Parallel Genetic Algorithms for Multi-Criteria and Diverse Path Routing on Large Transportation Networks

Maynooth University
National University of Ireland Maynooth

Harish Sharma

A THESIS SUBMITTED FOR THE DEGREE OF
Doctor of Philosophy

DEPARTMENT OF COMPUTER SCIENCE

MAYNOOTH UNIVERSITY

September 17, 2025

Head of Department: Dr. Aidan Mooney

Supervisors: Dr. Edgar Galván and Dr. Peter Mooney

Abstract

Finding an optimal path between a source and target node in a graph or network is a well-established problem in Computer Science. This problem has been studied extensively, with a range of algorithms developed to accommodate diverse objectives, domains, and real-world application. Classical approaches, such as Dijkstra's algorithm, are well-known for computing a single optimal path based on one objective, variable or criteria. This is typically distance or time and algorithms like this are not designed to handle multiple objectives or generate a usable set of diverse alternative routes. In the case of transportation networks, **as urban environments become more complex and mobility expectations increase, the ability to consider multiple objectives and offer diverse routing options on road networks has shifted from a theoretical challenge to a practical necessity**. A real-world example of this might be a driver who wishes to travel between two points in a road network and is seeking a route which minimises travel distance, maximises the number of EV charging stations available on the route, and minimises the overall cost of the journey in terms of toll charges. **Traditional algorithms (such as Dijkstra's) cannot solve scenarios where a path (or route) has multiple objectives to optimise**. While such methods are effective at returning a single shortest path, many real-world applications require a set of optimal (or near-optimal) alternative paths. Algorithms such as the k-shortest paths algorithm improve upon, and extend, earlier approaches by producing multiple paths of equal or near-equal length between two points. Nevertheless, **most algorithmic approaches remain limited to single-objective optimisation and do not guarantee diversity among the final set of generated paths**.

To address these limitations, population-based methods such as **Evolutionary Algorithms (EAs) have emerged as promising alternatives, capable of producing diverse and/or multi-objective solutions suited for the challenges outlined above**. In this thesis, we make several contributions to this rapidly growing research field by introducing novel computational frameworks capable of generating diverse, multi-objective routes or paths in large-scale real-world transportation networks. This thesis introduces **a novel framework for solving the K-Most Diverse Near-Shortest Paths (KMDNSP)** problem, which aims to find a set of k near-optimal paths that maximise path diversity while having near-optimal overall route lengths. To achieve this we have used a special class of EAs known as Parallel Genetic Algorithms (PGAs). The proposed PGA, named the **MultiPath Island-Based Genetic Algorithm (MIBGA)**, serves to facilitate vehicle routing mechanisms for the KMDNSP problem. This effectively balances the need for path diversity and optimality through innovative migration and adaptive selection strategies. Empirical results presented confirm the efficacy of MIBGA and demonstrate how its design supports the generation of diverse, near-shortest paths.

Building on the island structure of MIBGA and integrating it with a global parallelisa-

tion model, we present the **Parallel Optimal-route Search (POS)** as a multi-objective routing framework. POS is designed to address the real-world routing problems in the context of general public mobility needs by providing end users with a collection of optimal paths optimised across multiple criteria. To achieve this, **a novel fitness metric called Positional Count (PC)** is introduced to mitigate the bias caused by longer routes potentially offering better values for certain objectives. PC assigns weights to features based on their position and density along the route, shifting the focus from sheer quantity to the contextual relevance and spatial distribution of features along the route.

To demonstrate the robustness of our methods across different network topologies and scales, we perform evaluation on complex real-world road networks rather than the simpler synthetic networks commonly used in much of the key literature in this research. The smallest network tested is from Arizona, with 834 nodes and $1,547$ edges, while the largest is Texas, featuring $10,886$ unique nodes and $24,464$ edges. **Our framework for solving the KMDNSP problem, the development of MIBGA and POS demonstrate significant contributions to the area of path optimisation on real-world transportation networks**. A number of opportunities for future work are presented at the end of the thesis and these address theoretical, practical, and application-oriented aspects of the proposed methods and metrics in this work. All software code and network data used in our research is available as open-source resources to enhance the opportunities for reproducibility and reuse.

Declaration

I, Harish Sharma, declare that this thesis titled **Parallel Genetic Algorithms for Multi-Criteria and Diverse Path Routing on Large Transportation Networks** and the work presented in it are my own. I confirm that

- This work was conducted entirely during the PhD candidacy in Computer Science at Maynooth University.
- If any part of this thesis has previously been submitted toward a degree or other qualification at this or any other institution, it is clearly stated herein.
- Where I have consulted the published work of others, this is clearly acknowledged and cited.
- Where I have quoted from the work of others, the source is always provided. Apart from such quotations, all text in this thesis is entirely my own.
- I used external tools (Grammarly and ChatGPT) strictly for grammar, syntax, and spelling checks. These tools did not contribute to the original research, ideas, analytical conclusions, or substantive content of this thesis.
- The programming and coding tasks in this thesis were performed using Python, primarily within Google Colab and Jupyter Notebooks. Computational experiments were executed on the Irish Centre for High-End Computing (ICHEC) infrastructure, leveraging their high-performance computing resources.
- All figures in this thesis were generated by me using Pythons Plotly and Seaborn libraries, Microsoft PowerPoint, Google Maps, or Mermaid Chart for diagramming. Where any figure has not been generated by me, this is clearly stated.
- I have acknowledged all significant sources of assistance in the completion of this thesis.

_____

Signature

_____

Date

Acknowledgements

This thesis represents the most meaningful achievement of my life so far, and I would like to extend my sincere thanks to all those who helped make it possible. This honour belongs not only to me, but also to everyone who supported and guided me throughout this journey. I begin by thanking my incredible supervisors, Dr. Edgar Galván and Dr. Peter Mooney, whose expert guidance, unwavering support, and invaluable feedback were instrumental in shaping this work. Their belief in me and dedication to my development made this accomplishment possible. I would also like to express my sincere appreciation to Dr. Joseph Timoney for his willingness to support me and my colleagues whenever we needed assistance.

To my family, thank you for your unconditional love and unwavering support. Your faith in me has been a constant source of strength. A special note of gratitude goes to my mother, Pinki Sharma, for whom my education has always been a top priority. Her sacrifices, resilience, and constant encouragement have played a vital role in helping me reach this milestone. I also want to express my heartfelt gratitude to my late grandparents, Smt. Krishnawati and Shri Dwarka Das. Though I lost my grandmother at a young age and never had the chance to meet my grandfather, their blessings and values have remained a quiet but meaningful presence in my life.

I would also like to thank my friends, who made this journey not only easier but also filled with unforgettable moments. Shubham, my housemate and gym buddy, deserves special mention, not only for the endless laughs but also for supporting me financially when I needed it most. Azhar, thank you for patiently teaching me how to code during our master's program, you truly laid the foundation. Parry and Rishabh, your advice has always been sharp, timely, and deeply appreciated. Azeema, thank you for listening to all my complaints without ever judging, and Swati, thank you for taking care of me when I was unwell, it meant more than words can say. Pratik, your web series recommendations and endless stream of unnecessary Instagram reels brought much needed breaks and smiles. Sandy, thank you for always cooking something delicious, it made life so much better. Sankhadip, your push to apply for this PhD position was a turning point. And of course, thank you to Durgesh for getting me the wrong laptop and doing absolutely nothing, but still somehow managing to be part of it all.

Particularly, I am grateful for all the people who stood by me during moments of doubt and fatigue. Rare friendships like these turned countless stressful nights into memories filled with calm and laughter. Amid all the chaos of deadlines and revisions, their kindness reminded me to breathe and keep faith. Beyond the research itself, these moments of warmth and humour were the true treasures of this PhD. Having such people around made every challenge lighter, and their presence will stay with me forever.

Special thanks to the cast of Friends and The Big Bang Theory for providing countless moments of laughter and comfort during the long, quiet nights of writing. A heartfelt

mention to my favourite characters, Chandler Muriel Bing (the late Matthew Perry) and Bernadette Rostenkowski (Melissa Rauch), whose humour, warmth, and wit made even the toughest days lighter.

Finally, I express my deepest gratitude to Lord Ram for his strength and guidance, and to Lord Hanuman for his unwavering protection. Their blessings have been a source of courage and focus in my life. With folded hands, I say *Jai Siya Ram! Satnam Shri Waheguru! Jai Hanuman!*

List of Publications

[1] Sharma, H., Mooney, P. and Galván, E., 2023. A method for creating complex real-world networks using ESRI Shapefiles. MethodsX, 11, p.102426, Link to article.

[2] Sharma, H., Galván, E. and Mooney, P., 2025. MultiPath Island-Based Genetic Algorithm for the K-Most Diverse Near-Shortest Paths. Information Sciences, p.122495, Link to article.

[3] Sharma, H., Galván, E. and Mooney, P., 2025. A parallel genetic algorithm for multi-criteria path routing on complex real-world road networks. Applied Soft Computing, 170, p.112559, Link to article.

[4] Sharma, H., Mooney, P. and Galván, E., 2025. GeoGraphNetworks: Shapefile-Derived Datasets for Accurate and Scalable Graphical Representations, Link to dataset.

Contents

Part I: Motives, Background and Literature Review **1**

1 Introduction and Motivation **2**
1.1 Introduction . . . 2
1.2 Motives . . . 3
1.3 Research goals . . . 4
1.4 Reproducibility and data access . . . 4
1.5 Thesis outline . . . 5

2 Literature Review and Related Work **8**
2.1 Introduction . . . 8
2.2 Embedding routing constraints in graph networks . . . 8
2.3 The Vehicle Routing Problem (VRP) . . . 10
2.4 Problems in existing vehicle routing systems . . . 13
2.5 Routing where multiple suitable paths exists . . . 15
2.6 Evolutionary Algorithms (EAs) . . . 18
2.6.1 Genetic Algorithms (GAs) . . . 19
2.6.2 Multi-Objective Genetic Algorithms (MOGAs) . . . 20
2.6.3 Parallel Genetic Algorithms (PGAs) . . . 23
2.7 GAs for path routing . . . 25
2.7.1 Algorithm type . . . 28
2.7.2 Network size . . . 30
2.7.3 Encoding method . . . 30
2.7.4 Crossover method . . . 31
2.7.5 Mutation method . . . 34
2.7.6 Selection method . . . 35
2.8 Identified gaps in the literature . . . 36

Part II: Contributions of this Thesis. **38**

3 Experimental Setup **39**
3.1 Introduction . . . 39
3.2 Geographical systems . . . 39
3.3 Working with graph data structures . . . 42

3.3.1 Data gathering . . . 42
3.3.2 Pre-processing . . . 43
3.3.3 Transformation method . . . 45
3.4 Network repository overview . . . 47
3.5 Validation of generated networks . . . 50
3.6 Usage notes . . . 51
3.7 Overall software implementation details . . . 52
3.8 Chapter summary . . . 53

4 MultiPath Island-Based Genetic Algorithm for the K-Most Diverse Near-Shortest Paths 54
4.1 Introduction . . . 54
4.2 K-Most Diverse Near-Shortest Paths (KMDNSP) . . . 55
4.3 MultiPath Island-Based Genetic Algorithm (MIBGA) . . . 57
4.3.1 Genetic representation of solution paths . . . 58
4.3.2 Mending function for solution paths . . . 59
4.3.3 Fitness function . . . 59
4.3.4 Island formation strategy . . . 60
4.3.5 Migration strategy . . . 62
4.3.6 Offspring generation process . . . 63
4.3.7 Loop-Free Path-Composer: With crossover . . . 64
4.3.8 Loop-Free Path-Composer: Mutation integrated with crossover . . . 64
4.3.9 AvgIslandFit: A multi-step selection process . . . 66
4.3.10 Illustration of the overall MIBGA approach . . . 66
4.4 Algorithm description and configuration . . . 67
4.5 Experimental results . . . 69
4.5.1 Comparative study on the proportion of non-timeout instances . . . 70
4.5.2 Comparative study on the generation of most diverse path sets . . . 71
4.5.3 Comparative study on execution times of the algorithms . . . 74
4.5.4 Discussion on experimental results . . . 75
4.6 Practical applications of MIBGA . . . 76
4.7 Chapter summary . . . 77

5 Parallel Optimal-route Search for Multi-Criteria Vehicle Routing 79
5.1 Introduction . . . 79
5.2 Parallel Optimal-route Search (POS) . . . 80
5.2.1 Fitness evaluation metric . . . 80
5.2.2 Island formation strategy . . . 83
5.2.3 Offspring generation strategy . . . 86
5.2.4 Selection process . . . 87
5.2.5 Outline of the overall process . . . 87
5.3 Performance and evaluation metrics . . . 89
5.4 Algorithm description and configuration . . . 91

5.5 Experimental results . . . 94
5.5.1 Comparative studies: Genetic operators NBCPM vs. LFPC . . . 94
5.5.2 Comparative studies: Algorithms NSGA-II, NSGA-III, POS . . . 98
5.5.3 Statistical analysis . . . 101
5.6 Impact of our proposed fitness metric . . . 102
5.7 Practical applications of POS . . . 103
5.8 Chapter summary . . . 104

6 Conclusions and Future work **106**
6.1 Thesis overview . . . 106
6.2 Key contributions from this research . . . 106
6.3 Comprehensive empirical insights from key contributions . . . 109
6.4 Description of auxiliary computational challenges in this research . . . 110
6.5 Identified Limitations . . . 111
6.6 Conclusions and Future Work . . . 113

References **117**

List of Figures

2.1 An example of paths generated using Google Maps. 14
2.2 An example highlighting the limitation of shared edges among routes. . . . 16
2.3 Generic framework of a genetic algorithm. 20
2.4 Crowding Distance, Improved Crowding Distance, and Reference Points Methods. 22
2.5 Types of PGAs. 24
2.6 Sample graph illustrating a valid path, an invalid path, and a valid path with a loop. Each node is labelled with a unique ID, and different colours are used to distinguish between the paths. 26
2.7 Crossover operation generating one valid and one invalid path. 27
2.8 Mutation operation generating one valid and one invalid path. 27
2.9 Node Based Crossover (NBC). 32
2.10 Path Mutation (PM). 34

3.1 Example of a polyline representation as an edge in a graph. 40
3.2 Texas road network representation before and after cleaning. 44
3.3 Complete network transformation process. 46
3.4 Network transformation example. 46
3.5 Texas road network representation in ESRI Shapefile and NetworkX graph. 47
3.6 Additional networks used in this thesis. 48
3.7 Great Britain (GB) river network. 49
3.8 Validation of generated networks. 51

4.1 Undirected weighted graph. 57
4.2 An example of a candidate path. 58
4.3 An example of a path being repaired. 59
4.4 Migration method in MIBGA. 63
4.5 An example of LFPC operation using only Crossover. 65
4.6 Flowchart illustrating the overall process of MIBGA. 67
4.7 MIBGA: Non-Timeout Instances. 70
4.8 MIBGA: Variation in average Div(P_{KMDNSP}). 71
4.9 MIBGA: Variation in average execution time (in Seconds). 74
4.10 MIBGA: Average number of paths found. 76

5.1 POS: Graph with amenities. 83

5.2 POS: Island Formation Process. 85
5.3 Proposed POS algorithm following a hybrid model. 88
5.4 Example illustrating the distribution of non-dominated solutions in the search space. 96
5.5 Variation in average hypervolume for simulations using the NBCPM and LFPC operator. 97
5.6 Variation in average hypervolume for the simulations of NSGA-II, NSGA-III, and POS. 99
5.7 Illustration of slow convergence behaviour in NSGA-III. 101

List of Tables

2.1 Literature review overview. 9
2.2 List of dissimilarity metrics. 17
2.3 Key terms and definitions in Evolutionary Algorithms (EAs). 18
2.4 Summary of literature review of GAs for path routing problems. 29

3.1 Sample format for an edge list dataset. 47

4.1 MIBGA: Components of the genetic algorithms used in the study. 68
4.2 MIBGA: Dissimilarity evaluation results. 72

5.1 POS: Practical example of our fitness metric. 82
5.2 POS: List of parameters used for NSGA-II, NSGA-III, and the POS algorithm. 93
5.3 POS: Performance comparison between NSGA-II, NSGA-III, and the POS algorithm. 95

Part I

Motives, Background and Literature Review.

Chapter 1

Introduction and Motivation

1.1 Introduction

Given any two nodes S and T in a graph or network, the process of finding an optimal path relating to particular objectives is called path routing or path finding. It could be considered as one of the oldest and the most studied problem in the history of graph theory, with applications in numerous fields including transportation (Meng et al. 1999, Kechagiopoulos & Beligiannis 2014), logistics (Zhang et al. 2015), quality-of-service (QoS) (Rocha et al. 2011, Lu & Zhu 2013) robotics (Soueres & Laumond 1996, Bui et al. 1994, Karur et al. 2021), and others. Distance is typically considered as the primary objective for optimisation in path routing problems. However, other objectives such as time, cost, location-based constraints, and others, can also be taken into account, even if they cannot be expressed as a single numerical quantity for each edge in the network. With distance as the metric to minimise, the problem of finding an optimal path between two given nodes is called the Shortest Path (SP) routing problem.

Traditional Vehicle Routing (VR) systems have employed conventional algorithms like Dijkstra's to return the shortest path between a pair of source and target node (Peyer et al. 2009). With a complexity of $O((n+m)\ log\ n)$, Dijkstra's algorithm (EW 1959) return a single optimal solution, optimised for a single objective. However, in situations where multiple factors can influence navigation choices, usually just one shortest path or a path that is very close to the shortest one is not enough for most applications. For example, the shortest path between two nodes in a network might not be the fastest path due to several complicated scenarios like traffic tie-ups, congestion, blockages, and others. Therefore, we believe it would be better to have multiple paths returned rather than just one in the case of VR on road networks. **One way to achieve this goal is to provide the end user with a selection of multiple optimal solutions optimised on multiple objectives to choose from.** In such cases, multiple optimal paths will allow users to make personalised decisions (Liu, Jin, Yang & Zhou 2017) while distributing traffic across different roads. This helps prevent the overloading of specific routes, reduces congestion and delays, and minimises environmental impact.

1.2 Motives

Over the years, numerous algorithms like depth-first search (Tarjan 1972), A* (Hart et al. 1968), Bellman-Ford (Bellman 1958), Dijkstra's (EW 1959), and others have been used to find the Shortest Path (SP) in a network with variations in application focus such as navigation systems (Wen & Hsu 2005, Yin & Yang 2013, Liu, Zhou, Wu, Long & Wang 2017), Mobile Ad-hoc networks (Bhardwaj & El-Ocla 2020, Darwish et al. 2020), Internet protocols (Gonen & Louis 2006, Monita et al. 2020), maps (Prasetya et al. 2020, Rachmawati & Gustin 2020), and more.

VR is arguably one of the most prominent type of routing that depends heavily on these SP algorithms. However, the growing complexity of real-world traffic conditions has led researchers to move beyond SP algorithms and explore more complex routing solutions that optimise for travel time, either as a single objective or alongside other multiple objectives, multiple paths, multiple destinations, and additional practical factors. Given the need to generate and optimise multiple solutions across several objectives, **our aim is to demonstrate the efficiency, flexibility, and adaptability of Evolutionary Algorithms (EAs)** for solving distinct Vehicle Routing Problems (VRPs) across real-world networks of varying sizes and topologies.

Table 2.4 (page 29) summarises several EAs from the literature that have been applied to a broad range of path routing problems (not necessarily limited to vehicle routing). However, despite the widespread emphasis on using EAs to generate multiple candidate solutions, they rarely address the importance of producing diverse (less overlapping) routes. To address this issue, **we aim to introduce a EA based method for the K-Most Diverse Near-Shortest Paths (KMDNSP) (Häcker et al. 2021) problem, which aims to recommend a set of K alternative paths that satisfy a given length constraint while maximizing dissimilarity among them.** Leveraging the exceptional ability of EAs to generate diverse and high-quality solutions, our proposed methodology aims to provide users with a diverse set of optimal paths to choose from. This is significant because enhancing route diversity can improve system robustness, prevent congestion, and promote better load balancing across the network.

Section 2.3 of Chapter 2 provides a comprehensive review of established VRPs, discussing their constraints, problem statements, and the typical algorithms employed to solve them. From this review, one notable issue that emerges is the lack of VRP variants specifically addressing the broader needs of the general public. In many real-world applications, users may prioritise different aspects of a route, such as proximity to amenities like pubs, hotels, or charging stations, alongside minimising path length. A major challenge in pursuing such objectives is developing an appropriate fitness metric. Consider, for instance, the problem of travelling from a source node S to a target node T in a network, with the dual goals of minimising path length and maximising the number of charging stations along the way. While routes optimised for path length will naturally have shorter distances, those optimised for maximising amenities like charging stations may result in longer paths, as they tend to encounter more stations. This trade-off is com-

monly addressed in Tourist Trip Design Problems (TTDP) (Choachaicharoenkul et al. 2022). Leveraging the spatial distribution of desired amenities along the route, **our aim is to introduce a VR method that maximises amenities while penalising longer paths, providing a more holistic and user centric approach.**

1.3 Research goals

The primary objective of this research is to explore, how a special class of EAs called Parallel Genetic Algorithms (PGAs) can be leveraged to generate diverse (less overlapping) and optimised routing solutions on real-world networks. To achieve this, the following research questions are formulated:

- How can Island-Based Genetic Algorithms (IBGAs) be employed to provide an efficient computational approach to the KMDNSP problem, on large and complex real-world road networks?
- How can spatial information, including geographic coordinates and distances be used to mathematically formulate a fitness metric that penalises path length while maximising the number of amenities encountered along the path?
- How can Multi-Objective Genetic Algorithms (MOGAs) be employed to provide an efficient computational approach to multi-criteria VRP, on large and complex real-world road networks?
- Demonstrate that our introduced PGAs offers faster convergence and greater scalability, computationally outperforming other state of the art GAs on selected routing problems.

By addressing these research questions, this thesis aims to contribute to the field of EAs and path routing, particularly in the context of alternate and multi-criteria VR.

1.4 Reproducibility and data access

To ensure transparency and reproducibility, all network datasets detailed in Chapter 3 are publicly available on Figshare at GeoGraphNetworks. The code for generating graph network representations in Python using NetworkX is accessible on GitHub at GeoGraph-Networks. Additionally, visual representations of each network are provided to illustrate the scale of each graph.

The results of our study, introducing the MultiPath Island-Based Genetic Algorithm (MIBGA) for the K-Most Diverse Near-Shortest Paths (KMDNSP) problem in Chapter 4, have been made publicly available on GitHub at MIBGA. The experimental study was carried out on three real-world road networks comprising primary and secondary roads from Arizona, Washington, and Kansas. Along with these networks, Jupyter notebooks

containing the complete results, including analysis and significance testing, are also provided.

The results of our study, which introduces the Parallel Optimal-route Search (POS) algorithm for multi-criteria VR in Chapter 5, have been made publicly available on GitHub at POS. A folder named Network is provided, containing four real-world road networks in edge list format, along with the code needed to create multi-feature graph networks in NetworkX format. Jupyter notebooks detailing, the data preparation steps (network formation along with the code), equations used, fitness metric, mending function, and significance testing are also provided. Results for each network are provided in separate folders.

1.5 Thesis outline

This thesis is structured in two main parts.

- Part I provides the motivation for the study, relevant background, and a review of related literature. The structure of this part is as follows:
  - The present chapter has outlined the research motivations, objectives, and goals. Furthermore, we have also highlighted that existing approaches struggle to balance multiple practical constraints, reinforcing the need for new and more adaptable routing strategies.
  - **Chapter 2 (Literature Review and Related Work)** presents a comprehensive and insightful discussion introducing the foundational concepts in routing, alternate routing, Evolutionary Algorithms (EAs), and others. First, the chapter introduces routing constraints and policies, including their mathematical formulations on graph networks. It then reviews established VRPs, discussing their constraints, problem statements, and typical algorithms, followed by a discussion of the challenges and limitations present in existing VR systems. The chapter also explores routing methods that generate multiple suitable paths, detailing the key algorithms, associated challenges, and potential solutions. Following this, it provides detailed descriptions of EAs, GAs, PGAs, and MOGAs. Finally, it reviews GA-based methodologies for path routing and concludes by identifying the primary gaps in the literature that motivate the research in this thesis.
- Part II outlines the core contributions of this thesis and is structured as follows:
  - Given the importance of traditional routing domains, graphical networks that resemble real-world topologies are essential for this study. In **Chapter 3**, we introduce an **open-source**, Python-based workflow for constructing graphical representations directly in NetworkX using raw Shapefile data. The method describes the entire process, including data collection, transformation, and a

data cleaning strategy that reduces resource usage while preserving the structural integrity of the graphs. This makes the workflow reusable, transparent, and adaptable to any geographic region, allowing researchers to generate large-scale, reproducible networks. Furthermore, we introduce GeoGraphNetworks (Sharma et al. 2024), a benchmark dataset repository for scalable, spatially accurate graphical representations of the road and rail networks of the United States (USA) and the road and river networks of Great Britain (GB). GeoGraphNetworks facilitates network generation across multilingual environments, supporting a wide range of research and applications.

