)*, pp. 9–15, 2016.
2 citations in pages 65 and 67.

[157] M. Mitzenmacher, "A brief history of generative models for power law and lognormal distributions," *Internet Math.*, vol. 1, no. 2, pp. 226–251, 2004.
One citation in page 65.

[158] P. Holme and B. J. Kim, "Growing scale-free networks with tunable clustering," *Phys. Rev. E*, vol. 65, no. 2, p. 026107, 2002.
One citation in page 65.

[159] R. Albert, H. Jeong, and A.-L. Barabási, "Internet: Diameter of the world-wide web," *Nature*, vol. 401, no. 6749, pp. 130–131, 1999.
One citation in page 65.

[160] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *ACM SIGCOMM Computer Com. Review*, vol. 29, pp. 251–262, 1999.
One citation in page 65.

[161] A.-L. Barabasi and Z. N. Oltvai, "Network biology: understanding the cell's functional organization," *Nat. Rev. Genet.*, vol. 5, no. 2, p. 101, 2004.
One citation in page 65.

[162] R. Albert, "Scale-free networks in cell biology," *J. Cell Sci.*, vol. 118, no. 21, pp. 4947–4957, 2005.
One citation in page 65.

[163] H. Choset *Ann. Math. Artif. Intell.*, vol. 31, no. 1/4, pp. 113–126, 2001.
One citation in page 65.

[164] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," *IEEE Trans. Rob. Autom.*, vol. 7, no. 6, pp. 859–865, 1991.
One citation in page 65.

[165] S. Koenig and Y. Smirnov, "Graph learning with a nearest neighbor approach," in *Proc. ACM Annual Conf. Computational learning theory*, pp. 19–28, 1996.
One citation in page 65.

[166] S. Kwek, "On a simple depth-first search strategy for exploring unknown graphs," in *Lect. Notes Comput. Sci.*, pp. 345–353, 1997.
One citation in page 66.

[167] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg, "Graph exploration by a finite automaton," *Theor. Comput. Sci*, vol. 345, no. 2, pp. 331–344, 2005.
One citation in page 66.

[168] S. Hong, T. Oguntebi, and K. Olukotun, "Efficient parallel graph exploration on multi-core CPU and GPU," in *Proc. IEEE Int. Conf. Parallel Architectures and Compilation Techniques*, pp. 78–88, oct 2011.
One citation in page 66.

[169] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein, "Distributed covering by ant-robots using evaporating traces," *IEEE Trans. Rob. Autom.*, vol. 15, no. 5, pp. 918–933, 1999.
One citation in page 66.

[170] D. Ilcinkas, "Setting port numbers for fast graph exploration," *Theor. Comput. Sci*, vol. 401, no. 1-3, pp. 236–242, 2008.
2 citations in pages 66 and 85.

[171] Y. Higashikawa, N. Katoh, S. Langerman, and S. Tanigawa, "Online graph exploration algorithms for cycles and trees by multiple searchers," *J. Comb. Opt.*, vol. 28, no. 2, pp. 480–495, 2014.
One citation in page 66.

[172] K. Dentler, C. Guéret, and S. Schlobach, "Semantic web reasoning by swarm intelligence," in *Proc. Int. Workshop Scalable Semantic Web Knowledge Base Systems (SSWS)*, p. 1, 2009.
One citation in page 66.

[173] I. Tiddi, M. d'Aquin, and E. Motta, "Walking linked data: a graph traversal approach to explain clusters," in *Proc. Int. Conf. Consuming Linked Data-Volume 1264*, pp. 73–84, 2014.
One citation in page 66.

[174] A. Piyatumrong, P. Bouvry, F. Guinand, and K. Lavangnananda, "A study of token traversal strategies on tree-based backbones for mobile ad hoc - delay tolerant networks," in *Proc. IEEE Int. Conf. Ultra Modern Telecommunications (ICUMT)*, pp. 1–8, 2009.
One citation in page 67.