– Next, in **Chapter 4**, we introduce the **MultiPath Island-Based Genetic Algorithm (MIBGA) for the K-Most Diverse Near-Shortest Paths (KMDNSP) problem**. As one of the first metaheuristic approaches to address the KMDNSP problem, this algorithm aims to advance the field of alternate routing by generating a set of diverse (less overlapping) paths, each of which is constrained to fall within a specified GAP (%) from the overall shortest path. The structure of our MIBGA enabled the integration and use of strategically designed novel features, like a Migration strategy to facilitate the exchange of strong individuals among the islands, thereby preserving diversity while accelerating the convergence of the other islands and a metric AvgIslandFit, which serves as a self-adjusting selection strategy for GAs utilising the island model. New paths for evaluation are generated using a novel genetic operator referred to as Loop-Free Path-Composer (LFPC) which yields a more diverse set of solutions in contrast to commonly used Node Based Crossover and Path Mutation (NBCPM) (described in Chapter 2). The performance of MIBGA is evaluated against two established genetic algorithm approaches proposed by Ahn & Ramakrishna (2002) and Qiongbing & Lixin (2016), using large and complex real-world road networks from Arizona, Washington, and Kansas. Experimental results are presented to quantitatively compare the algorithms in terms of their ability to identify a set of K paths that satisfy the length constraint, maximise path diversity, within a specified time, thereby enhancing the overall usability.

– With **Chapter 5**, we introduce our **Parallel Optimal-route Search (POS)** algorithm designed for the multi-criteria VRP. POS follows a hybrid model incorporating a Global Parallelisation mechanism, with Island-Based model. Similar to MIBGA, POS aims to enhance the existing routing systems by offering users the ability to choose their preferred path from a set of optimal paths optimised on multiple objectives. We consider a total of four objectives including Distance as the metric to minimise while maximising the presence of three specific amenities. To mitigate the influence of additional amenities on longer routes, the objectives are optimised using a novel fitness metric called Positional Count designed to tackle the VRP in the context of general public needs, optimising paths by balancing proximity to amenities such as pubs,

hotels, or charging stations while minimising path length. The developed approach, is validated on four large-scale real-world road networks (described in Chapter 3). The performance of POS is evaluated in comparison to the well-known Non-dominated Sorting Genetic Algorithm II and III (Deb et al. 2002, Deb & Jain 2013). Additionally, a systematic study comparing the performance of our proposed LFPC operator against the traditional NBCPM operators is presented in this chapter.

– **Chapter 6** concludes this thesis by summarising the results and elaborating on the key findings regarding the use of EAs for path routing. It also discusses the challenges encountered and outlines potential avenues for improving alternate and multi-criteria routing systems, building upon the insights gained from this work.

Chapter 2

Literature Review and Related Work

2.1 Introduction

This chapter presents a systematic literature review aligned with the thesis objectives, providing essential background to understand the complexity of our work. Table 2.1 provides an overview of the key topics reviewed in this chapter, along with their respective sections and page locations. Section 2.2 introduces routing constraints and policies with mathematical formulations on graph networks. Section 2.3 reviews established Vehicle Routing (VR) problems, their constraints, problem statements, and typical algorithms. Section 2.4 is provided to describe the problems in existing VR systems. Section 2.5 covers routing methods that yield multiple suitable paths, detailing key algorithms, challenges, and potential solutions. In Section 2.6, we define Evolutionary Algorithms (EAs) and Genetic Algorithms (GAs) including Parallel GAs and Multi-Objective GAs. Section 2.7 explores GA based methodologies for path routing. Finally, Section 2.8 highlights the primary gaps in the literature.

2.2 Embedding routing constraints in graph networks

Given a directed graph network represented as $G = (N, E)$, where G denotes the graph, N is the set of all nodes and E is the set of all edges (Borgatti & Everett 2006). Each path p generated in the network can be represented using the adjacency matrix $P_{i,j}$ with all diagonal elements as zero and all non-diagonal elements calculated using the constraint defined in Equation 2.1. Using this underlying topology of a network, a set of constraints and policies, and a routing algorithm (a computable function that determines an optimal collection of edges), a routing system (distributed system) is designed to efficiently generate one or more valid paths (representing a sequence of nodes from a source node to a target node) that satisfy the specified constraints (Gavoille 2001). The source and target nodes in a path p are denoted using S and T, respectively. The weight attached to each edge $e \in E$ between two nodes is specified using the weight set W, where $W_{i,j}$ represents the

Title	Topics	Section / Page
Embedding routing constraints in graph networks	Mathematical Formulation	Sec. 2.2 / Page. 8
The Vehicle Routing Problem (VRP)	Classic and Capacitated VRP	Sec. 2.3 / Page. 10
	Time-related VRP	
	Green VRP	
	Uncertainty-aware VRP	
	Multi-Objective VRP	
Problems in existing vehicle routing systems	Theoretical Illustration	Sec. 2.4 / Page. 13
Routing with multiple suitable paths	K-Shortest Path (KSP)	Sec. 2.5 / Page. 15
	KSP with Limited Overlap (KSPwLO)	
	KSP with Diversity (KSPD)	
	K-Most Diverse Near SP (KMDNSP)	
Evolutionary Algorithms (EAs)	Genetic Algorithms (GAs)	Sec. 2.6 / Page. 18
	Parallel GAs (PGAs)	
	Multi-Objective GAs (MOGAs)	
GAs for path routing	Algorithm type	Sec. 2.7 / Page. 25
	Network size	
	Encoding method	
	Crossover and Mutation method	
	Selection method	
Identified gaps in the literature	Theoretical Illustration	Sec. 2.8 / Page. 36

Table 2.1: Literature review overview.

distance between the nodes (i, j) in the network.

$$P_{i,j} = \left\{ \begin{array}{ll} 1, & E_{i,j} \in p \\ 0, & E_{i,j} \notin p \end{array} \right\} \tag{2.1}$$

Constraints and policies define the desired qualities of a routing system, while its efficiency is evaluated based on how well it satisfies those qualities (Gavoille 2001). It is noteworthy that while certain constraints such as enforcing loop free paths, since loops inevitably lead to sub-optimal paths (Yang & Wang 2008, Van Wart et al. 2014) are commonly employed across most routing systems, others are specifically designed with particular application focus. Some of the widely used constraints in routing systems include,

- The path does not loop back to the source node, represented using Equation 2.2.

$$\sum_{(i,j)\in N}^{i=S} P_{i,j} - \sum_{(i,j)\in N}^{i=S} P_{j,i} = 1 \tag{2.2}$$

- The path terminates at the target node, and the target node must not appear elsewhere along the path, represented using Equation 2.3.

$$\sum_{(i,j)\in N}^{i=T} P_{i,j} - \sum_{(i,j)\in N}^{i=T} P_{j,i} = -1 \tag{2.3}$$

- The path is loop free (no repeating nodes), represented using Equation 2.4.

$$\sum_{(i,j)\in N}^{i\neq(S,T)} P_{i,j} - \sum_{(i,j)\in N}^{i\neq(S,T)} P_{j,i} = 0 \tag{2.4}$$

- The sum of the weights along the edges defines the length (or cost) $|p|$ of a path p, as represented in Equation 2.5. In most cases, this length represents the geographical distance between S and T, and as is common in shortest path problems on real-world networks, the overall length of the path $|p|$ must be minimised.

$$|p| = \sum_{i=S}^{T} \sum_{j=S}^{T} A_{ij} W_{ij}, \;\; i \neq j \tag{2.5}$$

Building on some or all of the aforementioned constraints, several studies have introduced new constraints and policies, leading to a variety of routing problems with different application domains. For instance, constraints such as a minimum available bandwidth and an upper bound on end-to-end delay in Quality of Service (QoS) routing (Wang & Crowcroft 1996, Chen & Nahrstedt 1998, Guck et al. 2017), low energy consumption constraints in energy aware routing (Shah & Rabaey 2002, Amgoth & Jana 2015), location (physical location of nodes) based constraints in geographic routing (Rao et al. 2003, Cadger et al. 2012, Kaur et al. 2016), length or cost based constraints in vehicle routing (Nanayakkara et al. 2007, Chondrogiannis et al. 2015, Häcker et al. 2021), and others.

In this thesis, we concentrate on the problem of vehicle routing on complex real-world road networks, commonly referred to as the Vehicle Routing Problem (VRP). Specifically, we aim to demonstrate the capabilities of GAs, a prominent class of EAs, to generate and optimise multiple paths (Inagaki et al. 1999), while simultaneously addressing multiple constraints and objectives. Although recent machine learning–based approaches, such as Reinforcement Learning (RL), Graph Neural Networks (GNNs), show growing potential in routing research (Bogyrbayeva et al. 2024), they currently rely heavily on large amounts of training data and offer limited control over explicit optimisation constraints. In contrast, GAs allow complete offline optimisation, direct constraint handling, and interpretable objective design, making them a more suitable choice for the problem settings addressed in this work.

2.3 The Vehicle Routing Problem (VRP)

Dantzig et al. (1954) introduced the first VRP, addressing a large-scale Travelling Salesman Problem (TSP) (Eksioglu et al. 2009). Since then, the VRP has emerged as one of the most important, and extensively studied combinatorial optimisation problems (Cordeau et al. 2007). Research on the VRP accelerated during the 1990s, mainly due to the growing capabilities and availability of microcomputers (Eksioglu et al. 2009). This development

enabled researchers to develop new and advance strategies for the VRP, supporting its wide-ranging applications in logistics and transportation. Some of the most widely studied extensions of the VRP, both in academic research and practical applications, include

- **Classic VRP:** Fundamental routing problems that were designed solely to optimise (minimise) path length. Examples include:
  - **Shortest Path (SP):** Finding the single shortest path between a given pair of source and target nodes (Nanayakkara et al. 2007). Dijkstras (EW 1959), Depth-First Search (DFS) (Tarjan 1972), Bellman-Ford (Bellman 1958), and the A* (A-star) (Hart et al. 1968) are some of the most widely used algorithms for SP routing.
  - **K-Shortest Path (KSP):** Finding the top K shortest path between a given pair of source and target nodes, used to provide multiple optimal paths to the end-user. Some of the most popular KSP algorithms were introduced in (Yen 1971, Katoh et al. 1982, Eppstein 1998).
- **Capacitated VRP:** This is an extension of the traditional TSP (Eksioglu et al. 2009), in which the objective is to find a circuit with the lowest possible cost, visiting each customer location exactly once (Toth & Vigo 2002a). Commonly employed constraints include the use of identical vehicles, each starting and returning to the same depot, each customer being visited exactly once, and all paths starting simultaneously, unless otherwise specified (Toth & Vigo 2002a, Mor & Speranza 2022, Zhang et al. 2022). Branch-and-Bound (Toth & Vigo 2002b) and Branch-and-Cut algorithms (Naddef & Rinaldi 2002) are among the most widely accepted methods for solving the Capacitated VRP.
- **Time-related extensions of VRP:** These are extensions of VRP that incorporate time as a crucial factor in the routing process. Some of the widely known examples include,
  - **Time-Dependent VRP:** Where travel times varies on factors like, time of day, traffic congestion, weather conditions, moving targets and others (Gendreau et al. 2015, Adamo et al. 2024). Several heuristic and metaheuristic approaches have been proposed to address the Time-Dependent VRP. Notable examples include the time-dependent model introduced in (Ichoua et al. 2003), and the ant colony optimisation method by Donati et al. (2008).
  - **Time-Window VRP:** Where a time window constraint is associated with each customer, specifying the earliest and latest allowable service times. Notably, each customer must be served within their designated time window, making time a critical aspect of the routing solution (Spliet & Gabor 2015, Zhang et al. 2022). Heuristic algorithms proposed in (Solomon 1987, Cordeau et al. 2001), along with the multi-objective genetic algorithm introduced in (Ombuki et al. 2006), are among the widely accepted algorithms for solving the Time-Window VRP.

- **Time-Dependent VRP with Time-Windows:** A combination of Time-Dependent and Time-Window VRP, considering both dynamic travel times and customer time constraints (Dabia et al. 2013, Gmira et al. 2021). Algorithmic approaches proposed for this complex variant include Branch-and-Price (Dabia et al. 2013), Tabu Search (Gmira et al. 2021), among others.

- **Green VRP:** This is a relatively recent extension of the classical VRP. The concept of Green VRP was introduced by Erdoğan & Miller-Hooks (2012), where the focus is on minimising environmental impact (Zhang et al. 2022, Garside et al. 2024). The two primary focuses of the Green VRP include, optimising fuel consumption by taking into consideration variables like load weight, vehicle speed, and travel distance and second, integrating alternative energy sources like hydrogen, compressed natural gas (CNG), electricity, and biodiesel (Garside et al. 2024). For single-objective Green VRP, metaheuristic algorithms (often in hybrid setting) are commonly employed. In the case of multi-objective Green VRP, Non-dominated Sorting Genetic Algorithm II (NSGA-II) is among the most frequently proposed algorithms (Jemai et al. 2012, Xu et al. 2019, Yin 2022, Heidari et al. 2023).

- **Uncertainty Aware VRP:** These are the extensions of VRP that account for uncertainty or real-time changes in one or more elements such as presence of customers, travel times, service times, and others (Oyola et al. 2018). Two widely studied variants in this category are

  - **Stochastic VRP:** This variation of the VRP uses one or more stochastic parameters, which means that some future events are represented as random variables with a known probability distribution (Ritzinger et al. 2016). Several algorithmic studies have been proposed to solve various variants of the Stochastic VRP, including heuristic approaches (Yang et al. 2000, Mirabi et al. 2010), Particle Swarm Optimisation (Marinakis et al. 2013), GAs (Wang et al. 2017), and others.
  - **Dynamic VRP:** It is also referred to as real-time or online VRP, where some input data is revealed during the execution of the plan (Ritzinger et al. 2016). For instance, arrival of new customer requests, while other elements like demands, service times, and travel times can also vary dynamically (Ritzinger et al. 2016). GAs (Hanshar & Ombuki-Berman 2007, Lau et al. 2009), Particle Swarm Optimisation (Okulewicz & Mańdziuk 2013), hybrid metaheuristic (Euchi et al. 2015), and other approaches have been commonly employed to solve variants of the Dynamic VRP.

- **Points of Interest (POI) VRP:** This variation of the VRP focuses on generating routes that deliberately pass through user specified POIs, while still minimising overall travel distance and/or time. Unlike the classical TSP (Eksioglu et al. 2009), the objective is not to visit all locations, but to optimise the trade-off between travel distance and the number or relevance of POIs visited. In other words, the goal is selec-

tive POI inclusion rather than full coverage, making POI-based VRP more suitable for applications such as Tourist Trip Design Problems (TTDP) (Choachaicharoenkul et al. 2022), where the aim is to minimise travel distance and/or time spent on the road while maximising the exposure to valuable tourist attractions. Although both heuristic and metaheuristic techniques have been applied to POI-based routing problems, the multi-objective structure of TTDPs has led to a stronger reliance on metaheuristic algorithms (Ruiz-Meza & Montoya-Torres 2022), which are better suited for balancing competing optimisation goals.

- **Multi-Objective (or Multi-Criteria) VRP:** This extension of the VRP refers to a class of problems in which the routing system aims to generate paths optimised over multiple objectives, which may or may not be conflicting in nature. While distance, time, and cost remain central considerations, application specific metrics are often needed to address particular needs. For example, in a study on electric vehicle routing for medical waste management (Lin et al. 2024), trips to waste disposal facilities were optimised. In another study on vaccine distribution (Al Theeb et al. 2024), minimising the deviation from vaccine centres and on-site locations was a key objective. Similarly, in forest fire fighting VRP (Li et al. 2024), the focus was on minimising total losses across fire spots to optimise paths. Additionally, extensive research has been conducted on reducing fuel consumption and CO2 emissions, often alongside other objectives in VRP formulations (Gong et al. 2018, Garside et al. 2024). The fast convergence and ability to maintain a diverse set of solutions contribute to the popularity of NSGA-II (Deb et al. 2002) as a preferred algorithm for multi-objective VRP (Garside et al. 2024).

- Other notable variants of the VRP include, the Simultaneous Delivery and Pickup VRP (Zhang et al. 2022), Multi-Trip VRP (Zhang et al. 2022), and Multi-Destination VRP (Hakeem et al. 2019) among others.

One noteworthy takeaway from the aforementioned VRP variations is that, whilst a large number of research projects/works concentrate on addressing certain VRP formulations, only a small number specifically address vehicle routing in relation to the needs of the general public. With rapid urban development and population growth, the need for advance routing systems has become increasingly evident. Recently, studies have begun to explore the relationship between mobility attributes, urban planning, and their effects on mental well-being and travel satisfaction (Conceição et al. 2023). Traffic congestion, commuting delays, uncertainty, and inadequate infrastructure are among the key factors that can negatively impact users mental well-being (Conceição et al. 2023).

2.4 Problems in existing vehicle routing systems

To comprehensively describe the problem addressed in this thesis, we use Figure 2.1 as an illustrative example. Consider a group of international students from Maynooth University

(source node: Maynooth, Ireland) plan to travel to Fanad Lighthouse in Donegal, Ireland (target node). The figure displays three paths created using Google Maps (Google n.d.), with the fastest path highlighted in dark blue, while alternative options appear in light blue.

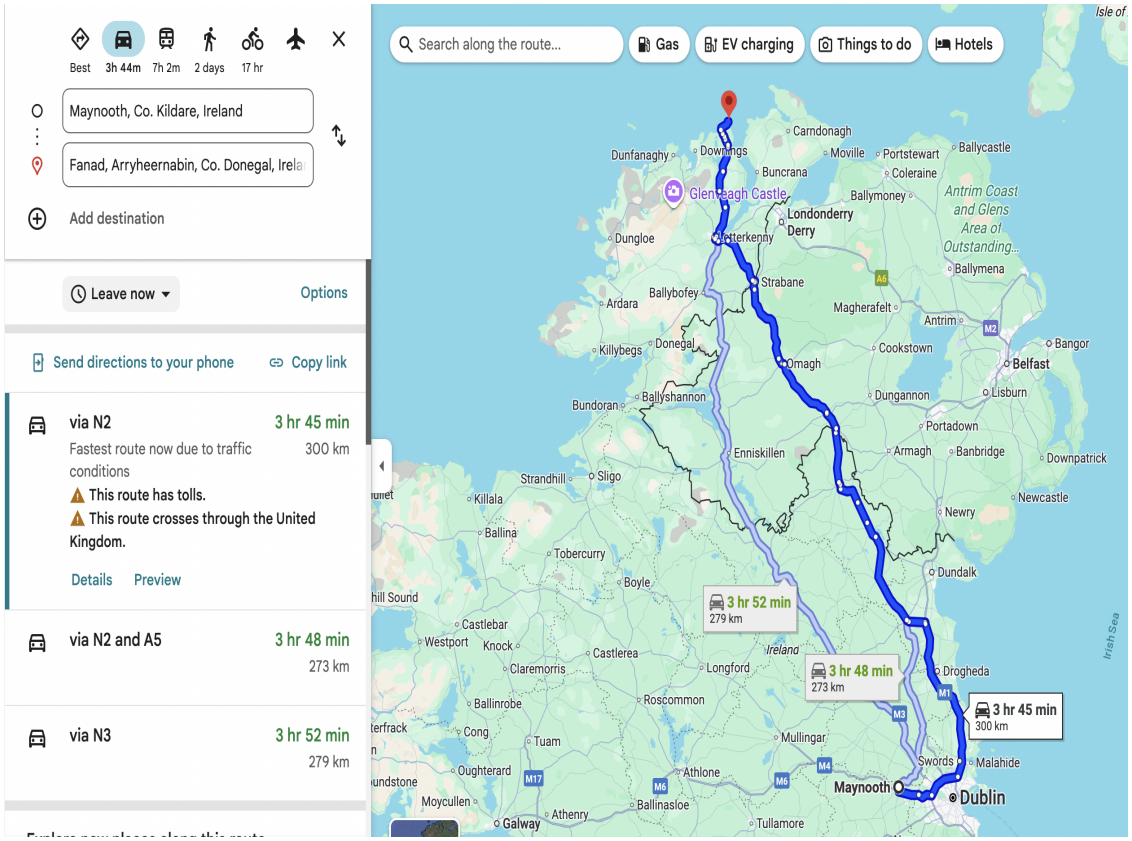

Figure 2.1: An example of paths generated using Google Maps (Google n.d.), with *Maynooth, Co. Kildare, Ireland* as the source node and *Fanad, Co. Donegal, Ireland* as the target node.

Notably, two of the alternative paths are shorter by 27 and 21 kilometres, respectively, yet neither is the fastest option. This demonstrates that the shortest path is not necessarily the quickest, due to the dynamic nature of real-world transportation systems. More importantly, this case also reveals several broader limitations in existing routing systems, including:

- **Lack of structural dissimilarity control:** Among the three paths, two share nearly 75% of the same route, indicating a high degree of overlap. The system offers no mechanism to generate structurally diverse (i.e., routes with minimal shared edges) paths, which is often desirable in applications such as tourism planning, logistics redundancy, or evacuation routing.
- **No support for multi-objective optimisation involving certain POIs:** Although the interface allows users to visually overlay points of interest such as gas stations, EV charging stations, and hotels, these attributes are not integrated into

the underlying optimisation model. As a result, users cannot request paths that explicitly optimise for such criteria.

A proposed approach to mitigate these challenges involves providing multiple optimal paths to the end-user, a concept known as **alternate routing** or **multipath routing**. By incorporating additional constraints and/or objectives to address the challenges associated with shared edges (overlapping segments) among alternate paths, as well as amenities such as gas and EV charging stations directly into the optimisation process, users are enabled to select the most suitable path based on their needs.

With this research, we are proposing the use of Genetic Algorithms (GAs) to solve the VRP, with a specific emphasis on promoting dissimilarity among paths and integrating amenity specific POIs such as gas and EV charging stations into applications designed for the general public. GAs operate by evolving a population of individuals, efficiently exploring the search space to identify ideal or near-ideal solutions to complex problems. By generating and optimising multiple paths across multiple objectives simultaneously, this evolutionary process naturally produces a diverse set of solutions, making GAs a desired candidate for the VRP. It is important to note that by framing the trade-off between travel distance and the number of POIs as a fitness function (described in Chapter 5.2.1), where each POI is assigned a weight based on its position in the route, the work presented in this thesis directly addresses a key limitation of POI-based VRPs, namely that the total travel distance tends to increase as more POIs are included in the route. In the following section, we delve deeper into the concept of alternate routing, discuss key algorithms, and describe the concept of structural dissimilarity as a means to reduce such overlap and ensure route diversity. Finally, we explore widely used dissimilarity metrics and discuss how these metrics are incorporated into modern routing systems to enhance quality.

2.5 Routing where multiple suitable paths exists

K-Shortest Path (KSP) algorithms (Yen 1971, Katoh et al. 1982, Eppstein 1998) are well known alternate routing algorithms that are extensively used in various applications, including route guidance systems (Yu et al. 2020), Risk-based path planning (Maidana et al. 2023), Constrained based routing (Zhang et al. 2021), dynamic network routing (Yu et al. 2020), and others. However, despite their benefits and widespread application, KSP algorithms (Yen 1971, Katoh et al. 1982, Eppstein 1998) also have limitations. First, they typically optimise for path length as the sole objective, making it difficult to incorporate other factors such as travel time, cost, or route features. Furthermore, the top-k shortest paths often exhibit significant overlap (Liu, Jin, Yang & Zhou 2017), sharing many common edges, which reduces route diversity and limits their practical usefulness. To illustrate this, we provide an example in Figure 2.2, displaying five distinct paths between a randomly selected source and target node on a 20 node network. These paths were obtained using the method by Yen (1971), implemented in NetworkX (Hagberg et al. 2008). Although all five paths are shortest paths of equal length, they exhibit significant overlap in the edges they share.

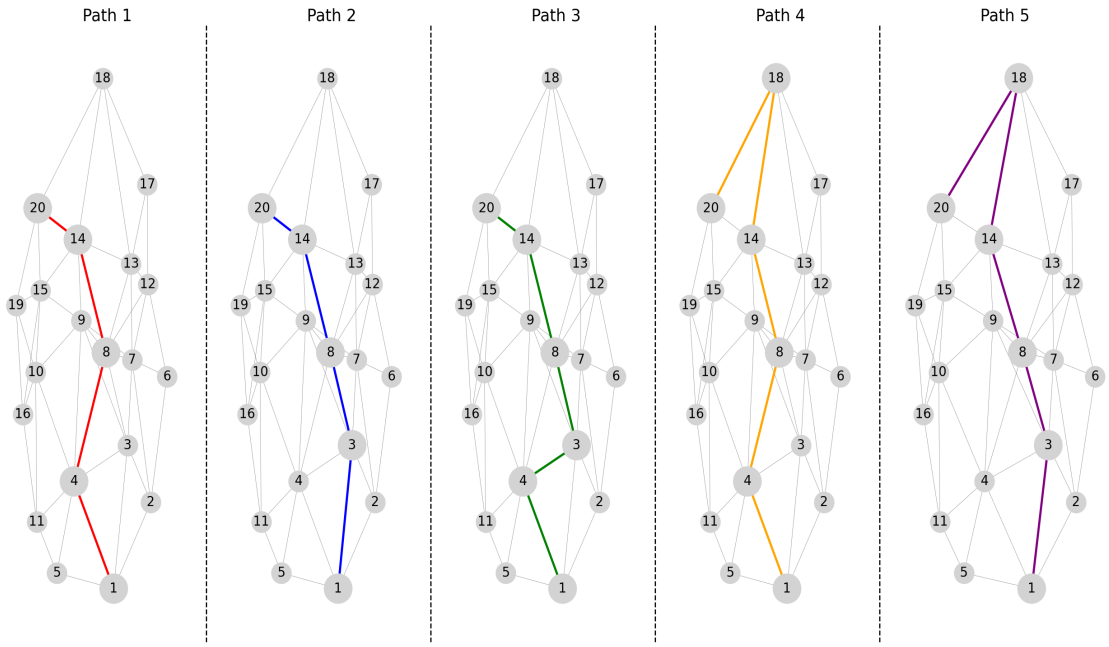

Figure 2.2: An example highlighting the limitation of shared edges among the routes generated using KSP algorithms.

To address the issue of shared edges (similar segments) in paths generated by KSP algorithms, the research community has actively explored heuristic and metaheuristic algorithms, to serve this purpose in transportation systems. Similar to the KSP problem (Eppstein 1998), the aim is to find multiple optimised routes between a source and a target node. However, unlike traditional KSP, the quality of a path is evaluated not only based on its length but also on its structural dissimilarity from other shortest paths. Using this additional constraint of structural dissimilarity, Chondrogiannis et al. (2015) introduced the K-Shortest Paths with Limited Overlap (KSPwLO) problem, which aims to recommend K alternative paths that are (a) as short as possible and (b) sufficiently dissimilar according to a user-controlled similarity threshold (Cheng et al. 2019). Similarly, Liu, Jin, Yang & Zhou (2017) proposed the K-Shortest Paths with Diversity (KSPD) framework, which evaluates path quality based on both length and structural dissimilarity from other shortest paths.

Both KSPwLO (Chondrogiannis et al. 2015) and KSPD (Liu, Jin, Yang & Zhou 2017) were proposed to ensure that there are reliable backup options available, especially in dynamic environments where path conditions can change rapidly, allowing for a more resilient and adaptable routing strategy. To achieve this, both approaches used a user defined similarity threshold for the selection of optimised dissimilar paths. For instance, a user defining a required percentage of dissimilarity between paths in a given set. However, requiring the user to specify a similarity threshold made the frameworks difficult to use, as determining an appropriate similarity level to ensure statistically meaningful diversity is often challenging (Häcker et al. 2021).

Addressing this need of simplicity, a new variant called the K-Most Diverse Near-Shortest Paths (KMDNSP) (Häcker et al. 2021) problem was introduced to shift the focus from similarity based threshold to a more intuitive approach requiring users to define

a length threshold (ϵ). For instance, a set of K diverse (less overlapping) paths, each no more than 10% longer than the shortest path. This is easier for users to understand compared to setting a similarity threshold, as specifying a maximum permissible deviation from the shortest path is more straightforward and directly corresponds to the length of the paths. This enhancement improves usability and provides a controlled trade-off between efficiency and diversity, making the output easier for users to understand. These studies (Chondrogiannis et al. 2015, Liu, Jin, Yang & Zhou 2017, Häcker et al. 2021) have significantly advanced alternate routing by providing multiple, distinct routing options to mitigate risks such as traffic congestion, road closures, and system failures. Such extensions have broad applications across various domains, including smart city planning, exploration, and rescue operations, among others.

Notation	**Definition**	**Example:** $Dis(p_1, p_2)$
$Dis_1(p, p')$	$1 - \frac{L(p \cap p')}{L(p \cup p')}$	$1 - \frac{10}{81} = 0.88$
$Dis_2(p, p')$	$1 - (\frac{L(p \cap p')}{2L(p)} + \frac{L(p \cap p')}{2L(p')})$	$1 - (\frac{10}{90} + \frac{10}{92}) = 0.78$
$Dis_3(p, p')$	$1 - \sqrt{\frac{L(p \cap p')^2}{L(p)L(p')}}$	$1 - (\frac{10*10}{45*46})^{1/2} = 0.78$
$Dis_4(p, p')$	$1 - \frac{L(p \cap p')}{\max(L(p), L(p'))}$	$1 - \frac{10}{46} = 0.78$
$Dis_5(p, p')$	$1 - \frac{L(p \cap p')}{\min(L(p), L(p'))}$	$1 - \frac{10}{45} = 0.78$

Table 2.2: Several commonly used similarity metrics were highlighted by Liu, Jin, Yang & Zhou (2017). Subsequently, Häcker et al. (2021) proposed dissimilarity metric $Dis_1(p, p')$ by subtracting the similarity score from 1. In this table, we adopt the same approach, deriving dissimilarity functions by subtracting the similarity score from 1. For illustration purposes, we calculate the value of each function for two sample paths p_1: $S \xrightarrow{10} n_1 \xrightarrow{25} n_4 \xrightarrow{10} T$, and p_2: $S \xrightarrow{10} n_1 \xrightarrow{6} n_3 \xrightarrow{30} T$.

Structural dissimilarity between two paths p and p' ($\text{Dis}(p, p')$) is commonly quantified using one or more of the dissimilarity functions presented in Table 2.2. The terms $L(p)$ and $L(p')$ denote the lengths of the paths p and p', respectively. The term $L(p \cap p')$ represents the length of the intersection between the two paths, and $L(p \cup p')$ denotes the length of their union. These functions return values ranging from 0 to 1, with higher value of $Dis(p, p')$ indicating fewer shared edges between the paths, which is a desired characteristic in alternate routing. Practical example for each function is also provided, based on two sample paths p_1: $S \xrightarrow{10} n_1 \xrightarrow{25} n_4 \xrightarrow{10} T$, and p_2: $S \xrightarrow{10} n_1 \xrightarrow{6} n_3 \xrightarrow{30} T$, as illustrated in Figure 4.1.

The incorporation of path dissimilarity constraints significantly increases the complexity of problems like KMDNSP. In particular, the requirement to evaluate combinatorial path sets based on structural dissimilarity render these problems NP-hard, with computational complexity growing exponentially as the number of paths (K) increases (Häcker

et al. 2021). Common approaches employed for optimising NP-hard problems, including complex routing problems, are Evolutionary Algorithms (EAs), also known as Evolutionary Computation systems (Eiben & Smith 2015). In contrast to the aforementioned algorithms, EAs can generate multiple paths optimised for multiple objectives while ensuring a certain level of dissimilarity among the paths. This makes EAs a prominent and efficient solution for ATS. When employed to optimise a sole objective, such as minimising distance (Ahn & Ramakrishna 2002), EAs enable the exploration of multiple diverse (less overlapping) shortest paths, rather than restricting the search to only the shortest path. While with Multi-Objective Evolutionary Algorithms (MOEAs), EAs bring flexibility and adaptability, allowing seamless integration with specially designed evaluation metrics.

In the following section, we briefly introduce the concept of EAs, along with an overview of key terms and specialised terminology used in this domain. Next, we provide an overview of the fundamental theories and concepts in Genetic Algorithms (GAs), including a detailed discussion of Parallel Genetic Algorithms (PGAs) and their variants. With a description of well known GAs called Non-Dominated Sorting Genetic Algorithm (NSGA) II (Deb et al. 2002) and III (Deb & Jain 2013, Jain & Deb 2013), we outline the key ideas in Multi-Objective Genetic Algorithms (MOGAs) (Srinivas & Deb 1994).

2.6 Evolutionary Algorithms (EAs)

EAs are a class of optimisation techniques inspired by Darwins theory of natural selection. By simulating the course of biological evolution over many generations, EAs seek to identify ideal or nearly ideal solutions to complex problems. The essential components, procedures, and results in EAs are generally described using a specialised set of keywords and terminologies, given in Table 2.3. It is important, before we proceed into the more complex parts of this research, to outline the nomenclature used in EAs. Addressing the importance of explaining terminology Luke (2013) provided a similar table to explain the implementation of an EA solution to a given problem.

Terms	**Meaning**
Individual	A candidate solution.
Population	Set of candidate solutions.
Initial population	Set of candidate solutions making the first generation, usually created using a random process.
Fitness value	A quantitative value depending upon the quality of a solution.
Evaluation metric	A quantitative metric designed to evaluate solutions.
Selection	A criterion designed to select individuals based on their fitness value.
Offspring generation process	A process to create new candidate solutions.
Genetic operators	Special operators used to create new candidate solutions in the offspring generation process.
Crossover	Exploitation-based genetic operator.
Mutation	Exploration-based genetic operator.
Genotype or Genome	Genetic representation of a candidate solution.
Phenotype	How the individual operates during fitness evaluation.
Encoding method	A method used to create the genetic representation of a candidate solution.
Generation	One cycle of fitness evaluation – offspring generation – selection.

Table 2.3: Key terms and definitions in Evolutionary Algorithms (EAs).

EAs have different types and most EAs can be divided into Generational Algorithms, which update the entire sample space once per iteration (also referred as generation), and Steady-state Algorithms, which update a few candidate solutions at a time (Luke 2013). Some of the most commonly employed EAs include, GAs, Evolutionary Programming (EP), Evolutionary Strategies (ES), and others. In general, EAs work with a population of potential solutions to a particular problem (Galván, Simpson & Ameneyro 2022) and iterate a finite number of times using different methods. First, the initial population is evaluated using a evaluation metric designed for the problem in hand. This evaluation metric is also known as *fitness function*. Then, by selecting the highest performing individuals (or fittest individuals as measured by the fitness function) from the initial population, a new population of individuals known as offspring population is created. This process of creating the offspring population is commonly known as offspring generation process. Lastly, the parent and offspring population are combined and a selection process is performed to select the next generation for the evaluation metric to evaluate, and the cycle continues for a predefined number of iterations. The core idea is that the EA will evolve a set of candidate solutions, over time, which have survived over several generations and are considered close to or nearly optimal solutions.

2.6.1 Genetic Algorithms (GAs)

John Holland is recognised as the pioneer behind the development of GAs (Goldberg 1989). GAs belong to the family of biologically inspired optimisation algorithms and are often regarded as one of the most popular class of EAs. Primarily employed for optimisation purposes GAs followed a general framework as shown in Figure 2.3 and consist of four main stages,

1. Initialisation, which involves creating an initial population of solutions.
2. Fitness Evaluation, which involves evaluating the population on an objective function.
3. Selection, where high-performing solutions are filtered to be parents for the next stage.
4. Generate Child Population, which refers to creating a new population of solutions using genetic operators such as crossover and mutation on the filtered population.
5. Replacement, which involves merging the parent and offspring populations to retain the fittest individuals for the next generation.

These stages are repeated for a finite number of generations (iterations), determined by one or more termination conditions, with the newly generated population serving as the initial population in the next generation. The evolutionary computation community have proposed several different variants of GAs over the decades. Some of the most famous variants of GAs include:

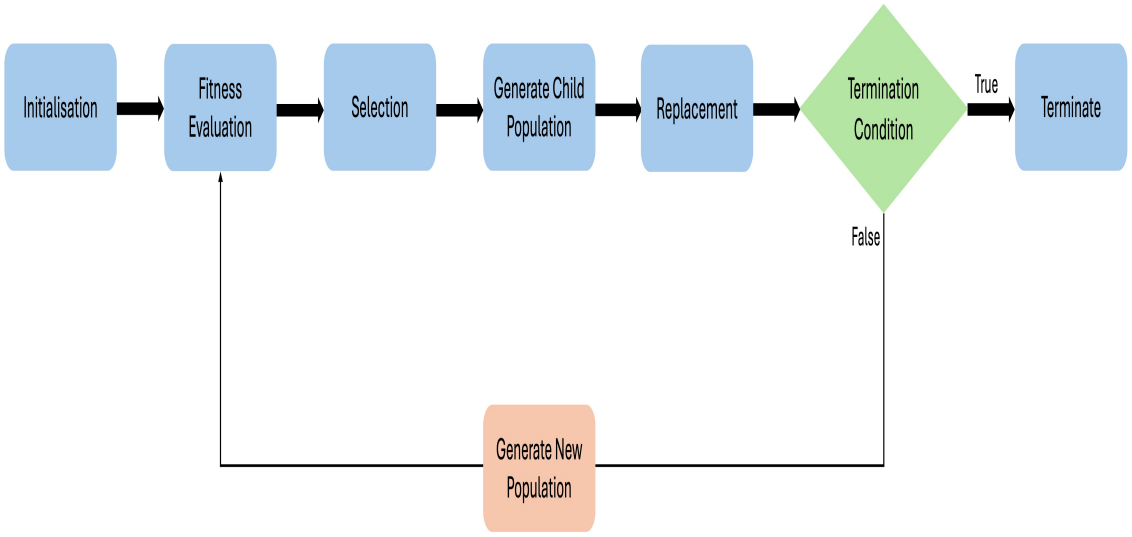

Figure 2.3: Generic framework of a genetic algorithm.

- **Multi-Objective Genetic Algorithms (MOGAs):** As the name suggest, these algorithms are designed to optimise multiple objectives simultaneously.
- **Parallel Genetic Algorithms (PGAs):** PGAs are known for their parallel processing capabilities where they perform genetic operations on multiple individuals simultaneously.
- **Genetic Programming (GP):** This is an extension of GAs where population consist of computer programs usually represented using a tree structure. This method is acknowledged but not utilised in this thesis.

2.6.2 Multi-Objective Genetic Algorithms (MOGAs)

GAs with two or more objectives to optimise simultaneously are classified as MOGAs. Often these objectives will be in conflict, so the focus then is to search for a set of trade-off solutions as a global optimum becomes unattainable (Srinivas & Deb 1994, Deb et al. 2002, Galván & Schoenauer 2019, Galván, Trujillo & Stapleton 2022, Galván & Stapleton 2023). One widely used concept to achieve this is Pareto Dominance. This relies on the principle of having a group of solutions that make up the pareto front. On this front no solutions fitness value for any of the objectives can be improved without degrading the fitness value for at least one of the other objectives. For a maximisation problem, solution (a) dominates solution (b), if the condition specified in Equation 2.6 is satisfied.

$$\begin{aligned} &\forall i \in \{1, 2, \ldots, n\} : \quad f_i(a) \geq f_i(b) \quad \wedge \\ &\exists j \in \{1, 2, \ldots, n\} : \quad f_j(a) > f_j(b) \end{aligned} \tag{2.6}$$

where n represents the total number of objectives, $f(a)$ and $f(b)$ represents the fitness value of solutions (a) and (b) for the ith and jth objectives, respectively (Zitzler & Thiele 1999). Using the concept of Pareto Dominance, (Srinivas & Deb 1994) proposed an selection

algorithm known as the Non-Dominated Sorting Genetic Algorithm (NSGA) for multi-objective optimisation. However, NSGA itself suffered from high computational and time complexity, a lack of diversity, and a lack of elitism (Verma et al. 2021, Horn et al. 1994). To address the aforementioned issues, (Deb et al. 2002) proposed a fast elitist non-dominated sorting genetic algorithm (NSGA-II), designed with the goal to preserve elitism throughout the algorithm using two main principles, Non-Dominated Sorting and crowding distance described as follows:

- **Non-dominated Sorting**: The initial population is sorted using the concept of Pareto Dominance, with non-dominated members receiving the first rank. These first-ranked members are then removed from the initial population and assigned to the first front. The remaining population will be subjected to the non-dominant sorting procedure in the same way. Non-dominated members of the remaining population receive the second rank and are placed in the second front. This process is repeated until all population members are assigned to different fronts based on their rank, as shown in Figure 2.4 (a–c).

- **Crowding Distance:** This metric was designed to assign better ranking to solutions belonging to less crowded regions. This was proposed to sort the population belonging to the same front. If f_j^i represents the fitness value of an ith solution for the jth objective function then the crowding distance for an ith solution is calculated using Equation 2.7. In this Equation, f_j^{max} and f_j^{min} are the maximum and minimum values for the jth objective function among all the individuals, $(i+1)$ and $(i-1)$ are the neighbouring solutions to the ith solution, as shown in Figure 2.4 (a). N is the number of objective functions. When two solutions with different crowding distances are compared the solution with the larger crowding distance is considered to be present in a less crowded region (Verma et al. 2021).

$$cd(i) = \sum_{N}^{i=1} \frac{f_j^{i+1} - f_j^{i-1}}{f_j^{max} - f_j^{min}} \tag{2.7}$$

An enhanced version of the Crowding Distance method, called Improved Crowding Distance (ICD), was proposed by (Yue et al. 2021). Unlike the original Crowding Distance approach, ICD calculates the crowding distance of a solution based on its neighbours from the entire search space, rather than just the neighbours from the same front. Figure 2.4 (b) illustrates the improved crowding distance method, where the crowding distance for the ith solution is determined by considering the fitness values of its true neighbours h and k relative to the fitness values of the $(i+1)^{th}$ and $(i-1)^{th}$ solutions for the jth objective, as shown in Figure 2.4 (a). The study conducted by (Yue et al. 2021) reported improved diversity in both decision and objective space as the result of this modification.

Similar to the crowding distance, reference-point-based nondominated sorting approach, NSGA-III (Deb & Jain 2013, Jain & Deb 2013) was proposed to sort the population belonging to the same front. It prioritises filling the under-represented reference directions

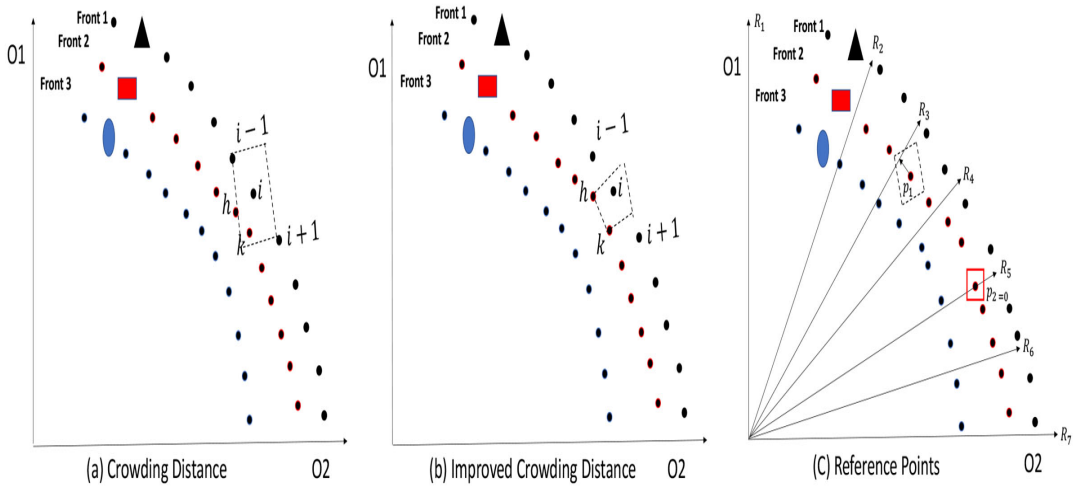

Figure 2.4: Comparison between Crowding Distance (Deb et al. 2002), Improved Crowding Distance (Yue et al. 2021), Reference Points (Deb & Jain 2013, Jain & Deb 2013).

(using a preset set of reference points) first, and if there is not a solution allocated for a reference direction, the solution that has the least perpendicular distance in the normalised objective space is chosen to survive (Blank & Deb 2020). Users can supply these reference points by assigning a certain number of divisions along each objective. The total number of reference points H in an M-objective problem with p divisions along each objective, can be calculated as

$$H = \binom{M + p - 1}{p} \tag{2.8}$$

H is not an algorithmic parameter in NSGA-III, as the population size N is approximately equal to H. Number of divisions (p) along each objective determines the distribution of reference points in the objective space. A large number of divisions results in more reference points and a more even distribution in the objective space, though at the cost of increased computational complexity (Deb & Jain 2013). Conversely, a small number of divisions leads to fewer reference points and a less even distribution. An example is shown in Figure 2.4 (c), where a solution with a perpendicular distance of zero, from the reference direction of R_5 is selected to survive from the second front.

While NSGA-III effectively handles multiple objectives by maintaining diversity, the increasing number of conflicting objectives and decision variables pose additional challenges for researchers. Optimisation problems with decision variables exceeding 100 are defined as large-scale multi-objective optimisation problems (LSMOPs) (Zhang et al. 2023). Initially, LSMOPs were handled by randomly grouping the decision variables (Antonio & Coello 2013, Tian et al. 2024). However, more sophisticated approaches, such as Pearson correlation-based grouping (Zhang et al. 2023), and multi-granularity clustering (Tian et al. 2024), have since been developed to enable better organisation of variables. However, these methods are beyond the scope of this thesis.

2.6.3 Parallel Genetic Algorithms (PGAs)

Multiple-deme (Distributed Evolutionary Model with Explicit Communication) are well known with different names, such as distributed GAs, parallel GAs, and others (Cantú-Paz et al. 1998). The study proposed by Anderson & Ferris (1990) to solve assembly line balancing problem with distributed structures in their GA was one of the first studies that showcased the potential of PGAs, which are based on the idea of parallel processing. PGAs are implemented by dividing the search process into multiple subprocesses with their own set of initial population and genetic operators. There are four main types of PGAs (Harada & Alba 2020), described as follow:

- **Global Parallelisation (Master–Slave) model:** New solutions are generated using a centralised population (entire population), but evaluated in parallel. After the solutions are evaluated, they are added back to the centralised population.
- **Island model:** Several sets of decentralised population called Islands are evolved separately, with migration (exchange of information about solutions) at certain moments of evolution (Gong et al. 2015, Del Ser et al. 2019, Harada & Alba 2020).
- **Cellular model:** The population is arranged on a grid like structure and different genetic operators are employed on certain neighbourhoods of the grid.
- **Hybrid model:** Combination of any two or more models discussed above.

These aforementioned models, depicted in Figure 2.5 and explained in detail next, are well known for their ability to explore multiple regions in the search space synchronously while reducing search time and improving solution quality due to their parallel processing capabilities (Harada & Alba 2020). The implementation of a PGA with the Global Parallelisation model (Figure 2.5) is a straightforward process that employs a master-slave parallel architecture (Harada & Alba 2020, Del Ser et al. 2019, Gong et al. 2015). In this approach, a centralised population serves as the master computing node and guides the evolutionary process throughout the algorithm by delegating specific tasks, such as generating new solutions to sub-populations that are carried by slave computing nodes. After evaluation, new solutions are added to the centralised population and selection is performed. This can be done in two ways: generational and steady-state. The generational approach refers to the process of adding new solutions to the centralised population at the end of each generation and then using the selection operator. However, in steady-state approaches the centralised population gets updated with new solutions from each slave computing nodes continuously, usually yielding a faster convergence rate with respect to generational models (Harada & Alba 2020).