[175] A. Piyatumrong, P. Ruiz, P. Bouvry, F. Guinand, and K. Lavangnananda, "Token traversal strategies of a distributed spanning forest algorithm in mobile ad hoc - delay tolerant networks," in *Proc. Int. Conf. Advances in Information Technology (IAIT)*, vol. 55 of *Communications in Computer and Information Science*, pp. 96–109, Springer, 2009.
One citation in page 67.

[176] A. H. Gandomi, X.-S. Yang, S. Talatahari, and A. Alavi, "Firefly algorithm with chaos," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 18, pp. 89–98, jan 2013.
One citation in page 67.

[177] D. Liu and G. Chen, "Hybrid algorithm for ant colony optimization based on chaos technique," in *Proc. IEEE Int. Conf. Natural Computation*, vol. 5, pp. 2628–2632, 2010.
One citation in page 67.

[178] R. Lozi, "Emergence of randomness from chaos," *Int. J. Bifurcation Chaos*, vol. 22, no. 02, p. 1250021, 2012.
One citation in page 67.

[179] R. Lozi, "Un attracteur étrange (?) du type attracteur de Hénon," *J. Phys.*, vol. 39, no. C5, pp. C5–9, 1978.
One citation in page 68.

[180] O. E. Rössler, "An equation for continuous chaos," *Phys. Lett. A*, vol. 57, no. 5, pp. 397–398, 1976.
2 citations in pages 68 and 69.

[181] V. Botella-Soler, J. Castelo, J. Oteo, and J. Ros, "Bifurcations in the Lozi map," *J. Phys. A: Math. Theor.*, vol. 44, no. 30, p. 305101, 2011.
3 citations in pages 68, 70, and 89.

[182] J. Sprott and C. Li, "Asymmetric bistability in the Rössler system," *Acta Phys. Pol. B*, vol. 48, no. 1, p. 97, 2017.
One citation in page 69.

[183] J. Starrett, "Non-strange chaotic attractors equivalent to their templates," *Dyn. Sys.*, vol. 27, pp. 187–196, jun 2012.
2 citations in pages 73 and 99.

[184] S. Mangiarotti and C. Letellier, "Topological analysis for designing a suspension of the Hénon map," *Phys. Lett. A*, vol. 379, pp. 3069–3074, dec 2015.
2 citations in pages 73 and 99.

[185] T. Hamaizia, R. Lozi, and N.-e. Hamri, "Fast chaotic optimization algorithm based on locally averaged strategy and multifold chaotic attractor," *Appl. Math. Comput.*, vol. 219, no. 1, pp. 188–196, 2012.
One citation in page 87.

[186] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM T. Model Comput. S.*, vol. 8, no. 1, pp. 3–30, 1998.
One citation in page 94.

[187] I. Koyuncu, A. T. Ozcerit, and I. Pehlivan, "Implementation of FPGA-based real time novel chaotic oscillator," *Nonlinear Dynamics*, vol. 77, pp. 49–59, feb 2014.
One citation in page 95.

[188] C. Lo, "Automated population and dwelling unit estimation from high-resolution satellite images: a gis approach," *Remote Sensing*, vol. 16, no. 1, pp. 17–34, 1995.
One citation in page 98.

[189] S. Ural, E. Hussain, and J. Shan, "Building population mapping with aerial imagery and gis data," *International Journal of Applied Earth Observation and Geoinformation*, vol. 13, no. 6, pp. 841–852, 2011.
One citation in page 98.

[190] B. George, S. Kim, and S. Shekhar, "Spatio-temporal network databases and routing algorithms: A summary of results," in *International Symposium on Spatial and Temporal Databases*, pp. 460–477, Springer, 2007.
One citation in page 98.

[191] M. Aziz-Alaoui, C. Robert, and C. Grebogi, "Dynamics of a Hénon–Lozi-type map," *Chaos, Solitons & Fractals*, vol. 12, no. 12, pp. 2323–2341, 2001.
One citation in page 99.