Island model (Island-Based Genetic Algorithms), shown in Figure 2.5 (b), are powerful optimisation techniques that find applications in engineering design, financial forecasting, biological modelling, and other fields. In essence, multiple-deme GAs can be linked to an island model where the main population is divided into subsets or groups to optimise a specific objective function. Each subpopulation within a group evolves independently,

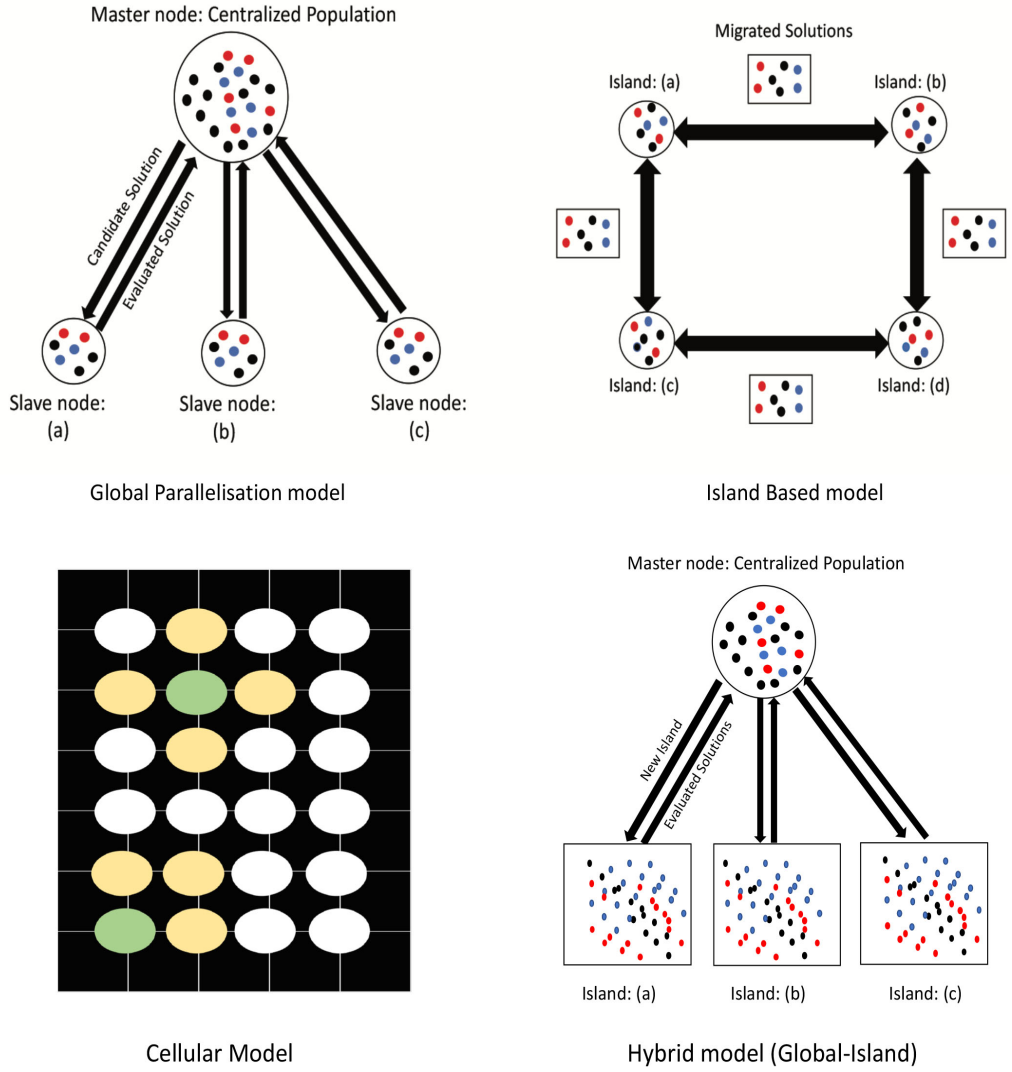

Figure 2.5: Types of PGAs.

with its own unique allele set and evolutionary process. A migration strategy, which includes information like the type of solutions to exchange, migration frequency, replacement conditions, and others, is employed to guide the search space. Through this process, each island explores multiple zones of the search space concurrently, discovering solutions with various characteristics, leading to increased efficiency of solutions (Harada & Alba 2020).

A Cellular model, shown in Figure 2.5 (c), is known for its fine-grained and spatially structured nature, featuring a single population arranged on a grid, ideally with one individual per processor (cell) (Del Ser et al. 2019, Gong et al. 2015). The exchange of information among individuals depends on the position of individuals on the grid, enabling each individual to interact solely with its neighbours. Overlapping neighbourhoods of individuals allows better individuals to disseminate to the entire population (Gong et al. 2015, Del Ser et al. 2019). Figure 2.5 (c) shows an example for a cellular model where two distinct individuals (in green) are shown with their respective neighbourhood, indicating the individuals (in yellow) with whom they can interact.

PGAs incorporating a Hybrid model, shown in Figure 2.5 (d), are also referred to as

Hierarchical models (Gong et al. 2015). These models are produced by combining two (or more) distributed models hierarchically (Gong et al. 2015), allowing the use of the benefits provided by all the incorporated models. In the past, several hybrid models have been proposed, such as Global-Island (Lim et al. 2007), Island-Cellular (Folino & Spezzano 2006), and Island-Island (Hidalgo & Fernández 2005).

In this thesis, we focus on two PGA variants, an Island-based model and a hybrid Global–Island model. Cellular models were not adopted, as their design emphasises maintaining diversity by using overlapping neighbourhoods and promoting slow diffusion of solutions across the population grid (Lim 2014). While this property is advantageous in problems where exploration dominates, routing problems such as KMDNSP and POI-based VRP require a stronger balance between exploration and fast convergence, since producing optimal paths within a reasonable time frame depends heavily on efficient exploitation of promising solutions. For this reason, PGA models based on Global Parallelisation and the Island model were adopted to enable higher exploitation pressure than Cellular models, allowing faster convergence without sacrificing diversity entirely.

2.7 GAs for path routing

Optimal route search in graphs or networks is a classical optimisation problem with various real-world applications. While traditional routing algorithms, including Dijkstras algorithm (EW 1959), depth-first search(Tarjan 1972), and Bellman-Ford algorithm (Bellman 1958), have been widely used for solving this problem, the use of GAs has gained attention as an alternative approach. Several studies (Ahn & Ramakrishna 2002, Chitra & Subbaraj 2012, Dong-liang et al. 2023, Aibinu et al. 2016) promoting the use of GAs in routing systems, including Robot routing (Soueres & Laumond 1996, Bui et al. 1994), internet routing (Ahn & Ramakrishna 2002, Gonen & Louis 2006) and vehicle routing (Ghoseiri & Ghannadpour 2010, Li et al. 2024) have been proposed over the years. But for a routing problem, designing an effective GA is a challenging task due to the following constraints,

1. A Chromosome must be encoded in such a way that it represents a validate path from source node (S) to the target node (T).
2. Each chromosome must not contain any loops (repeating nodes).
3. All offspring chromosomes produced using crossover and mutation must represent a valid path.

Figure 2.6 displays three distinct paths each marked with a different colour. The *Valid Path* is depicted in yellow, *Invalid Path* in red and a *Valid Path (with loop)* is highlighted in blue. It is important to note that the *Valid Path* corresponds to an existing path within the graph whereas the *Invalid Path* does not have any existence within the graph. The *Valid Path (with loop)* is a legitimate path that consists of repeating nodes, forming a loop structure.

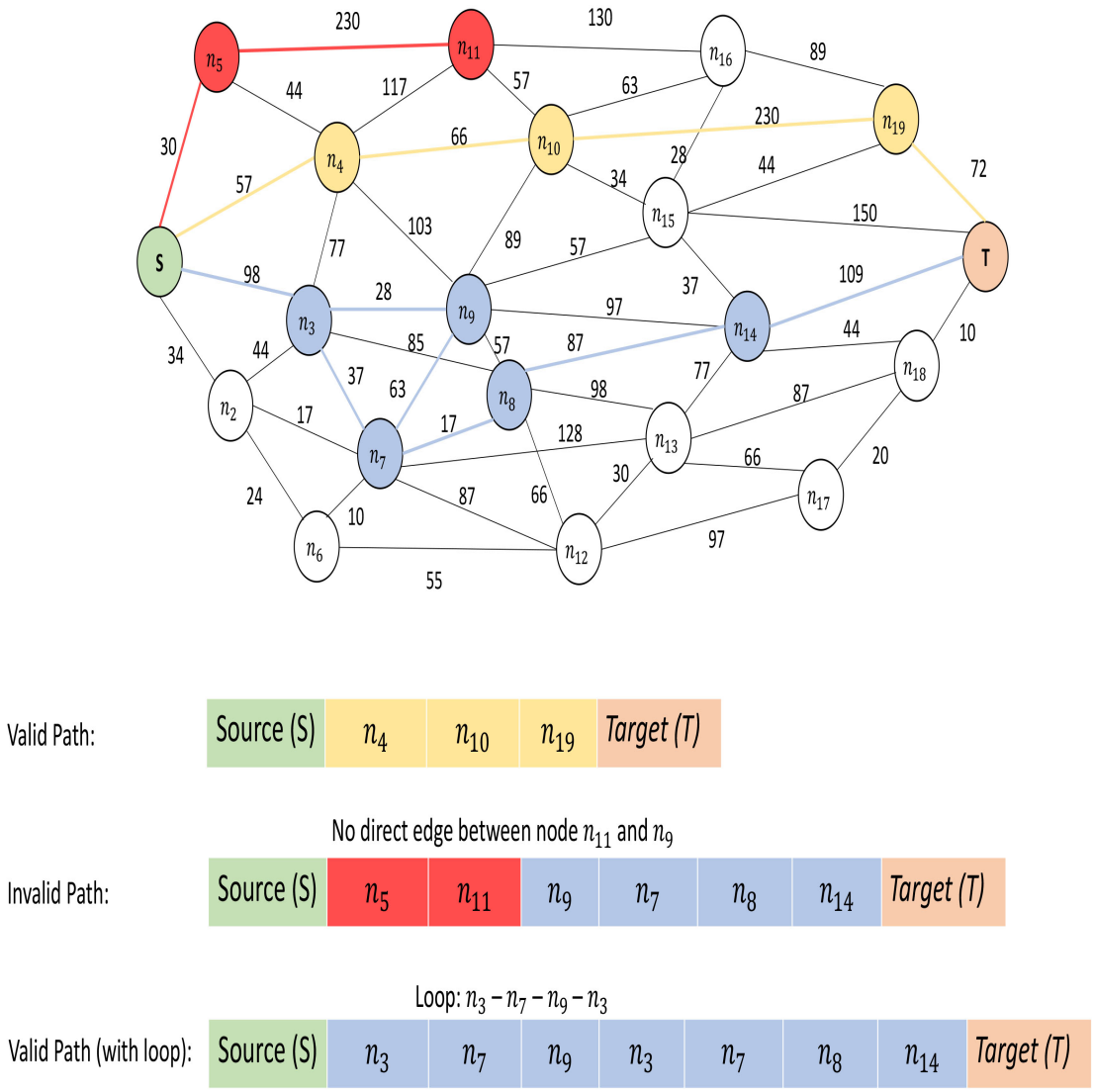

Figure 2.6: Sample graph illustrating a valid path, an invalid path, and a valid path with a loop. Each node is labelled with a unique ID, and different colours are used to distinguish between the paths.

Crossover is a fundamental genetic operator used in GAs to generate new solutions by combining selective portions of two parent chromosomes or solutions from the current generation. Various crossover operators, such as single-point (Kora & Yadlapalli 2017), uniform (Kora & Yadlapalli 2017), and N-point (Kora & Yadlapalli 2017) crossover, are often utilised in GAs. However, these conventional crossover operators are not always suitable for path routing problems, as they may fail to generate valid paths, as shown in Figure 2.7. The crossover operation depicted in Figure 2.7 involves a process of selecting a random node in two parent paths and performing a swap of the segments following the selected nodes. Through this process, new offspring paths with distinctive genetic information inherited from both parents are created. However, as stated these conventional crossover operations may also generate invalid paths depicted in Figure 2.7 and hence, require several modifications and constraints to be compatible for the path routing problem.

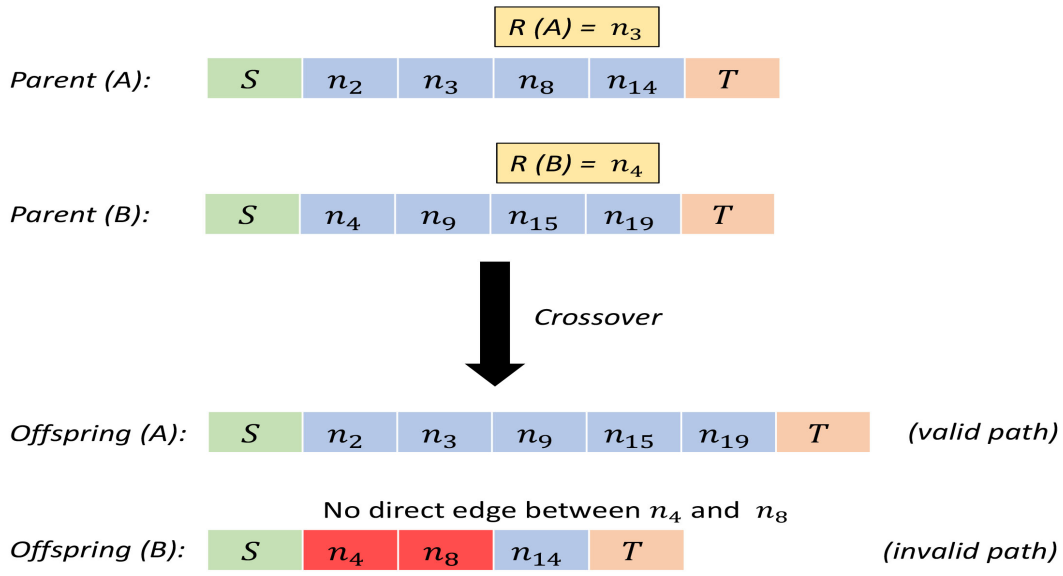

Figure 2.7: Crossover operation generating one valid and one invalid path.

The mutation operator in GAs is commonly utilised to introduce diversity in the population by randomly altering the genes of a chromosome (solution). This can contribute to preventing premature convergence of the GA. However, when dealing with the routing problem, general mutation operators such as bit mutation (Ye et al. 2019) and swap mutation (Deep & Mebrahtu 2011) may not always generate valid paths thereby making them unsuitable. Figure 2.8 show two examples of *Bit Mutation* on a valid path from graph shown in Figure 2.6. Following the mutation procedure, when node n_3 is changed to node n_7, the resulting path remains valid since it exists in the graph. However, when node n_3 is changed to node n_6, the resulting path becomes invalid as it does not exist within the graph. This highlights the necessity for careful modifications during the mutation process, similar to what was observed in the crossover operation.

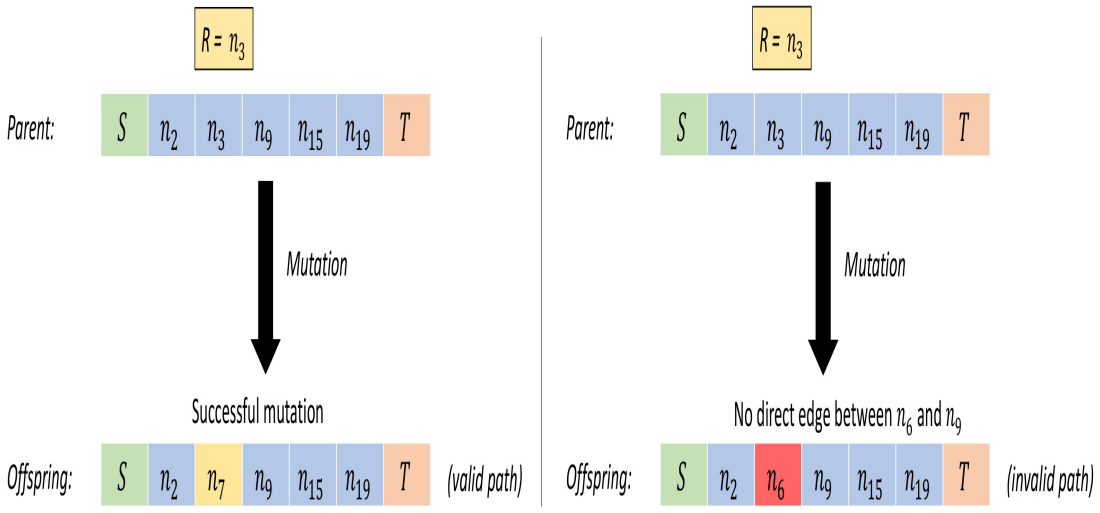

Figure 2.8: Mutation operation generating one valid and one invalid path.

As stated above in Chapter 1, GAs have received significant attention in the context of path routing over the past few decades, establishing it as a well-explored area of study. Next, we report on our structured review of this literature. Several modified encoding schemes and genetic operators have been proposed for the routing problem satisfying all of the specified constraints. We provide a tabular summary of our review of the state of the art literature in Table 2.4 on page 29. The following sections then carefully explain the meaning and context related to each of the columns in table 2.4. Entries marked with − in Table 2.4, indicates that no information was provided by the researchers for those specific areas. The columns in Table 2.4 are discussed as follows:

- **Algorithm type**: Section 2.7.1 describes the type of GA that researchers have proposed in their studies.
- **Network size**: Section 2.7.2 describes the size of networks used by researchers. The number of nodes is stated in all cases.
- **Encoding method**: Section 2.7.3 describes some of the most commonly used encoding schemes to create genetic representations for a candidate path.
- **Genetic operators**: Section 2.7.4 presents some of the modified crossover operators proposed by researchers while section 2.7.5 then outlines the corresponding modified mutation operators.
- **Selection method**: Section 2.7.6 discusses some of the commonly used selection operator in GAs that have been employed for path routing.

2.7.1 Algorithm type

Five distinct types of algorithms are represented in Table 2.4, namely, SOO (Single Objective Optimisation) with path length as the objective to minimise between a given pair of source and target node (Gen et al. 1997, Ahn & Ramakrishna 2002, Qiuqi et al. 2004), MOO (Multiple Objective Optimisation) with two or more application specific objectives (Chitra & Subbaraj 2012, Sharma et al. 2025b), MDR (Multiple Destination Routing) with path length as the objective to minimise between one source and multiple target nodes (Leung et al. 1998), the KSP (K-Shortest Path) (Hamed 2010, Moza & Kumar 2020) and the KMDNSP (K-Most Diverse Near-Shortest Paths) (Sharma et al. 2025a) approaches, generating a set of K paths to facilitate alternate routing strategies. These algorithms employed various modified encoding schemes and genetic operators that adhered to all the specified constraints. We also highlight that, although some studies listed in Table 2.4 are categorised as SOO, they address more complex problems, such as the fuzzy shortest path routing (Lin et al. 2021) and dynamic shortest path routing (Yang et al. 2009, Zhu et al. 2014).

Table 2.4: Summary of literature review of GAs for path routing problems. Please note that a dash symbol (−) indicates that this information was not available in the corresponding paper.

Authors (References)	Algorithm type	Network size	Encoding method	Chromosome length	Crossover method	Mutation method	Selection method
Gen et al. (1997)	SOO	6 – 70	Priority based	Variable	Position based	Swap mutation	Roulette wheel+ Elitist
Munetomo et al. (1998)	SOO	–	Random	Variable	Node based	Path mutation	Local, Global
Leung et al. (1998)	MDR	20 – 100	Set-valued mapping	Fixed	One point	Bit mutation	Proportional
Inagaki et al. (1999)	SOO	–	Random	Fixed	Locus based	–	Roulette wheel, Elitist
Ahn & Ramakrishna (2002)	SOO	15 – 50	Random	Variable	Node based	Path mutation	Tournament Size =2
Qiuqi et al. (2004)	SOO	100	Random	Variable	Node based	Path mutation	Roulette wheel, Elitist
Gonen & Louis (2006)	SOO	20	Random	Variable	Node based	–	–
Gen & Lin (2006)	SOO	20 – 320	Random-key based	Variable	Arithmetical	Swap mutation	Roulette wheel
Nanayakkara et al. (2007)	SOO	10,000	Random	Variable	Node based	Path mutation	Tournament Size = 2
Yang et al. (2009)	SOO	100	Random	Variable	Node based	Path mutation	Tournament Size = 2
Hamed (2010)	KSP	8 – 20	–	Variable	Cut point	Bit mutation	–
Ji et al. (2011)	MOO	933	Random	Variable	Node based	Path mutation	Tournament (Size = 4), Elitist
Chitra & Subbaraj (2012)	MOO	20	Random, Priority based	Variable	Node based, Partially mapped	–	Tournament, Elitist
Wang et al. (2014)	KSP	11 – 26	Random	Variable	Node based	Path mutation	Linear ranking
Zhu et al. (2014)	SOO	100	Random	Variable	Node based	Path mutation	Tournament Size =2
Qiongbing & Lixin (2016)	SOO	100	Random	Variable	Same adjacency	Path mutation	Roulette wheel
Liu et al. (2016)	SOO	2,000	Priority based	Variable	Node based (I)	Path mutation (I)	Roulette wheel, Elitist
Mohammed et al. (2017)	SOO	–	Random	Variable	Hybrid	Adaptive mutation	Roulette wheel
Moza & Kumar (2020)	KSP	9 – 22	Random	Variable	Node based	Path mutation	–
Lin et al. (2021)	SOO	24	Random	Variable	Substring based	–	Tournament Size =2
Sharma et al. (2025a)	KMDNSP	834 – 2,224	Random	Variable	LFPC	LFPC	AvgIslandFit
Sharma et al. (2025b)	MOO	3,000 – 10,000	Random	Variable	LFPC	LFPC	Ordinal ranking

2.7.2 Network size

The choice of an graphical representation significantly impact the effectiveness of an approach. For instance, Dijkstras algorithm (EW 1959) is one of the most used pathfinding algorithms with implementations in numerous path-finding applications. However, it is a blind search algorithm that dissipates a major portion of its resources while finding the optimal path in the network. As the size and complexity of the network increase, the amount of resource dissipation also increases. Therefore, to ensure the effectiveness and efficiency of a graph algorithm, it must be tested on numerous networks of varying sizes and topologies. Ahn & Ramakrishna (2002) further emphasised the importance of such testing, to demonstrate an algorithms robustness and insensitivity to network structure.

Initial studies (Gen et al. 1997, Leung et al. 1998, Ahn & Ramakrishna 2002, Qiuqi et al. 2004) employing GAs for the routing problems, were conducted on very small networks containing nodes between $6 - 100$. The road network of Iwaki, a city in Japan, was used in (Inagaki et al. 1999) but no information about the number of nodes and edges was provided. Munetomo et al. (1998) also did not provided any information about the size of the network they used. To the best of our knowledge, Nanayakkara et al. (2007) proposed the first study to evaluate the effectiveness of a GA on a real-world road network. The simulations for their study were executed on the road network of Singapore with more than $10,000$ nodes. GAs in (Ji et al. 2011, Liu et al. 2016) were tested on networks of significant sizes containing 933 and $2,000$ nodes, respectively.

2.7.3 Encoding method

In GAs, encoding methods are used to encode the genetic representation of chromosomes based on the problem in hand. In the context of routing problems, a candidate path solution (chromosome) can have either a fixed or variable length, and can be initialised in two ways namely heuristic initialisation and random initialisation (Munetomo et al. 1998). The genes forming a chromosome are usually represented by the node identifiers (numbers or characters) from the graph. Some of the commonly used encoding methods are as follows:

- **Priority based encoding:** It is a heuristic initialisation method that has been used in several studies (Gen et al. 1997, Chitra & Subbaraj 2012). In this method, the source node is added and set as the current node in the chromosome. Then, a random process is used to allocate priorities to all adjacent nodes to the current node. The node with the highest priority is added to the chromosome following the current node and set as the new current node, this process is repeated until the current node is same as the destination node. Gen & Lin (2006) introduced Random-key based encoding, where floating point numbers were randomly assigned as priority keys. Liu et al. (2016) also use a form of priority based encoding, in which priority is assigned to nodes with the minimum weight during the initialisation process. A notable drawback of using this encoding scheme is the difficulty in assigning priorities, as a new set of priorities would be required for each chromosome and this could limit the

diversity in the population. Furthermore, the use of heuristics limits the generality of the conclusions with respect to the problem (Osaba et al. 2014), and consequently reduces diversity within the population. As a result, the majority of the studies (as shown in Table 2.4) employed random encoding methods to facilitate faster processing and diversity among solutions.

- **Random encoding:** As the name suggest, it is a random initialisation method. In GAs on routing, it is a preferred method for initialisation due to its ability to produce a diverse (less overlapping) set of solutions, as used by most studies summarised in Table 2.4. This method also starts by adding the source node to the chromosome and setting it as the current node. Then a new node is randomly selected from the collection of nodes forming the edges with the current node. The selected node gets added to the chromosome and set as the new current node, this process is repeated until the current node is same as the destination node.

To comply with the constraints given in Section 2.7, these encoding methods are usually implemented in two ways,

1. With constraints, prohibiting the fabrication of any loop (repeating node) from the encoding stage, typically by tacking pre-existing nodes. Ahn & Ramakrishna (2002), Qiongbing & Lixin (2016), and Lin et al. (2021), employed with constraints encoding methods.

2. Without constraints, allowing the formation of loops at the encoding stage and then using a mending function to remove all the loops from the chromosome. Studies by Liu et al. (2016), and Nanayakkara et al. (2007) employed without constraints encoding methods.

Only the studies proposed in (Leung et al. 1998, Inagaki et al. 1999) explicitly advocated the use of fixed length chromosomes. Notably, these were also among the earliest works to explore the application of GAs to path routing problems. The limitations of fixed length chromosomes, including their inflexibility in representing high quality candidate solutions, have been noted in studies such as (Ahn & Ramakrishna 2002). Consequently, many subsequent works (as shown in Table 2.4) have favoured variable length encodings for greater adaptability in routing applications.

2.7.4 Crossover method

Crossover is a genetic operator in GAs that produces a new chromosome (solution) by combining the selective portion of two chromosomes (existing solutions) from the current generation. Crossover operators that are often employed in GAs include, Single-point crossover, Uniform crossover, Two-point crossover, order crossover, and others. However, these crossover operators are inappropriate for routing, due to their inability to generate a valid path. As a result, numerous modified crossover operators have been proposed over the years for solving routing problems. In this context, some of the commonly employed crossover operators include:

- **Position Based Crossover:** Gen et al. (1997) proposed the use of Position Based Crossover in path routing. To produce a new chromosome, it randomly selects some genes from one parent and fills the remaining gaps with genes from the other parent using a left-to-right scanning process. Path validity is ensured using a repairing procedure.

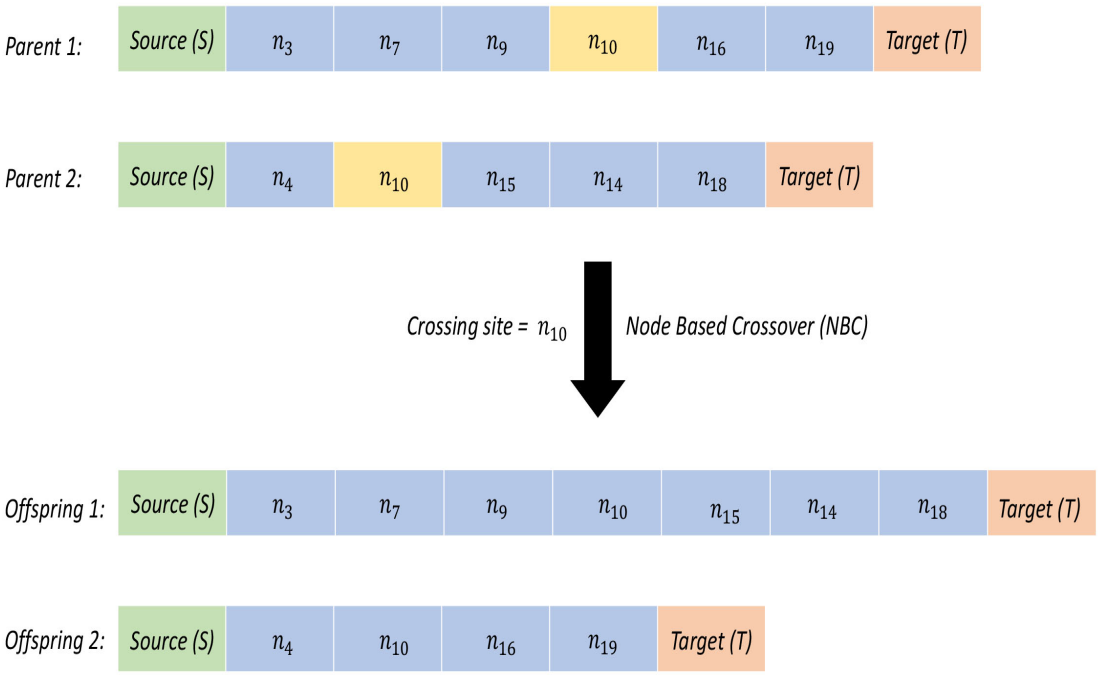

Figure 2.9: Node Based Crossover (NBC).

- **Node Based Crossover (NBC):** This crossover scheme is an adaptation (Chitra & Subbaraj 2012) of the one-point crossover. Due to its simplicity, it is the most frequently employed crossover technique. First, two parent chromosomes are selected and then a crossover point is determined depending upon the existence of at least one common node (excluding the source and destination nodes) between the two parent chromosomes. Crossover is then carried out using the crossover point on both chromosomes, generating two new valid chromosomes. Consider the example shown in Figure 2.9, p_1 and p_2 are, respectively, $S \rightarrow n_3 \rightarrow n_7 \rightarrow n_9 \rightarrow n_{10} \rightarrow n_{16} \rightarrow n_{19} \rightarrow T$ and $S \rightarrow n_4 \rightarrow n_{10} \rightarrow n_{15} \rightarrow n_{14} \rightarrow n_{18} \rightarrow T$. The sole possible crossover point between p_1 and p_2 is n_{10} as the shared node. The new paths p'_1 and p'_2 would then be $S \rightarrow n_3 \rightarrow n_7 \rightarrow n_9 \rightarrow n_{10} \rightarrow n_{15} \rightarrow n_{14} \rightarrow n_{18} \rightarrow T$ and $S \rightarrow n_4 \rightarrow n_{10} \rightarrow n_{16} \rightarrow n_{19} \rightarrow T$, respectively. In case of multiple crossover points, one crossover point is randomly selected. Issues with this approach arises when:
  - Parent chromosomes does not share a common node between them. In this situation, the parent chromosomes are usually copied to the next generation.
  - Both parent chromosomes share the common node at the same locus. In this situation, the resulting offspring chromosomes may be very similar or even identical to the parents.

- **Partially Mapped Crossover:** It is a permutation based crossover technique that guarantees each element in the offspring will be present exactly once, thus eliminating the need for a repair function (Chitra & Subbaraj 2012). First, two parent chromosomes are selected and for each chromosome, two random cut points are set to define a mapping section (sequence of genes between the cut points). Then the mapping section of both parents are copied into empty structures of two new chromosomes. The chromosome with the mapping section of first parent, is completed using the remaining nodes of the second parent in a left-to-right fashion. Similarly, the chromosome with the mapping section of second parent, is completed using the remaining nodes of the first parent. Predefined mapping is used to handle chromosome validity and duplicate nodes.

- **Same Adjacency Crossover:** Qiongbing & Lixin (2016) proposed this crossover mechanism to address the issue of requiring a common node between the two parent chromosomes in NBC. Instead of relying on a shared node, this approach performs crossover using the neighbours of the node at a crossover point. Consider an example where p_1 and p_2 are, $S \rightarrow n_1 \ ... \ n_{i-1} \rightarrow n_i \ ... \ T$ and $S \rightarrow n_1 \ ... \ n_{j-1} \rightarrow n_j \ ... \ T$, respectively. For $(n_{i-1} \rightarrow n_i)$ in p_1 and $(n_{j-1} \rightarrow n_j)$ in p_2, if node $n_i \in N(n_{j-1})$ (n_i is a neighbour of n_{j-1}) and $n_j \in N(n_{i-1})$ (n_j is a neighbour of n_{i-1}), then the edge pair (n_{i-1}, n_{j-1}) can be selected as the crossover point. This means that the two paths can be crossed at this point, ensuring that the resulting paths remain valid and connected. This approach proved very efficient in generating a more diverse (less overlapping) set of paths as it allows for crossover to be performed even on parent chromosomes that do not share a common node. A repair function is used, to eliminate loops.

Liu et al. (2016) also proposed a crossover strategy, represented as Node based (I) in Table 2.4, that bears a strong resemblance with NBC, relying on the existence of a common node between two parent chromosomes. Similarly, Hamed (2010) employed Cut point crossover, which also shares similarities with the NBC. The GAs proposed in (Leung et al. 1998, Inagaki et al. 1999, Mohammed et al. 2017) employed One-point crossover, Locus-based crossover, and Hybrid crossover respectively. These crossover operators were tailored for specific types of chromosomes like fixed-length chromosomes, binary chromosomes, and others. Hence, will not be discussed further.

NBC is the predominantly used crossover technique in GAs on routing. However, the constraint of a shared node between both parents for a successful crossover operation introduces bias and degrades the evolutionary process. Also, if both chromosomes have a common node at the same locus, then the resulting offspring could be identical to the parents. In Chapters 4.5 and 5.5, we observed that this risk is much lower when working on a smaller networks with a small population size, in comparison to when working on a bigger network with a larger population size. This decreases the diversity in the population and increases the risk of premature convergence (Riget & Vesterstrøm 2002), often requiring a larger population size to mitigate this risk (Shukla et al. 2015).

2.7.5 Mutation method

The mutation operator in GAs is typically used to introduce diversity in the population by randomly altering the genes of a chromosome. This can help prevent premature convergence. However, in routing problems, general mutation operators like bit mutation and swap mutation are not suitable because they cannot always generate valid paths. As a result, a specific mutation operator referred to as Path Mutation (PM) is used.

- **Path Mutation (PM):** Imagine, we have a parent chromosome with source node S and destination node T. To produce a new chromosome using this, a random node n (excluding the source and destination node) is selected. Parent chromosome is divided into two parts, *a:* S to N and *b:* N to T. A new path c is generated using the initialisation method of the respective approach, *c:* N to T. Then, a new chromosome is formed by combining a and c. A repair function on the new chromosome is employed to remove any duplicates. To illustrate the process of creating a new path using PM, an example is given in Figure 2.10. Alternatively, the new path c could also be generated as, *c:* S to N and in that case the new chromosome would be formed by combining the genes of c and b, respectively.

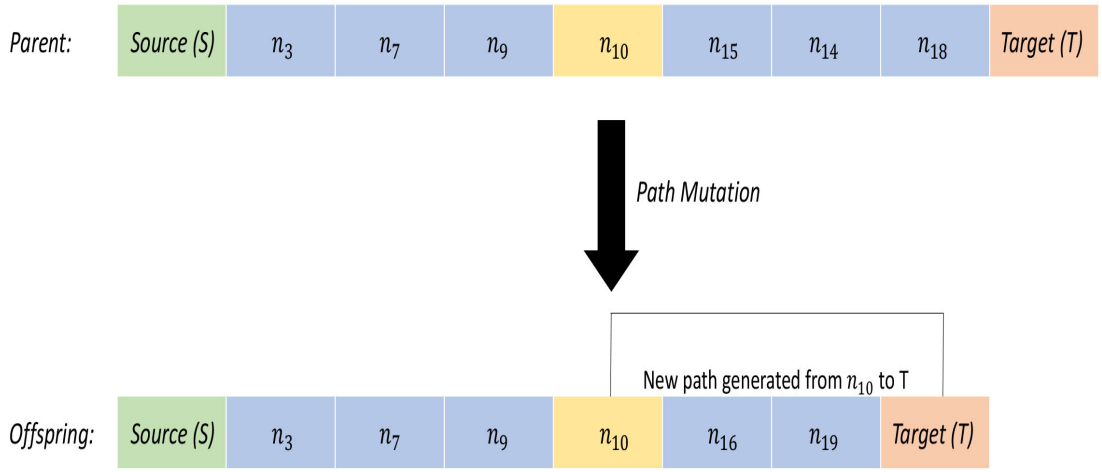

Figure 2.10: Path Mutation (PM).

Similar to the crossover operators, mutation operators such as Swap mutation (Gen et al. 1997), Bit mutation (Leung et al. 1998) and Adaptive mutation (Mohammed et al. 2017) were tailored with specific constraints. Therefore, will not be discussed further. A mutation operator similar to Path Mutation was used by Nanayakkara et al. (2007) and Liu et al. (2016), where two nodes in the parent chromosome are selected, and the segment between them is replaced with a newly generated path. The mutation strategy in (Munetomo et al. 1998) used Dijkstras algorithm to produce the partial route and this added to the methods growing computational and temporal complexity. The majority of GAs given in Table 2.4 used PM as the mutation operator to introduce diversity. However, in GAs on routing, this operator is used to produce only a few chromosomes, and any potential bias introduced by it can be disregarded (Ahn & Ramakrishna 2002).

2.7.6 Selection method

The selection operator in GAs is widely recognised as one of the most crucial operators (Razali et al. 2011, Jebari & Madiafi 2013). The selection operator determines the solutions to produce offspring and the solutions to move to the next generation. This operator also increases and maintains the average quality of solutions in each generation (Ahn & Ramakrishna 2002). Most importantly it guides the convergence of the GA. If the GA selects only the top-performing solutions the algorithm will suffer from premature convergence due to the lack of diversity. On the other hand if too many solutions get selected the algorithm will have a slow convergence rate and this results in high computational cost. It is worth noting that, unlike crossover and mutation, the selection operator is a generic operator and is not unique to the SP routing problem. As a result, GAs on routing frequently use popular selection operators like

- **Tournament selection:-** It is one of the most commonly employed parent selection method, where a group of individuals (typically two or more) is randomly selected from the population, and the one with the highest fitness is selected as the parent (Eiben & Smith 2015). In this thesis, the algorithm proposed by Ahn & Ramakrishna (2002) and NSGA-II (Deb et al. 2002) employ tournament selection as the parent selection method.

- **Roulette wheel selection:-** It is also a widely used parent selection method, where each individual is selected using its selection probability (Eiben & Smith 2015), with higher fitness corresponding to a higher probability of selection. In this thesis, the algorithm proposed by Qiongbing & Lixin (2016) employs roulette wheel selection as the parent selection method.

- **Elitist selection:-** Elitism is a strategy used to retain top performing solutions across generations, ensuring that the best individuals are not lost during the evolutionary process (Purshouse & Fleming 2002). Using this method, a certain number of the best performers (based on fitness) are passed directly on to the following generation, maintaining solution quality and protecting against loss from stochastic variation. In this thesis, NSGA-II (Deb et al. 2002), NSGA-III (Jain & Deb 2013, Deb & Jain 2013), and the proposed Parallel Optimal-route Search (POS) algorithm introduced in Chapter 5 employ elitist selection by retaining the best performing individuals across combined parent and offspring populations.

- **Linear ranking selection:-** It is a parent selection method used in GAs, where individuals are selected based on their ranks within the population rather than their raw fitness values (Wang et al. 2014). This method is particularly useful in scenarios where the fitness values of individuals differ by small margins, for example, in shortest path routing problems, where many candidate solutions may have similar path lengths. Linear ranking is a type of ordinal-based selection method (Wang et al. 2014), which is employed in our POS algorithm described in Chapter 5.

Overall, tournament selection and roulette wheel selection were the most commonly employed selection strategies, often used with replacement to maintain diversity and reduce redundancy. Elitist selection strategies are widely adopted in multi-objective systems (Chitra & Subbaraj 2012), particularly where algorithms like NSGA-II (Deb et al. 2002) are used for promoting faster convergence.

2.8 Identified gaps in the literature

The motivation for this research is to demonstrate the ability of GAs to simultaneously generate, optimise, and retrieve multiple solutions on complex, large-scale networks. A thorough review of the literature reveals key limitations in existing approaches. These gaps not only limit the scope of existing findings but also highlight potential directions for future investigation. The identified gaps are as follows:

- Most existing studies applying GAs to routing problems are limited to small-scale network topologies, raising questions about scalability and real-world applicability.
- Genetic operators used in GAs for path routing often suffer from instability, as small modifications introduced during crossover or mutation can cause disproportionately large changes in path cost. This highlights the need for more effective and problem specific genetic operators tailored to the structure and constraints of the routing problems. Many studies employ Node Based Crossover (NBC) (Ahn & Ramakrishna 2002) operator without sufficiently addressing their reliance on shared nodes between parent chromosomes, which significantly constrains the generation of diverse (less overlapping) and viable solutions. Additionally, Path Mutation (PM) is often underutilised due to its high randomness.
- When applying GAs to path routing problems, especially those focused on generating the shortest paths, preserving high quality solutions during evolution is critical. However, many studies rely on selection methods like Tournament selection and Roulette wheel selection, which can inadvertently discard optimal paths. While these strategies promote diversity, they may compromise convergence and solution quality highlighting the need for selection mechanisms that better balance exploration with the preservation of elite solutions.
- Conventional GAs often face challenges such as slow convergence and computational inefficiency, which constrain their effectiveness in real-time or large-scale routing applications. Although PGAs have the potential to address these limitations by offering faster convergence and greater scalability, they remain a relatively unexplored method within the context of routing problems.
- As discussed in Section 2.3, although extensive research exists on the Vehicle Routing Problem (VRP) and its numerous variants, only a limited number of studies specifically address vehicle routing from the perspective of public service or general societal needs.

- Despite frequent mention of GAs for generating multiple solutions in routing tasks, there is a lack of emphasis on producing diverse (less overlapping) paths that could improve robustness and load balancing.

In this thesis, we introduce two Parallel Genetic Algorithms (PGAs). The first, called the MultiPath Island-Based Genetic Algorithm (MIBGA) employs the island model and is specifically developed for the K-Most Diverse Near-Shortest Paths (KMDNSP) problem, as detailed in Chapter 4. The second, the Parallel Optimal-route Search (POS) algorithm utilises a hybrid approach that integrates global parallelisation with island-based features. It is applied to a multi-criteria VRP designed to address the needs of the general public, as discussed in Chapter 5. The performance of MIBGA is evaluated against well-established genetic algorithms for routing introduced in (Ahn & Ramakrishna 2002, Qiongbing & Lixin 2016), while POS is compared with two of the most widely used Multi-Objective Genetic Algorithms, NSGA-II (Deb et al. 2002) and NSGA-III (Deb & Jain 2013). Both MIBGA and POS are tested on multiple large-scale, real-world road networks, which are described in detail in the following chapter.

Part II

Contributions of this Thesis.

Chapter 3

Experimental Setup

Related publications:

[1] Sharma, H., Mooney, P. and Galván, E., 2023. A method for creating complex real-world networks using ESRI Shapefiles. MethodsX, 11, p.102426, Link to paper.

[2] Sharma, H., Mooney, P. and Galván, E., 2025. GeoGraphNetworks: Shapefile-Derived Datasets for Accurate and Scalable Graphical Representations, Link to dataset.

3.1 Introduction

Before we proceed to the description of our Genetic Algorithms (GAs), we need to provide some background and information about our experimental setup. In this chapter, we shall illustrate the type of graph data structure or graphical representation used for the evaluation of our GA approaches in Chapters 4, and 5. Section 3.2 emphasises the critical role of geographical systems, specifically Environmental Systems Research Institute (ESRI) Shapefiles, in representing and analysing geospatial data for applications like urban planning and environmental management. Following on, in Section 3.3, we describe the open-source method we have developed to create complex real world road networks from ESRI Shapefiles (Sharma et al. 2023). This allows us to obtain real-world road networks from online-based sources and convert them to a Python NetworkX[1] compatible data structure. Sections 3.4, 3.5, and 3.6 provides an overview of the introduced network repository, its validation, and usage notes, respectively. Section 3.7 outlines the overall software implementation, including key information about the programming language, libraries, platform, resources, and other critical factors involved in developing these networks. Finally, in Section 3.8 we provide a brief summary of this chapter.

3.2 Geographical systems

Geographical systems (Karduni et al. 2016a) play a vital role in providing accurate and comprehensive data for analysis, mapping, and decision-making in various fields such as

[1] https://networkx.org/documentation/stable/tutorial.html

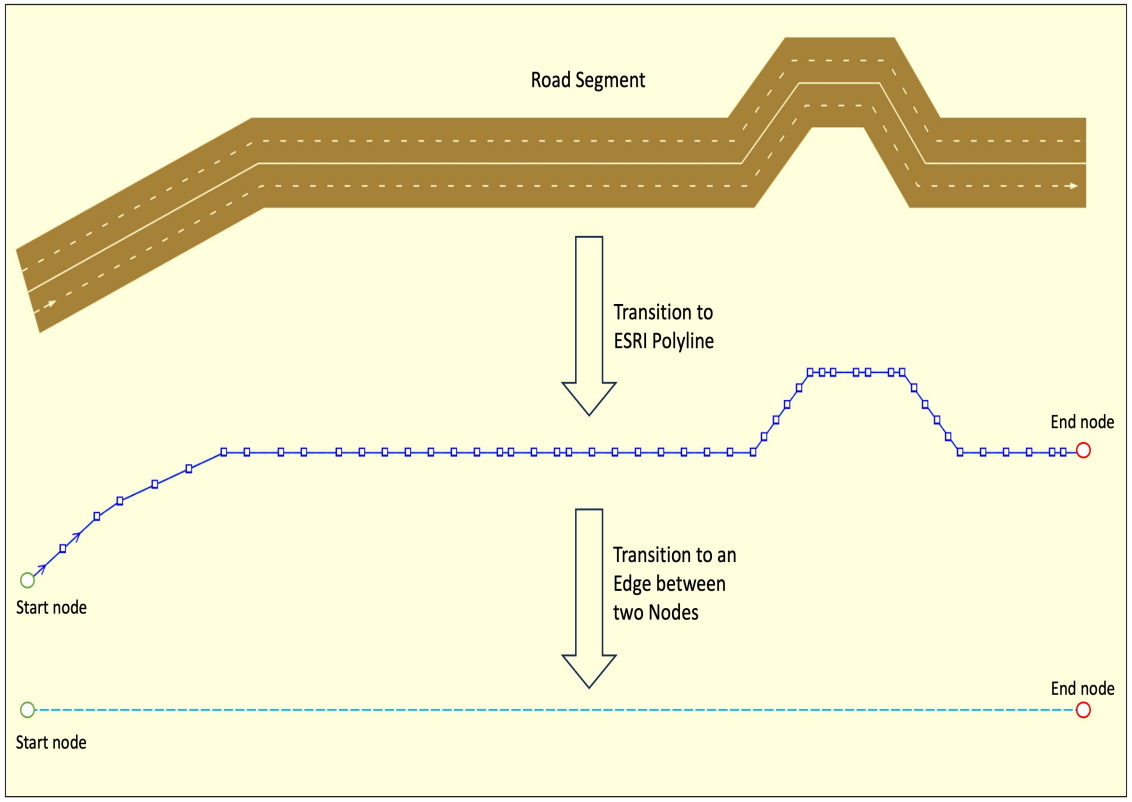

Figure 3.1: Example of a polyline representation as an edge in a graph.

emergency response, urban planning, transportation, and environmental management. A fundamental prerequisite for geographical systems is the availability of precise data that accurately represents geographic objects and features, including their positions, areas, and distances from other objects. For a long time, one of the most dependable data sources for collecting and storing such geographic objects and features has been ESRI Shapefiles. These Shapefiles store geospatial data, such as buildings, traffic signals, railway lines, roads, rivers, and more, using a vector data format that utilises different geometries like polygons, polylines, and points for representation (Sharma et al. 2023).

To fully understand the significance of the datasets used in the thesis, it is essential to comprehend the storage of road, railway lines, and river data in ESRI Shapefiles versus the graphical networks used in graph theory (Euler 1741, Karduni et al. 2016a). To illustrate this, we present Figure 3.1, showing the transformation of a road segment (same principle applies to railway lines and rivers) into a polyline geometry, with points connected by line segments, and then into a single edge between two nodes in a graphical network. The crucial point to note here is the understanding of how the same road having the same length is represented with just two nodes in the graphical network. Understanding this concept is crucial as it significantly reduces the size of the network in terms of nodes and edges, leading to faster processing times by algorithms using these networks. For instance, the route pruning algorithm (Schweimer et al. 2021) used in graph construction currently operates with a time complexity of $O(n^3)$. Meanwhile, Dijkstra's algorithm has a current space complexity of $O(n)$ and a time complexity of $O((n+m)logn)$, where n and m denote the number of nodes and edges, respectively.

A plethora of global geographic data, including detailed road, rail, and rivers networks, has been produced in ESRI Shapefile format through crowd-sourced sources like OpenStreetMap (OSM) (Karduni et al. 2016a). Several road network graph datasets created using OSM data are available. Notable examples include the dataset of 80 most populated cities (Karduni et al. 2016b), a Global Feature-Rich Network Dataset (Yap & Biljecki 2023b) covering 50 cities, and Stanford Large Network Dataset (Rossi & Ahmed 2015a,b) which provides unweighted (without length) road graph files for California, Pennsylvania, and Texas. However, the data contributions to OSM mainly rely on technical instruments like GPS receivers, laptops/PCs, and smartphones (Kaur et al. 2017). Consequently, the contributed data may suffer from quality issues (Kaur et al. 2017, Fonte & Martinho 2017, Forghani & Delavar 2014, Basiri et al. 2016). To avoid such data quality issues, government agencies of almost all countries store, maintain, and publish network data of roads, rails, and rivers in ESRI Shapefile format.

Numerous tools and software are currently available for researchers to conduct network and graph analysis on ESRI Shapefiles. Quantum Geographic Information System (QGIS) and Arc Geographic Information System (ArcGIS) are two of the most widely used tools for interacting with ESRI Shapefiles (Karduni et al. 2016a). Both tools are renowned for their excellent visualisation capabilities, allowing users to easily combine multiple Shapefile layers with a simple drag and drop. Several programming languages offer various libraries and packages for handling spatial data in ESRI Shapefile format, as well as graphical format involving nodes and edges for example, Python provides libraries such as GeoPandas (Jordahl et al. 2021), Shapely (Gillies 2013), Urbanity (Yap & Biljecki 2023a), NetworkX (Hagberg et al. 2008), OSMnx (Boeing 2017), pygraphviz (available at), and igraph (Csardi & Nepusz 2006), R provides packages like sf (Pebesma et al. 2018), sp (Pebesma & Bivand 2005), igraph (Csardi 2013), network (Butts 2008), and dagitty (Textor et al. 2016), Java offers libraries like GeoTools (Turton 2008) and JGraphT (Michail et al. 2020), and others. All these libraries and packages are widely known for the advanced analytical operations they offer. However, unlike QGIS and ArcGIS, they lack a user-friendly GUI and require users to have some expertise to use ESRI Shapefiles within these language environments.

As discussed in Chapter 2.7, network size and configuration significantly impact the design and analysis of algorithmic approaches to path routing problems. Accordingly, a key contribution of this work is testing our GAs on real-world networks. ESRI Shapefiles containing polyline geometries can be transformed into graphical networks using ArcGIS (Karduni et al. 2016a), Urbanity (Yap & Biljecki 2023a), and OSMnx (Boeing 2017). ArcGIS (Karduni et al. 2016a), a subscription-based tool that can be financially very expensive, particularly when generating multiple large networks like those used in this thesis. Urbanity (Yap & Biljecki 2023a) and OSMnx (Boeing 2017) focus primarily on using OSM data, allowing users to download and visualise this data as graphical networks. However, these tools are not specifically designed for working with pre-existing Shapefile data, which may necessitate substantial preprocessing or conversion before they can be used.

Consequently, the networks used in this thesis are constructed from the ground up utilising the method detailed in Section 3.3, which presents a simplified method for converting ESRI Shapefile data into graph representations using Python (detailed in Section 3.7). For users handling geographic network data, this open-source approach enhances accessibility and flexibility over proprietary solutions like ArcGIS, but it requires functional Python programming knowledge. The NetworkX package is used to create our graph or network data structure. The ESRI Shapefiles used in this thesis are from the open data provided by the US Census Bureau[2] which offers ESRI Shapefiles for road networks of every state in the United States of America (USA). This method is reusable and reproducible for others tackling the same problem. Using this method, researchers will be able to create efficient and accurate graph representations of real-world networks to test their graph theory approaches. To replicate this method, an ESRI Shapefile consisting of polylines in a geometry format data is required.

3.3 Working with graph data structures

Python packages NetworkX (Hagberg et al. 2008) and OSMnx (Boeing 2017) are probably the most popular approaches in industry for creating and analysing real world graphical networks. However, generating such networks from raw geospatial data remains a non-trivial process, as these libraries require data to be pre-formatted in specific structures, and do not natively support direct conversion from ESRI Shapefiles. While Karduni et al. (2016a) outlined high-level procedures for transforming Shapefiles into graph networks and released their implementation as a tool within the proprietary ArcGIS platform, this approach still relies on commercial software and does not offer an open-source, reproducible, code-based pipeline.

With this method, we address this gap by presenting a fully open-source workflow for converting raw Shapefile data into NetworkX-compatible graph representations. The workflow documents the complete process from dataset selection and extraction to cleaning, transformation, and edge-list construction allowing users to generate large-scale, reproducible routing networks without dependence on proprietary GIS tools. The method is organised into three main phases: Data Gathering, Pre-processing, and Transformation, each of which is described in detail in the sections that follow.

3.3.1 Data gathering

The initial phase of our workflow involved data gathering, a non-trivial task given the requirements such as credible data source, size, and scope of the datasets introduced. We created the following specialised factors and conditions for the data collection process in order to give a consistent and uniform procedure:

- **Selection of Country:** USA is selected as the focus area due to its extensive net-

[2] https://www.census.gov/programs-surveys/geography/guidance/tiger-data-products-guide.html

work coverage across the entire country and the availability of open-source geospatial data. The ease of data access and the US Census Bureau policies of frequently updating its repositories further supported USA as an ideal choice for this thesis.

- **Data Source:** To ensure high quality datasets in this thesis, we relied solely on data repositories from reputable sources. The US Census Bureau supports the government, private sector, and public by conducting official surveys and providing accurate, up-to-date geographic and demographic data, including mapping resources like TIGER/Line files.
- **Industry-Standard GeoSpatial Data:** We also aimed to ensure that the ESRI Shapefiles we are using have a track record of being utilised in academic research. For instance, Tiger/line dataset has been cited in several studies (Her & Yu 2021, Walker 2016).
- **Availability:** All the ESRI Shapefiles used in this thesis are open source.
- **Coverage:** We selected ESRI Shapefiles that provide complete national coverage.

3.3.2 Pre-processing

A graph is a set of edges that connect two nodes (also known as vertices), mathematically described in Section 2.2 of Chapter 2. In a graph, each node is positioned using the longitude (position on the x-axis) and latitude (position on the y-axis) coordinates. Each edge in the graph is assigned a distance (weight), which represents the cost of moving from one node to another along that edge. Euclidean distance between two nodes $X_{(x_1,y_1)}$ and $Y_{(x_2,y_2)}$ is calculated using Equation 3.1, where (x_1, y_1) and (x_2, y_2) represent the longitude and latitude coordinates of nodes X and Y, respectively.

$$Distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3.1}$$

As the position of the nodes and the distance along the edges depend only on the longitude and latitude coordinates, the first stage of the pre-processing phase is to remove the elevation coordinate (position on the z-axis), if present in the ESRI Shapefiles used in our thesis. Next, we need to extract the actual lengths of the road segments in metres. This step is crucial because longitude and latitude coordinates in ESRI Shapefiles are not always stored in a meter-based projection system. If they are not, directly using these coordinates in Equation 3.1 will yield distances in degrees, which are typically unsuitable for most applications. To address this, conversion to practical units such as metres is required to ensure that the polyline geometries stored in ESRI Shapefiles are expressed in a meter-based projection system. The European Petroleum Survey Group (EPSG) is a non-profit organisation that keeps and maintains geographic records using standard codes known as EPSG codes, which define the Coordinate Reference System (CRS) such as degrees or metres, for ESRI Shapefile projections. The metre-based projection code for the geographic features in the USA is EPSG: 2163. The database of EPSG codes can be searched for further details.

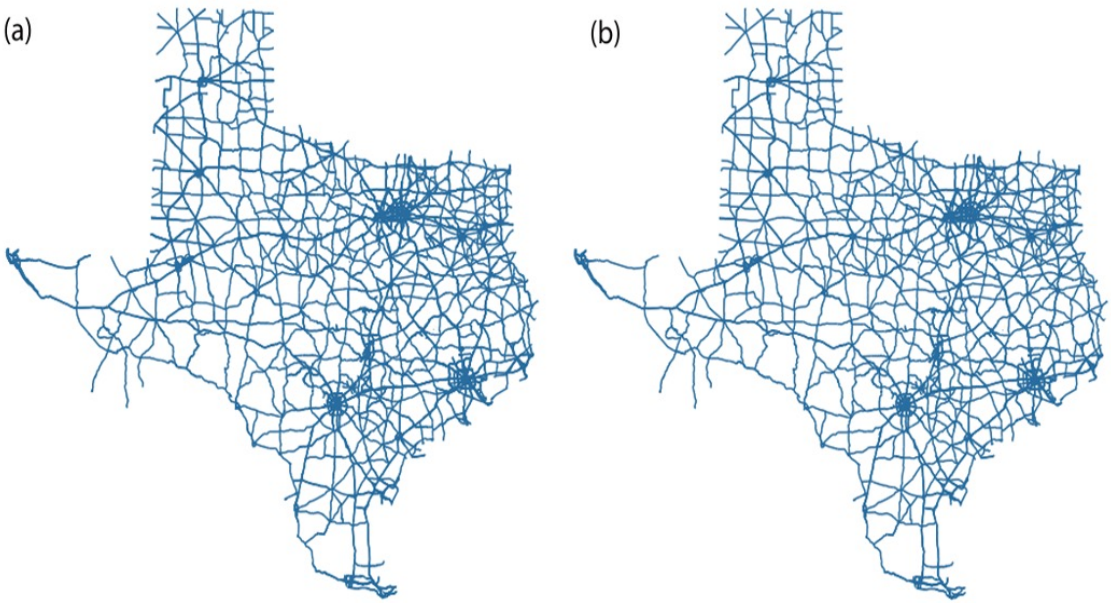

Figure 3.2: (a) ESRI Shapefile representation of Texas before data cleaning, showing unprocessed geometries. (b) ESRI Shapefile representation of Texas after data cleaning, with duplicate and unnecessary polyline geometries removed.

Data cleaning is the last stage of our pre-processing phase. The cleaning stage includes not only finding and eliminating duplicate records, but also finding and eliminating multiple entries that might not make a substantial contribution to the network. ESRI Shapefiles use polyline geometries to store linear features like road segments, railway line segments, river watercourse, and others. These geometries are highly efficient for storing and representing real-world networks, along with all their features and attributes. However in graphical networks, these features are expressed using a single edge between two nodes, as shown in Figure 3.1. Therefore, some features or attributes which might be very crucial to an network in ESRI Shapefile become inadequate to a graphical network. We considered a polyline inadequate when:

- One polyline lies completely within another polyline. For instance, when a new road segment is built over an pre-existing road, representing an overpass.
- One polyline contains a piece of another polyline (more than one consecutive point). This situation is exemplified by city road segments that are part of major highways.
- One polyline intersects with a same polyline at more than one location. For instance, a road segment that crosses a highway twice while running parallel to it.

By employing the aforementioned criteria, we can eliminate a significant portion of the road segments from our graph without changing the graphs overall structure. For instance, the road network of Texas comprising the primary and secondary roads had 15,257 road segments. After removing the polylines complying with the aforementioned conditions, we had 7,997 (only 52.4% of the original) road segments without any considerable difference between the networks, as shown in Figure 3.2.

3.3.3 Transformation method

This section describes the complete process, step-by-step, of converting a network from an ESRI Shapefile format to a graphical network format. The process is described using the Python programming language and its associated libraries. However, the general idea remains the same and could be implemented using other programming languages. The steps involved in transforming a polyline network from an ESRI Shapefile to an graphical network of nodes and edges are as follow:

1. An ESRI Shapefile containing a polyline network is a prerequisite. It can be downloaded from the US Census Bureau data repositories.
2. Read the respective ESRI Shapefile using GeoPandas (Jordahl et al. 2021) library.
3. Drop the elevation coordinate (if present) from all polyline geometries in the network.
4. Transform the EPSG code of the stored geometries to a meter-based projection system if necessary. GeoPandas (Jordahl et al. 2021) provides the functionality for the transformation.
5. Remove all polylines that lies within another polyline. This can be achieved using the functionalities from GeoPandas (Jordahl et al. 2021).
6. Extract the coordinates of the first and last node of each polyline using Shapely (Gillies 2013) and create a set of edges.
7. Using GeoPandas (Jordahl et al. 2021), track the intersection of each polyline with others. Only if the intersecting geometry is a point, add two edges (from the first node to the intersection node, and from the intersection node to the last node) to our set of edges. This step will eliminate the inadequate polylines as per the conditions specified in our data cleaning strategy.
8. Remove any duplicate edges, if formed.
9. Using Shapely (Gillies 2013), calculate the length of each edge (in metres). If correct EPSG code is not used, length assigned will not be correct.
10. Download or save your network in the edge list format, as shown in Table 3.1, using a Pandas DataFrame or saving as a CSV or XLSX file for reusability.

With data in the edge list format as shown in Table 3.1, graphical representations of any network can be created in multilingual programming environments. Figure 3.3 provides a detailed description of the overall process and Figure 3.4 is used to demonstrate a practical example of our transformation, where the first layer represents the ESRI Shapefile, the second layer shows the nodes extracted from it, and the final layer represents the edges connecting these nodes. Figure 3.5 displays the Shapefile representation of Texas after cleaning (left) and the graph visualisation of Texas using NetworkX (right), respectively. Data cleaning steps described in Section 3.3.2 reduce the network size by 48% but still the graphical representation obtained consists of 10,886 unique nodes and 24,464 edges.

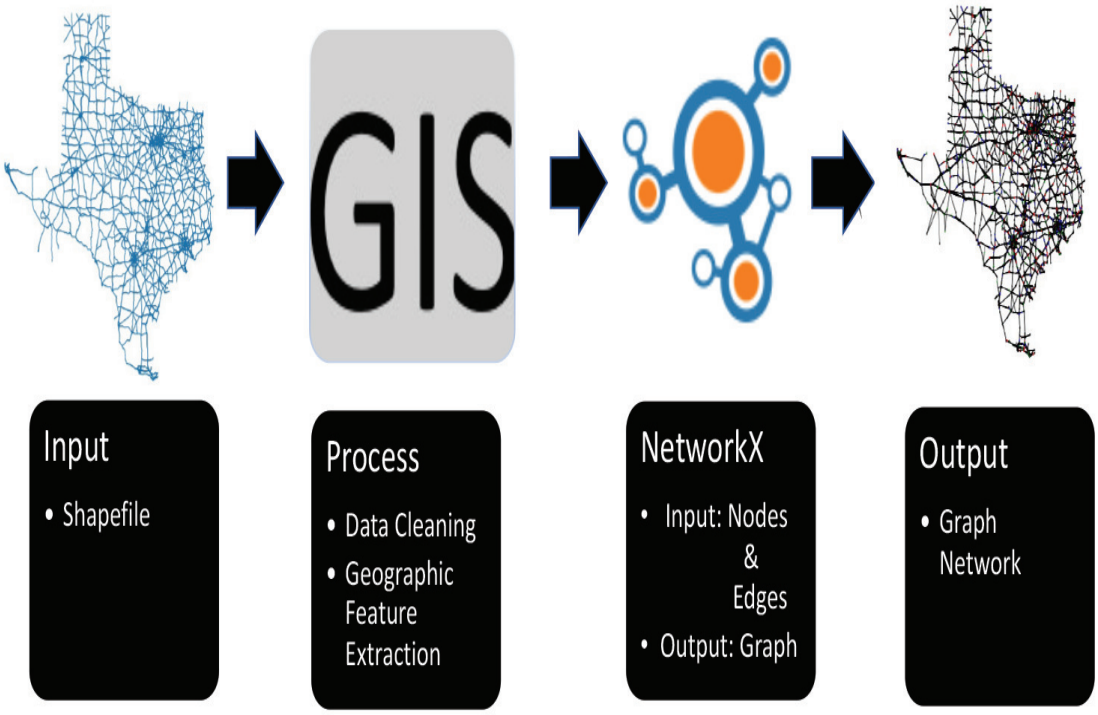

Figure 3.3: A flowchart indicating the main steps in creating a graphical network in NetworkX from an ESRI Shapefile.

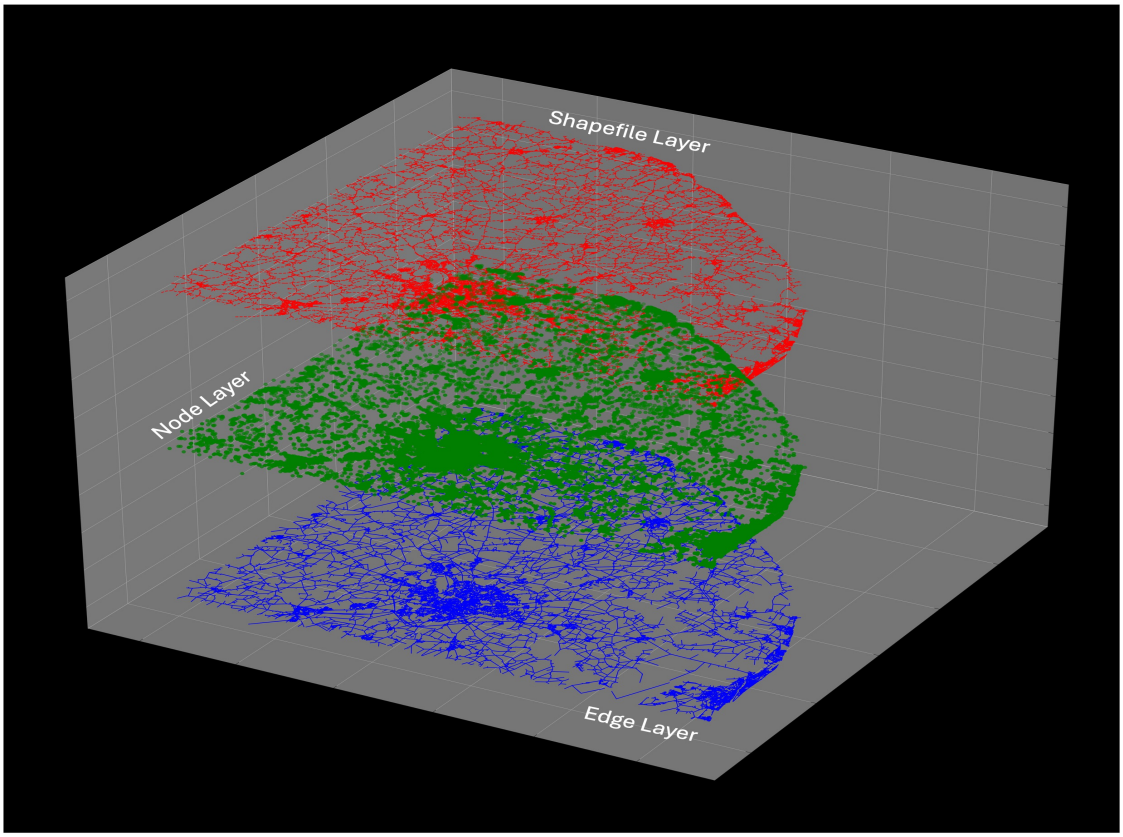

Figure 3.4: This example demonstrates the transformation method in action. The top layer represents the ESRI Shapefile, the middle layer displays the extracted nodes, and the bottom layer illustrates the edges connecting these nodes.

$XCoord_{Start}$	$YCoord_{Start}$	$XCoord_{End}$	$YCoord_{End}$	Edge Length (in Kilometres)
-118749.3966	-1585063.817	-115759.4154	-1584031.711	3.267227536
49596.98604	-1926479.922	48188.64197	-1927080.254	1.531055301
55000.00963	-1934421.636	55232.95755	-1934652.405	0.328446509
55205.99387	-1934646.97	55324.77848	-1935078.048	0.461638671
55205.99387	-1934646.97	55325.57701	-1935059.139	0.44271258
55205.99387	-1934646.97	55265.11589	-1935647.634	1.035162766

Table 3.1: Sample format for an edge list dataset, including the longitude (X) and latitude (Y) coordinates of the start and end nodes that define each edge, as well as the edge length in Kilometres. The coordinates in this table are based on the meter-based projection code EPSG: 2163. To convert them to standard longitude and latitude coordinates, a transformation to EPSG: 4269 is required.

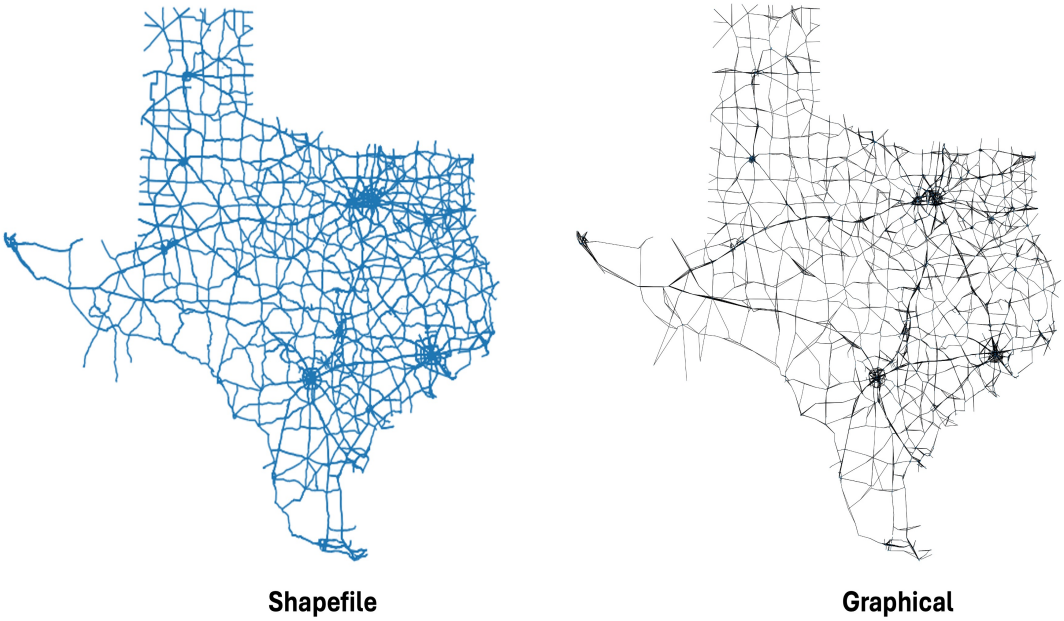

Figure 3.5: Shapefile representation of Texas after data cleaning and preprocessing (left), displaying refined geometries. The graphical representation of Texas (right) is constructed using extracted nodes and edges in NetworkX, illustrating the spatial network structure.

3.4 Network repository overview

To ensure the efficiency and effectiveness of a graph algorithm, it is critical to test it on multiple networks of varying sizes and topologies. In Chapters 4, and 5, we test the performance of our GAs using multiple networks tailored to the respective path routing problems. Frequently used networks include the road network of Texas displayed in Figure 3.5, and the road networks of Louisiana, Oklahoma, and Arkansas displayed in Figure 3.6. These examples demonstrate the versatility and reproducibility of our approach in generating graphical representations of real world networks using ESRI Shapefiles.

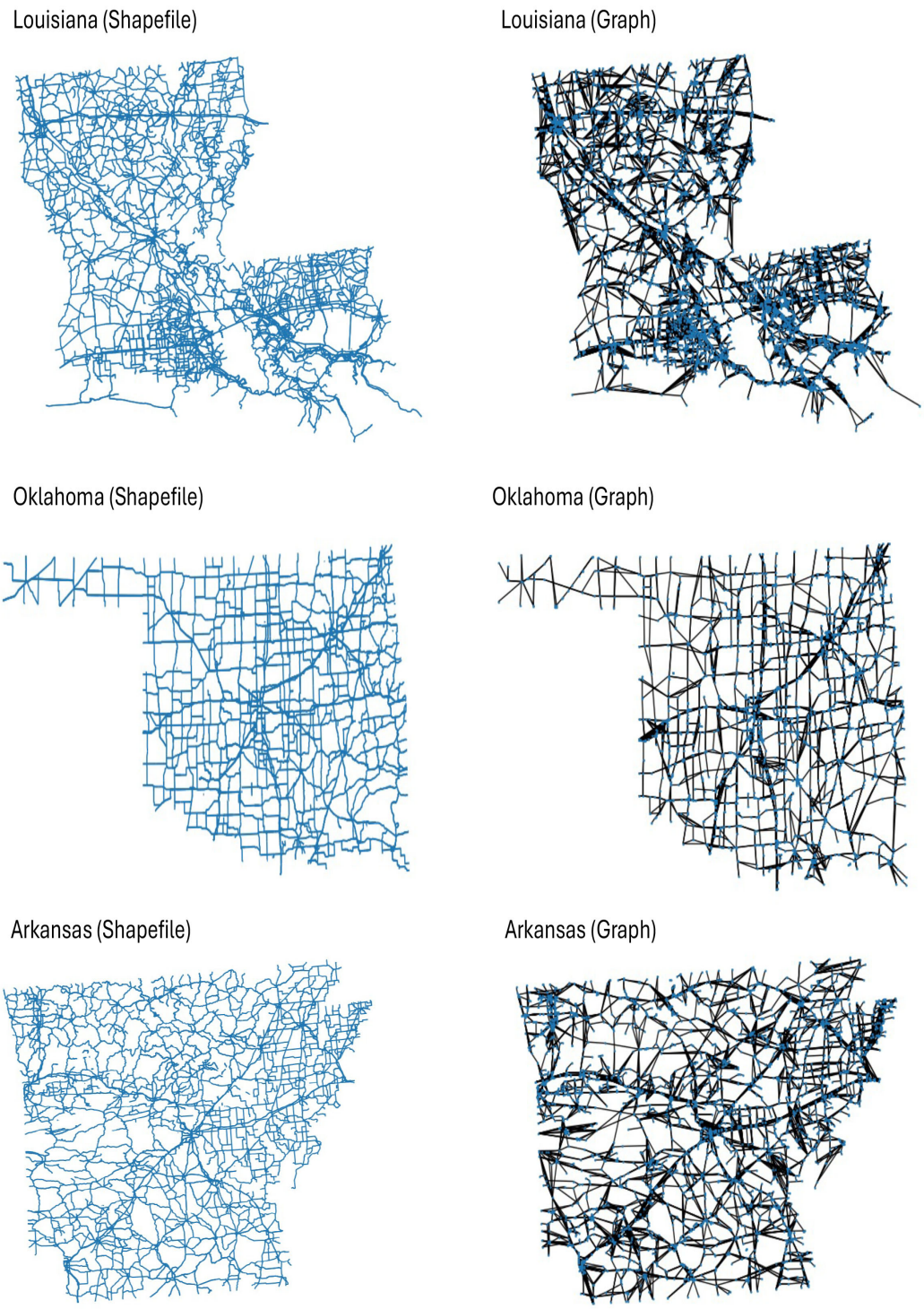

Figure 3.6: On the left, we present the Shapefile representations of Louisiana, Oklahoma, and Arkansas, respectively. On the right, their graphical representations are shown using NetworkX.

Furthermore, to promote the development of this rapidly growing field of study, we introduce **GeoGraphNetworks (Sharma et al. 2024)**, a data repository that offers comprehensive and publicly accessible spatial network resources for the road and rail networks of the USA and the road and river networks of Great Britain (GB). Similar to the selection of the USA, GB is selected as the focus areas because its networks comprehen-

sively cover the entire country, and open-source geospatial data is readily available. In GB, Ordnance Survey (OS) is the national mapping agency that serves the government, private sector, and public by conducting official surveys and providing accurate, up-to-date geographic data. Therefore, to ensure dataset quality, the ESRI Shapefiles for the road and river network of GB are sourced exclusively from OS. Additionally, the frequent updates to repositories by OS further establish GB as the ideal choice for the introduced repository, alongside the USA.

Figure 3.7: Complete river network of Great Britain (GB) as a NetworkX (Hagberg et al. 2008) graph, featuring 194,974 unique nodes and 194,951 unique edges.

ESRI Shapefile data used to create the networks of GB can be downloaded from OS Open Roads and Open Rivers, while the networks for the USA were developed using the data provided by the US Census Bureau (Tiger Data) and the North American Rail Network (NARN Lines). In the USA, the road infrastructure consists of primary and secondary roads, covering all 50 states, 1 federal district (Washington, D.C.), and 5 territories, resulting in a total of 56 networks. Additionally, the rail line infrastructure is represented as a single network that spans the entire country and includes connectivity to Canada. These networks are provided as 114 files, 57 in Excel (.XLSX) format and 57 in JSON format comprising 56 road networks and 1 rail network. The dataset for GB includes a total of 106 files, consisting of 53 Excel (.XLSX) and 53 JSON files, representing 52 road networks and 1 river network, illustrated in Figure 3.7. Each file is named using a unique code assigned by Ordnance Survey (OS) that divides Great Britain systematically into 100 km by 100 km tiles, as detailed in the OS Open Roads overview PDF.

The JSON files provided are graph objects generated using the popular Python library, NetworkX (Hagberg et al. 2008), and are ready for immediate use without any pre-processing. The XLSX files are provided to facilitate the creation of these graph networks using a variety of tools and languages, ensuring flexibility across different platforms.

Nodes in each graph are assigned a unique ID and positioned according to their original geospatial location. The dataset is hosted under a cc license at Figshare GeoGraphNetworks, and the Python code for generating the introduced networks is available on the GitHub profile.

The GeoGraphNetworks repository offers several advantages over existing public spatial datasets such as OpenStreetMap (OSM) dumps, Urbanity (Yap & Biljecki 2023a), and 80 cities network (Karduni et al. 2016b). OSM exports are typically large and complex, as each polyline feature is represented through numerous intermediate nodes and metadata tags, leading to substantial storage and processing overhead. In contrast, the networks in GeoGraphNetworks are stored in a simplified graph format, where each polyline is represented by its two terminal nodes, the start and the end points. This simplification leads to a substantial reduction in data size and improves computational efficiency for algorithmic benchmarking and large-scale routing experiments. Unlike Urbanity (Yap & Biljecki 2023a) and 80 cities network (Karduni et al. 2016b), which provides datasets for major cities across multiple countries, GeoGraphNetworks delivers complete national-scale coverage, including the entire road networks of the United States and Great Britain, enabling research across continuous transport systems rather than isolated city networks. Furthermore, the repository extends beyond road networks to include the complete river network of Great Britain and the complete rail network of the United States, supporting multi-domain applications such as hydrological, transportation, and environmental modelling. Collectively, these characteristics make GeoGraphNetworks a compact, validated, and algorithm ready resource that bridges the gap between raw GIS data and graph-based analytical workflows, complementing existing datasets like OSM and Urbanity by providing a cleaned, reproducible, and fully open-source alternative suitable for large-scale geospatial and network science research.

3.5 Validation of generated networks

A key advantage of using datasets from OS was that they also offered ESRI Shapefiles containing **point geometries** explicitly representing junctions within the network. This enabled us to move beyond the community-based or assumed validation used in prior repositories Karduni et al. (2016a) and instead perform a **direct quantitative validation** of the generated graph networks. Specifically, we assessed the proportion of nodes in each generated graph that aligned with official junction points in the corresponding OS Shapefiles, thereby confirming topological correctness rather than merely assuming source data quality. Figure 3.8 displays these proportions for all 52 road networks of GB included in the study. The names to each network are assigned using the OS Open Roads overview. The largest road network (termed TQ RoadLink in Figure 3.8) contained $373,079$ unique nodes and $445,192$ unique edges and over 89% of the nodes in the generated graph align with junction nodes of the ESRI Shapefile, while in the GB river network (Figure 3.7), 80.96% of nodes show this alignment.

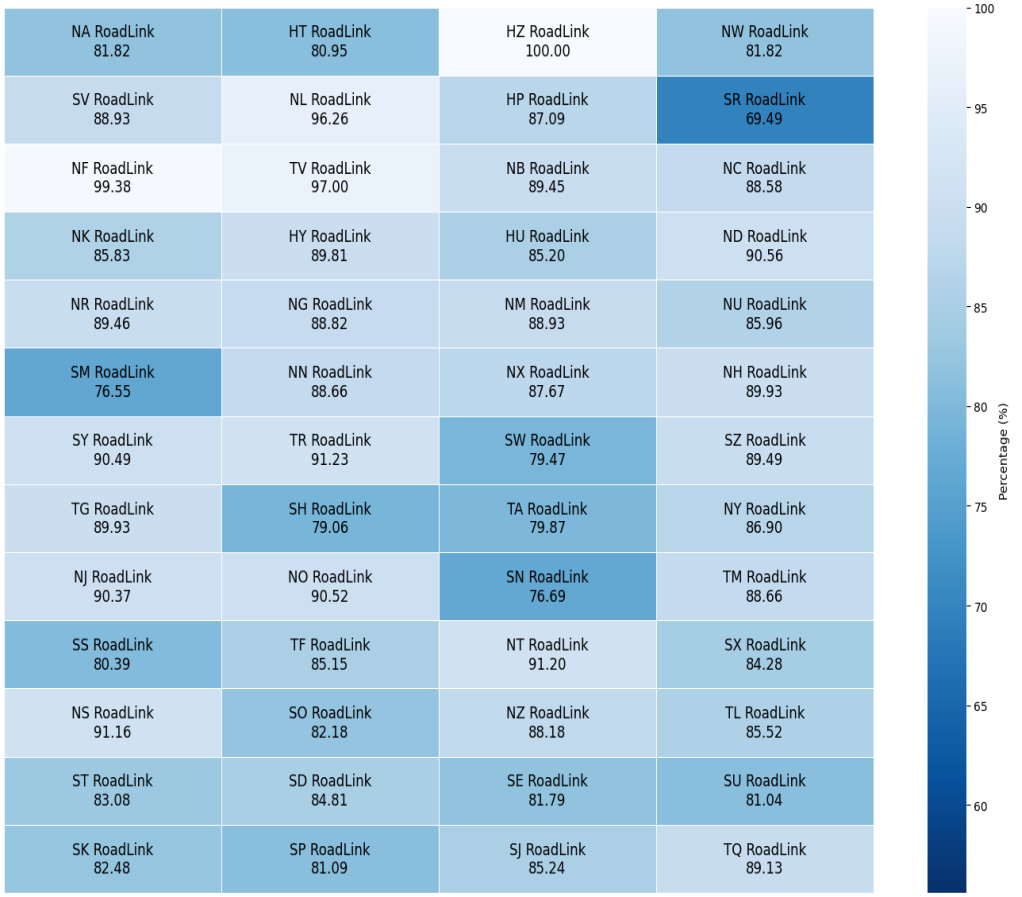

Figure 3.8: Proportion of nodes in each generated graph that match the junction points in ESRI Shapefiles for the 52 road networks analysed in this study.The names to each network are assigned using the OS Open Roads overview PDF.

No artificial nodes or edges were introduced during processing, the pipeline preserves the original geometry and semantics of the Shapefiles. Additional nodes appear only where true intersections occur, enhancing accuracy by representing each junction as a unique node. To the best of our knowledge, this study is the **first to validate national-scale, Shapefile-derived graph networks using a reproducible, quantitative, and geometry-based method**, rather than relying on only community validation and visual cross-validation. For complementary verification, we also performed visual cross-validation Yap & Biljecki (2023a), basic graph analyses, and confirmed successful back-conversion of each network into ESRI Shapefile format Karduni et al. (2016a).

3.6 Usage notes

The introduced networks can be utilised for a wide range of applications including, Transportation and Route Optimisation (Toth & Vigo 2002b, Kim et al. 2015, Laporte 2007), Urban Planning and Infrastructure Development (Yap & Biljecki 2023a), Graph and Network Science Studies (Häcker et al. 2021), Watercourse Management and Environmental Analysis (Rodriguez-Iturbe et al. 2009, Lehner et al. 2011, Lehner & Grill 2013), and oth-

ers. Since these networks represent the infrastructure of GB, they can also be combined to create larger, more complex systems. For instance, integrating multiple road networks enables intercity transportation, while combining road and river networks allows for exploring advanced topics like multimodal routing (Wolfinger et al. 2019, Park & Cheng 2023), multi-criteria location analysis (Yap & Biljecki 2023a, Sharma et al. 2025b) utilising factors like proximity to water and transportation networks to assess site suitability, where closeness to these networks serves as an influential factor and predictive analysis, such as modelling elements in the London housing market, where accessibility and connectedness to numerous infrastructures are crucial. These networks can also be used for graph machine learning (Yap & Biljecki 2023a) along with popular deep learning frameworks like PyTorch Geometric (PyG) (Fey & Lenssen 2019) and Deep Graph Library (DGL) (Wang 2019).

3.7 Overall software implementation details

The graphical representations of the real-world networks used in this thesis were constructed using the Python programming language because of its flexibility and large library of tools that are appropriate for our study goals. In the development of graphical networks, several prominent libraries played a significant role, including:

1. NetworkX (Hagberg et al. 2008)
2. Matplotlib.pyplot (Hunter 2007)
3. Pandas (McKinney et al. 2011)
4. Geopandas (Jordahl et al. 2019)
5. Shapely (Westra 2015)

Detailed in Section 3.3.3, GIS capabilities of Shapely (Westra 2015) were used to extract geographical features from ESRI Shapefiles. These features were stored in a dataframe layout using the capabilities of Pandas (McKinney et al. 2011) and Geopandas (Jordahl et al. 2019). The library utilised for network modelling and analysis was NetworkX (Hagberg et al. 2008). The visualisation of complex graphs was performed efficiently using Matplotlib.pyplot (Hunter 2007).

As we move to Chapter 4 and 5, we shall describe our experimental approaches. However, at this point, it is useful to provide some insight into our implementation details. To perform our computations and experiments, we used Jupyter Notebooks and Irish Center for High End Computing (ICHEC). A Jupyter Notebook is an open-source web-based interactive computing environment that allows users to develop and share Python Code in a collection of Small Cells forming a notebook. It is a completely free coding platform provided that relies on the CPU of the local machine to execute code. Performance depends on the processing capabilities of the local laptop such as number of cores, clock speed, and

so on. It is usually used for prototyping machine learning models but also offers robust markdown support for adding text, headings, links, and LaTeX for equations. This makes it valuable for documenting workflows and sharing findings.

ICHEC is one of the biggest High-Performance Computing (HPC) facility in Ireland. It provides super-computing resources for all academic and industrial research across the country of Ireland. To support innovative scientific research and development, ICHEC works with numerous research organisations, academic institutions, and business partners. ICHEC provides a Node based environment to its user, allowing multiple users from the same organisation to run their jobs simultaneously. This node based environment at ICHEC allow users to create personalised virtual environments for their specific jobs.

To help researchers utilise and implement their jobs properly on their cloud, ICHEC[3] provide many tutorials on topics including Check Node Utilisation, Convert Jupyter Notebooks for use on Kay, Training a PyTorch Net with GPUs, and others. For our research, we created a virtual environment including the dependencies for the aforementioned Python libraries using these tutorial resources and completed our jobs on Queue: ShmemQ that provide a maximum of 72 CPU hours per job.

3.8 Chapter summary

As previously stated, the complexity of a routing algorithm typically increases with the size of the network due to the growing number of nodes, edges, and possible paths that must be considered. Consequently, testing an algorithm on multiple networks of varying sizes and topologies is essential to assess its effectiveness and efficiency comprehensively. To evaluate the performance of our GAs, we therefore required multiple network representations of varying and sizes and topologies. To accomplish this, we presented an effective method for generating network representations using polyline geometry data from ESRI Shapefiles. Using this method, we generated multiple networks resembling real-world representations, as shown in Figure 3.5 and 3.6, and created GeoGraphNetworks, a comprehensive benchmark data repository for accurate and scalable graphical representations of the road and rail networks of the United States (USA) and the road and river networks of Great Britain (GB). In Chapter 4, the performance of our MultiPath Island-Based Genetic Algorithm (MIBGA) is evaluated on the road networks of Arizona, Washington, and Kansas, while our Parallel Optimal-route Search (POS) algorithm in Chapter 5, is evaluated on the road networks of Oklahoma, Arkansas, Louisiana, and Texas. The smallest network is Arizona with only 834 nodes and $1,547$ edges, while Texas has the largest network, with $10,886$ unique nodes and $24,464$ edges.

[3] https://www.ichec.ie/academic/national-hpc/documentation/tutorials

Chapter 4

MultiPath Island-Based Genetic Algorithm for the K-Most Diverse Near-Shortest Paths

Related publication: Sharma, H., Galván, E. and Mooney, P., 2025. MultiPath Island-Based Genetic Algorithm for the K-Most Diverse Near-Shortest Paths. Information Sciences, p.122495, Link to paper.

4.1 Introduction

In this chapter, we introduce our Parallel Genetic Algorithm (PGA) following the island model for application to the K-Most Diverse Near-Shortest Paths (KMDNSP) problem. KMDNSP focuses on identifying a set of K paths that maintain structural dissimilarity while remaining close to the shortest or optimal path in overall path length. This makes KMDNSP very well-suited for applications such as vehicle navigation requiring access to multiple diverse (less overlapping) paths rather than a single shortest path. Our approach, called the MultiPath Island-Based Genetic Algorithm (MIBGA), incorporates novel migration and selection strategies that promotes diversity in paths for this challenging routing problem where solution paths may appear identical but are significantly distinct. We show that in contrast to previous studies that employed Evolutionary Algorithms (EAs) for path routing, MIBGA prioritises path diversity in order to enhance practical usability. Extensive experiments on large, complex real-world road networks from Arizona, Washington, and Kansas demonstrate MIBGAs superior performance in terms of solution diversity, computational efficiency, and convergence speed compared to established GA based approaches. Major contributions from this chapter include,

- To the best of our knowledge, this work is the first to propose the use of Genetic Algorithms (GAs) for solving the K-Most Diverse Near-Shortest Paths (KMDNSP) problem, contributing a novel approach addressing both near-shortest paths and path diversity.

- We propose MIBGA, a novel Island-Based Genetic Algorithm (IBGA) to efficiently address the KMDNSP problem across multiple real-world road networks (described in Chapter 3).
- The structure of our MIBGA (described in Section 4.3) enabled the integration and use of strategically designed novel features, like
  - A migration strategy (described in Section 4.3.5), to facilitate the exchange of strong individuals among the islands, thereby preserving diversity while accelerating the convergence of the other islands.
  - A novel selection operator, AvgIslandFit (described in Section 4.3.9), is introduced to mitigate premature convergence and enhance solution quality. It functions as a self-adjusting selection strategy for GAs utilising the island model.
- We propose Loop-Free Path-Composer (LFPC), an independent genetic operator that does not require a shared node between the two parents for the crossover operation. This operator incorporates both crossover and mutation mechanisms (described in Sections 4.3.7 and 4.3.8, respectively), which facilitate the generation of diverse solutions and enhance the effectiveness and efficiency of our MIBGA.
- The performance of our MIBGA is evaluated against two widely known GAs by Ahn & Ramakrishna (2002) and Qiongbing & Lixin (2016).
- The results of our study justify the use of GAs for the KMDNSP problem, showing that GAs reduce average execution time as ϵ increases.
- Our results demonstrate the effectiveness of MIBGA, with less than 5% timeout instances and average computation time remaining under 10 seconds across all networks for length thresholds ($\epsilon \geq 0.10$).

Next, in Section 4.2, we provide the description of the KMDNSP problem, including notations, mathematical description, and evaluation metrics. Following this, we outline the structure of our MIBGA, starting with the variable-length chromosome representation in Section 4.3.1, a mending function (Section 4.3.2) to prevent loops, and a fitness function (Section 4.3.3) that minimises path length to ensure adherence to length constraints from Equation 4.1. Finally, Section 4.3.10 and Figure 4.6 are used to provide an illustration of the overall MIBGA approach that details the complete process from input (NetworkX graph network) to output (set of optimal paths returned by MIBGA).

4.2 K-Most Diverse Near-Shortest Paths (KMDNSP)

A graph network with a unique representation of each node and positive weights on the edges connecting them is a primary requirement for evaluating the KMDNSP problem. In our study, we use undirected graphs of three distinct real-world road networks, all of

which can be mathematically described as outlined in Chapter 3. The process of finding the KMDNSP can be performed using the following two steps:

- Finding a set of all shortest paths that falls within the user-defined path length threshold. For a path p to be considered as a nearest shortest path, it must satisfy the length constraint specified in Equation 4.1. $\epsilon \geq 0$, is the user-defined parameter that sets the allowable threshold for path length, denoted by $|p|$. For instance, if ϵ is set to 0.10 then a path p will be considered a near-shortest path, if its length is no more than 10% longer than the overall shortest path length, represented as $|p_s|$. All paths p that meet the length constraint defined in Equation 4.1, make a set of near-shortest paths represented using P_A in Equation 4.2.

$$\forall p \in P_{KMDNSP} : |p| \leq (1 + \epsilon) \cdot |p_s| \tag{4.1}$$

- From P_A selecting the subset P_{KMDNSP} containing the KMDNSP. It is performed using Div(P), as shown in Equation 4.3, which measures the diversity of a set P as the minimum pairwise dissimilarity among the contained paths (Häcker et al. 2021). The set with the maximum value of Div(P) is selected as P_{KMDNSP}. The dissimilarity between two paths p and p' (Dis(p, p')) between a given pair of source and target node can be evaluated using different functions, as presented in Table 2.2. These functions return values ranging from 0 to 1, with higher values indicating less number of shared edges between the two paths. In our study, we used $Dis_1(p, p')$ from Table 2.2 (reproduced in Equation 4.4), to measure dissimilarity between paths p and p' as it is more frequently used.

$$P_{KMDNSP} = \arg \max_{\forall P \subseteq P_A} \{Div(P)\}, \text{ with } |P| = K \tag{4.2}$$

$$Div(P) = \min_{\forall p, p' \in P} Dis(p, p') \tag{4.3}$$

$$Dis(p, p') = 1 - \frac{L(p \cap p')}{L(p \cup p')} \tag{4.4}$$

Example: Consider the weighted graph depicted in Figure 4.1. The shortest path (p_s) between the source node S and the target node T is $S \rightarrow n_2 \rightarrow T$, with a length $|p_s| = 35$. For the KMDNSP problem, with $K = 3$ and $\epsilon = 0.5$, a recommended near-shortest path must have a length $|p| \leq 52.5$. Apart from (p_s), possible solutions include, p_1: $S \rightarrow n_1 \rightarrow n_4 \rightarrow T$, p_2: $S \rightarrow n_1 \rightarrow n_3 \rightarrow T$, and p_3: $S \rightarrow n_2 \rightarrow n_4 \rightarrow T$. The path $S \rightarrow n_1 \rightarrow n_3 \rightarrow n_2 \rightarrow T$ is not considered because its length exceeded the specified threshold. The dissimilarities between the paths p_s, p_1, p_2, and p_3 are calculated as follows:

- $Dis(p_s, p_1)$, $Dis(p_s, p_2)$, and $Dis(p_2, p_3) = 1$ (no common edge)
- $Dis(p_s, p_3) = 0.75$ (common edge: $S \rightarrow n_2$)

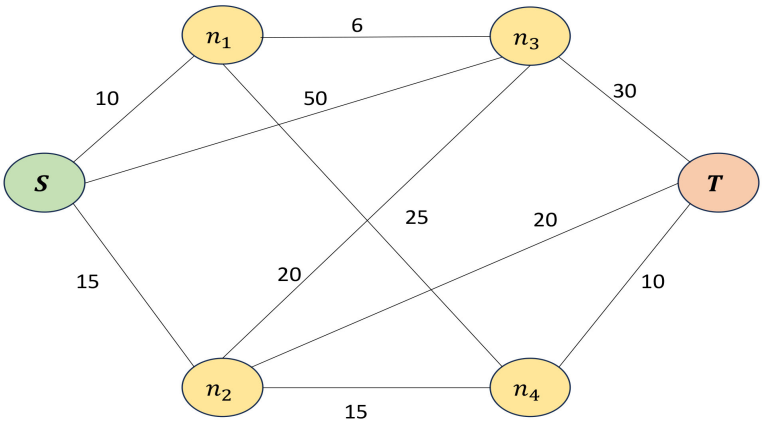

Figure 4.1: Undirected weighted graph.

- $Dis(p_1, p_2) = 0.88$ (common edge: $S \rightarrow n_1$)
- $Dis(p_1, p_3) = 0.87$ (common edge: $n_4 \rightarrow T$)

Four distinct combinations of 3 paths are possible, and their diversities are:

- (p_s, p_1, p_2): Diversity $= 0.88$, calculated as $\min(1, 0.88, 1)$.
- (p_s, p_2, p_3): Diversity $= 0.75$, calculated as $\min(1, 1, 0.75)$.
- (p_s, p_1, p_3): Diversity $= 0.75$, calculated as $\min(1, 0.87, 0.75)$.
- (p_1, p_2, p_3): Diversity $= 0.87$, calculated as $\min(0.88, 1, 0.87)$.

The solution to the given KMDNSP problem is the set containing paths (p_s, p_1, p_2), which has the highest diversity score of 0.88. The Python implementation of the above example is available as a Jupyter Notebook at MIBGA. This process of constructing all possible paths from S to T, filtering the paths based on the length constraint in Equation 4.1, and then selecting the subset P_{KMDNSP} using one or more of the dissimilarity functions in Table 2.2 is termed as the Exact process. For small networks, Exact process works can provide ideal results. However, on a real-world road network, finding all near-shortest paths is computationally infeasible (Häcker et al. 2021). Thus, heuristic approaches that can generate multiple near-shortest paths within the specified GAP (%) while maintaining path diversity are used.

4.3 MultiPath Island-Based Genetic Algorithm (MIBGA)

As described in Chapter 2.6.3, multiple variants of Parallel Genetic Algorithms (PGAs) have been proposed in the literature, including the Global Parallelisation (Master–Slave), Island, Cellular, and Hybrid models. Each of these architectures offers distinct benefits and trade-offs in terms of convergence, population diversity, scalability, and computational overhead. Given the objective of the KMDNSP problem, to identify a set of K paths with

maximum structural dissimilarity, the ability to simultaneously optimise multiple candidate paths while preserving diversity becomes crucial. Since K is typically small (often fewer than 10), this naturally aligns with the structure of the Island model (described in Chapter 2.6.3), where a limited number of islands (subpopulations) evolve independently, with occasional migration (exchange of solutions). This configuration provides a strong balance between exploration (diversity across islands) and exploitation (optimisation within each island).

In contrast, the Cellular model employs a grid-based structure in which each solution interacts only with its local neighbours. While this promotes diversity, it reduces the convergence capability of the algorithm and is therefore less efficient for problems like KMDNSP, where convergence is equally important. Additionally, since all experiments are performed on a single computational node to ensure a fair comparison with GAs proposed by (Ahn & Ramakrishna 2002) and (Qiongbing & Lixin 2016), the Global Parallelisation (Master–Slave) model was not suitable, as its primary advantage lies in distributed fitness evaluation (Cantú-Paz et al. 1998).

4.3.1 Genetic representation of solution paths

We initialised our proposed GA randomly without imposing any constraints. The genetic representation of a solution in the POS is a sequence of node IDs that starts from the source node (S) and follows the IDs of other nodes in path order until it terminates at the target node (T). Each node in the graph is represented by a positive integer, which can be mapped onto place names or other identifiers using a simple lookup table. The length of each solution path is variable but cannot exceed the total number of nodes in the network. Although this constraint is unlikely to be tested in any large real-world road network, it could be a realistic scenario for small networks. To keep the record of all edges in the network, we implemented a topological database.

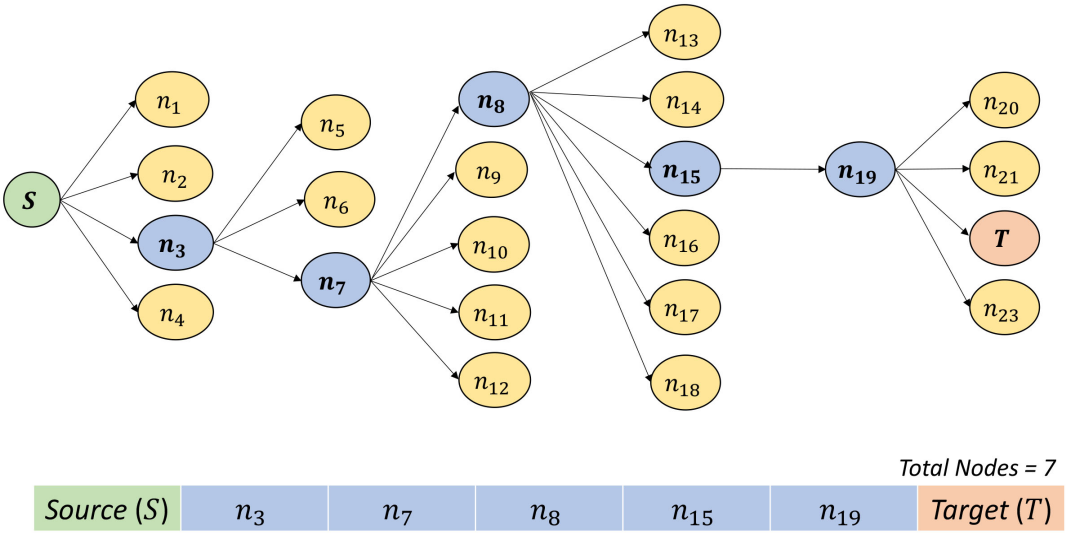

Figure 4.2: An example of a candidate path initialised using random encoding (without constraints).

Figure 4.2 shows a sample path from the source node (S) to the target node (T). The total number of nodes in the path is seven. The first node in the path is the source node (S). N is appended to all node integers for clarity. N3 is chosen first and then the process of random selection of connected nodes is followed until we reach the target node (T).

4.3.2 Mending function for solution paths

A significant issue with the random encoding approach (described in Section 4.3.1), is the possibility of loop formation. To ensure that the path generated satisfies the constraints mentioned in Chapter 3, we employ a mending function similar to the one discussed in (Ahn & Ramakrishna 2002). We illustrate our implementation of the mending function in Figure 4.3. This external function removes loops from a path after the path has been initialised. In the example given in Figure 4.3, a route ($S \rightarrow n_4 \rightarrow n_6 \rightarrow n_8 \rightarrow n_{13} \rightarrow n_{15} \rightarrow n_8 \rightarrow n_{17} \rightarrow n_{19} \rightarrow T$) is initialised. The constraint in Equation 2.4 (described in Chapter 2, Section 2.2) is not satisfied due to the loop ($n_8 \rightarrow n_{13} \rightarrow n_{15} \rightarrow n_8$) in the path, which needs to be repaired or mended. The proposed mending function takes the invalid path as input and searches for a duplicate node in the path. Once a duplicate node has been found, it removes all the nodes encountered between the two duplicate nodes with one of the duplicate nodes. This process is repeated until all nodes in the path are unique. It should be noted that the constraint given in Equation 2.4 is applied to paths only after the initialisation process is complete.

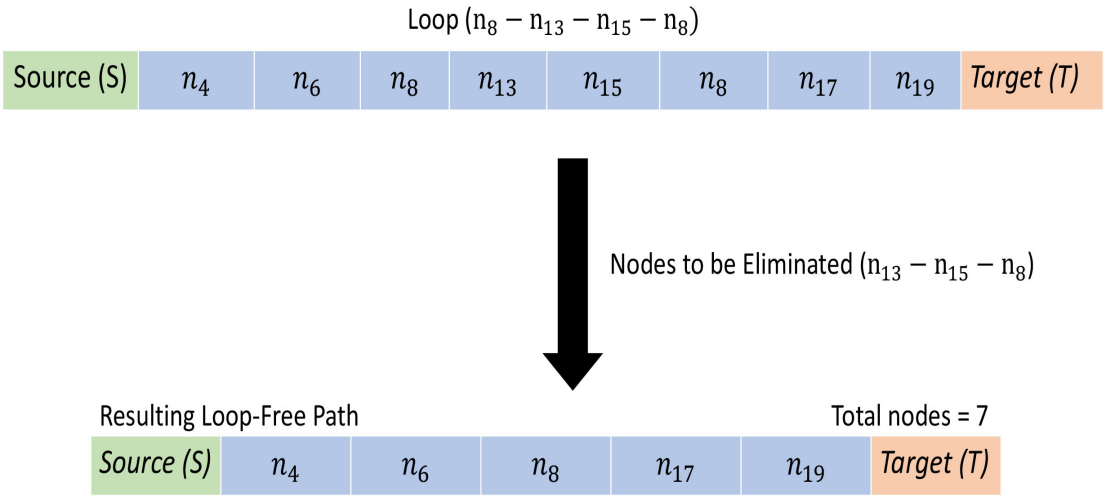

Figure 4.3: An example of a path being repaired using the mending function.

4.3.3 Fitness function

The extent of success of a GA in generating high quality individuals is directly related to the fitness of the solutions produced by it. Then, these solutions are preserved using the principle of elitism, depending on their fitness values, to ensure that the quality in successive generations will be better (Lambora et al. 2019). For the VR problem, the aim is to find multiple routes with the smallest overall distance to travel from a source node

(S) to a target node (T) over a valid path p. In Section 4.3.1, we discussed the length of a path p in terms of the number of nodes in p, whereas the fitness evaluation considers a higher-level metric, such as the geographical distance represented by the path p. The fitness function is defined in Equation 4.5, where f_p denotes the fitness and $|p|$ represents the length of path p, calculated using Equation 2.5.

$$f_p = \frac{1}{|p|} \tag{4.5}$$

4.3.4 Island formation strategy

Premature convergence to a local minima is a problem that many GAs suffer from (Dominico & Parpinelli 2021). A common strategy to mitigate this issue is to maintain a diverse (less overlapping) set of solutions throughout the algorithm, employing techniques such as fitness sharing to preserve diversity, as discussed by Goldberg et al. (1987). IB-GAs utilise multiple islands evolving independently within their search space to achieve this purpose. These islands help generate multiple local minima and maintain diversity in the solution set throughout the algorithm. Inspired by this idea of living in groups and promoting elitism, the concept of survival of the fittest by providing more opportunities for the strongest candidates to prevail and propagate their attributes. We try to replicate this process by having candidate solutions (possible solution paths) in groups of a few strong candidates and other general candidate solutions. The pseudocode to perform the island formation process is given in Algorithm 1. This process allows us to have a variable number of islands with variable sizes, within the search space. It should be noted that the maximum group size specified as input is not fixed and will change as the algorithm is executed over successive generations. The minimum group size parameter is used to ensure that each group has at least one individual to ensure offspring production. The Island Formation Strategy is outlined as follows:

1. The initial population of solutions is first sorted based on the fitness metric explained in Section 4.3.3 (see line 3 in Algorithm 1).

2. Once sorted, the population is divided into two global groups: the Superior Population and Centralised Population, using a parameter called the Selection Threshold. This refers to the portion of current best solutions that we shall consider as Superior solutions. For instance, if the Selection Threshold is 0.25 or 25% this means that the top 25% of solutions from a generation will be Superior Population while the Centralised Population will consist the entire population of the current generation (see line 4 and 5 in Algorithm 1). It is also important to note that only top-performing solutions are allowed to appear in both groups, which enhances the algorithms convergence capabilities and overall performance.

3. Following that, a random number within the range of minimum and maximum island size (provided manually as input) is chosen to preset the size of the current island (see line 11 in Algorithm 1).

Algorithm 1 Island Formation Strategy

1: **Input:** Initial population (Pop), Selection threshold (E), Minimum island size (S), Maximum island size (M).
2: **Output:** Collection of Islands (G), consisting groups of population.

3: Sort population Pop in descending order of Fitness;
4: Superior Population: s = Top $E\%$ of Pop;
5: Centralised Population: $r = Pop$;
6: Let Island Collection: $G = \{\}$;
7: **while** $|s| \neq 0$ **and** $|r| \neq 0$ **do**
8: Let Current Island: $c = \{\}$;
9: Let Parent from Superior Population: $P_{sp} = \{\}$;
10: Let Parent from Centralised Population: $P_{cp} = \{\}$;
11: $|c| \leftarrow$ random number in the range $[S, M]$;
12: $|P_{sp}| \leftarrow E\%$ of $|c|$;
13: $|P_{cp}| \leftarrow |c| - |P_{sp}|$;
14: **if** $|P_{sp}| < |s|$ **and** $|P_{cp}| < |r|$ **then**
15: Randomly extract $|P_{sp}|$ elements from s and add to P_{sp}: $P_{sp} \leftarrow P_{sp} \cup \{x_1, \ldots, x_{|P_{sp}|}\}$;
16: Randomly extract $|P_{cp}|$ elements from r and add to P_{cp}: $P_{cp} \leftarrow P_{cp} \cup \{x_1, \ldots, x_{|P_{cp}|}\}$;
17: Add P_{sp} and P_{cp} to c: $c \leftarrow c \cup \{P_{sp}, P_{cp}\}$;
18: Add c to G: $G \leftarrow G \cup \{c\}$;
19: **else if** $|s| > 1$ **and** $|r| > 1$ **then**
20: Extract the remaining elements from s and add to P_{sp}: $P_{sp} \leftarrow P_{sp} \cup \{x_1, \ldots, x_{|P_{sp}|}\}$;
21: Extract the remaining elements from r and add to P_{cp}: $P_{cp} \leftarrow P_{cp} \cup \{x_1, \ldots, x_{|P_{cp}|}\}$;
22: Add P_{sp} and P_{cp} to c: $c \leftarrow c \cup \{P_{sp}, P_{cp}\}$;
23: Add c to G: $G \leftarrow G \cup \{c\}$;
24: **else if** $|s| > 1$ **and** $|r| < 1$ **then**
25: Randomly selected an island c from G;
26: Extract remaining $|s|$ elements from s and update P_{sp} of the selected c: $P_{sp} \leftarrow P_{sp} \cup \{x_1, \ldots, x_{|s|}\}$;
27: **else** $\{|s| < 1$ **and** $|r| > 1\}$
28: Randomly selected an island c from G;
29: Extract remaining $|r|$ elements from r and update the P_{cp} of the selected c: $P_{cp} \leftarrow P_{cp} \cup \{x_1, \ldots, x_{|r|}\}$;
30: **end if**
31: **end while**
32: **return** G

4. Using the Selection Threshold and current island size (defined in the previous step), we define the size for two groups: P_{sp}*: Parents from Superior Population* and P_{cp}*: Parents from Centralised Population* (see line 12 and 13 in Algorithm 1).

5. Once the P_{sp} and P_{cp} group sizes are established, the specified number of solutions are drawn at random from the Superior Population and Centralised Population, respectively (see line 15 and 16 in Algorithm 1). Next, the P_{sp} group comprising solutions from the Superior Population, and the P_{cp} group comprising solutions from the Centralised Population, are added to the island (see line 17). It is noteworthy that the island size is the total number of solutions within the island, including the P_{sp} and P_{cp} groups.

6. In case, if it is not possible, to extract the specified number of solutions from the Superior and/or Centralised population, the process follows one of the following

steps based on the situation:

- If both Superior Population and Centralised Population have one or more solutions, a new island is created with groups P_{sp} (remaining Superior solutions) and P_{cp} (remaining centralised solutions), and the process is terminated (see line 19, 20, 21 in Algorithm 1).
- If either one of the Superior Population and Centralised Population have no solutions, the remaining solutions of the other are adjusted in any one group of its respective type, and the process is terminated (see line 25-30 in Algorithm 1).
- If both Superior Population and Centralised Population have no solutions, the process is terminated.

4.3.5 Migration strategy

In Island-Based Genetic Algorithms (IBGAs) diversity is introduced and conserved by dividing the population between defined islands as described in Section 4.3.4 and by implementing a migration strategy, as illustrated in Algorithm 2. Migration refers to the process of exchanging solutions (generally a copy) between sub-populations on different islands at a certain time. The selection of islands to exchange solutions depends on the topology of islands. This topology also guides the evaluation process across all islands, promoting global convergence towards the optimal solution, as discussed by Harada & Alba (2020). Some of the commonly used topologies include ring, mesh, random, and others. To implement a migration strategy, it is important to define migration frequency and a replacement strategy defining the rate, quantity, and type of solutions that will participate in the process of migration.

Algorithm 2 Migration Strategy

1: **Input:** Collection of Islands (G), where each island consists of groups P_{sp} (Parents from the Superior Population) and P_{cp} (Parents from the Centralised Population).
2: **Output:** Collection of Islands (G'), where each island contains groups P'_{sp} (Updated Parents from the Superior Population) and P_{cp} (Parents from the Centralised Population).

3: Let New Island Collection: $G' = \{\}$;
4: Number of Islands: $N_{islands} = |G|$;
5: Generate a random permutation of indices: $I \leftarrow [0, 1, ..., N_{islands} - 1]$;
6: **for** $i = 0$ to $N_{islands} - 1$ **do**
7: Current Island: $c \leftarrow G[i]$;
8: **Note:** c contains groups P_{sp} and P_{cp}: $c = \{P_{sp} : G[i][P_{sp}], P_{cp} : G[i][P_{cp}]\}$;
9: Update the P_{sp} group of c: $c[P_{sp}] \leftarrow G[I[i]][P_{sp}]$;
10: Add c to G': $G' \leftarrow G' \cup \{c\}$;
11: **end for**
12: **return** G'

For the islands in our study, we utilised a randomly generated topology described in Section 4.3.4. A migration strategy that swaps the entire P_{sp} group of a island with the P_{sp} group of a different randomly selected island from the available islands in the search

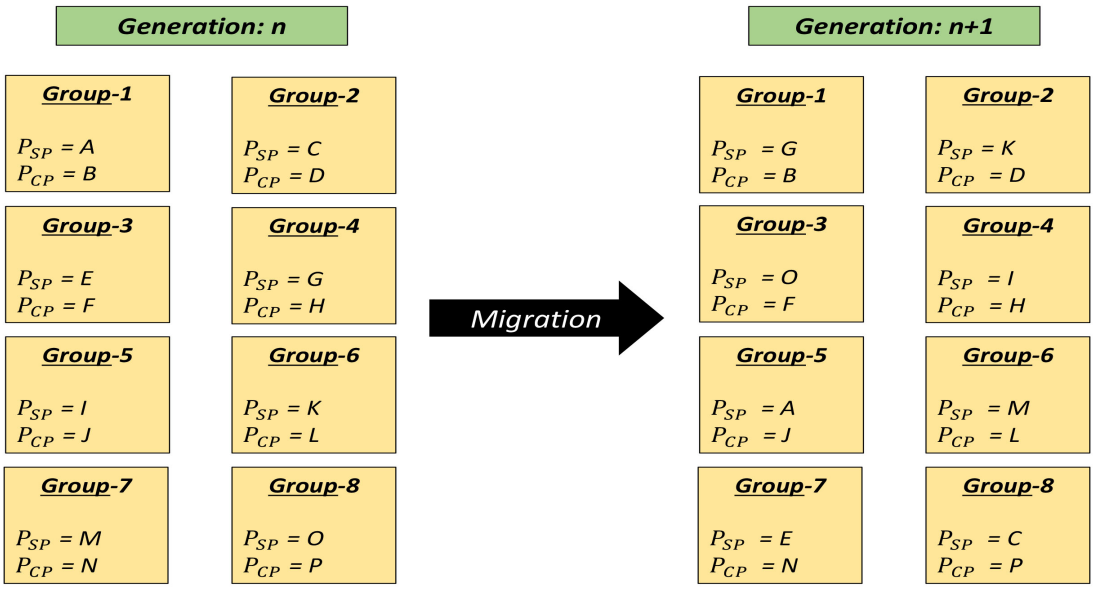

Figure 4.4: An example of the migration process between two generations

space at each generation is employed (see lines 6–10 in Algorithm 2). Figure 4.4 shows an example of this migration approach. The group of P_{sp} solutions and P_{cp} solutions are represented using the alphabetical characters. The topology of solution that was formed during generation (n) is shown on the left and the new topology formed to produce offspring for generation $(n+1)$ is shown on the right. The only difference between both structure is the change of P_{sp} groups in each island.

4.3.6 Offspring generation process

New path solutions for evaluation are generated using a group-wise offspring generation process that restricts the process to members of the same island within the solution space. Through this operation, offspring produced by each island are stored separately. These will be used in the first phase of the bipartite selection process described in Section 4.3.9. The pseudocode to perform the offspring generation process is given in Algorithm 3. Each solution from the centralised group in all islands participate in the offspring generation process (see line 8 in Algorithm 3). Therefore, each island of the current generation will produce $2X$ offspring, where X refers to the number of solutions in the respective P_{cp} group. However, the selection of the parent from the P_{sp} group will be performed by using a weighted selection process (see line 9 in Algorithm 3). The weight to each parent of the P_{sp} group is assigned using Equation 4.6, where W_{ga} represents the weight assigned to Parent (A) of P_{sp} group (g), f_{ga} represents the fitness of Parent (A) of the P_{sp} group (g), and T_g represents the cumulative sum of fitness of all parents of the P_{sp} group (g). In the offspring generation process, a new path can be generated in two ways. The most prominent means is using only crossover operation (see line 16 in Algorithm 3). The second approach is to use the mutation operation (see line 13 in Algorithm 3) on the top of the crossover operator. It is noteworthy that mutation is applied to only a very small number of paths.

Algorithm 3 Offspring Generation Process using the LFPC operator

Input: Collection of islands (G'), where each island contains groups P'_{sp} (Parents from the Superior Population, after Migration) and P_{cp} (Parents from the Centralised Population), and the Mutation probability (M_p).
Output: Collection of islands (C), where each island contains an offspring population generated using the LFPC operator.

Initialise empty Island collection: $C \leftarrow \{\}$
for each island c in G' **do**
  Retrieve P'_{sp} group (migrated population) from Island c: $m \leftarrow c[P'_{sp}]$
  Retrieve P_{cp} group (centralised population) from Island c: $f \leftarrow c[P_{cp}]$
  Initialise empty group $o \leftarrow [\ ]$ to store offspring generated by the solutions from Island c
  for j (solution path) in f (centralised population) **do**
    Select a solution s from m using weighted selection
    $r \leftarrow$ a random integer in the range $[1, 100]$
    if $r \leq M_p$ **then**
      //Generate new solutions using LFPC ((Mutation Integrated with Crossover))//
      $offspring_1, offspring_2 \leftarrow \text{LFPC}(s, j)$
    else
      //Generate new solutions using only LFPC (with Crossover)//
      $offspring_1$, $offspring_2 \leftarrow \text{LFPC}(s, j)$
    end if
    Append $offspring_1$ and $offspring_2$ to o
  end for
  Add group o to collection C: $C \leftarrow C \cup \{o\}$
end for
return C

$$W_{ga} = f_{ga}/T_g \tag{4.6}$$

4.3.7 Loop-Free Path-Composer: With crossover

Here, we propose a method of crossover for variable-length chromosomes that is independent of common nodes or node positions in the parent chromosome. Loop-Free Path-Composer (LFPC) selects a random node in both parent chromosomes to divide the paths into two parts. Then, the first part (from the source to node α) is combined with the second part (from node β to the target node) using a third partial route between them. Here, α and β represent randomly selected nodes in Parent (A) and Parent (B), respectively. While this process generates a feasible path, it may contain loops. To ensure that the generated path is loop-free and adheres to the conditions explained in Section 2.2 of Chapter 2, we use the mending function described in Section 4.3.2. Figure 4.5 provides an example of LFPC operation using crossover. In this example, R(A) and R(B) correspond to the randomly selected node α from Parent (A) and β from Parent (B), respectively.

4.3.8 Loop-Free Path-Composer: Mutation integrated with crossover

The mutation operator plays an important role in GAs by enabling exploration of the solution space through the introduction of small changes in the chromosome to promote

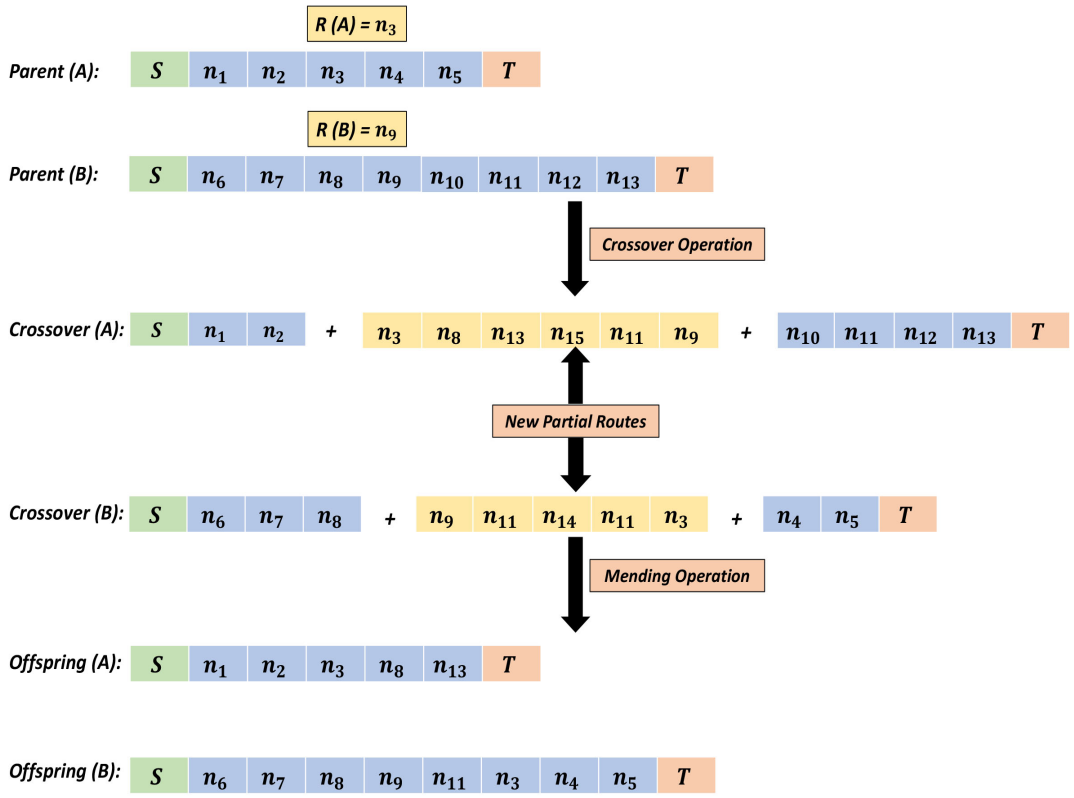

Figure 4.5: An example of LFPC operation using only Crossover.

diversity (Sharma et al. 2025b). In case of routing, similar to the crossover operator, we need to ensure that the path generated with mutation will be valid. To achieve this, we applied the mutation operation alongside the crossover operator described in Section 4.3.7. The process of generating new paths using the LFPC operator with mutation is outlined as follows:

- Select two parent chromosomes, one from the P_{sp} group referred to as Parent (A) and the other from the P_{cp} group referred to as Parent (B). It is noteworthy that the mutation operation is performed only on the chromosome selected from the P_{sp} group.

- Two random nodes, R(A) and R(B) are selected from Parent (A) and Parent (B), respectively.

- Using the network database, a random node (R(C)) is then chosen from the set of adjacent nodes connected to the node preceding R(A).

- The selected node R(C) replaces R(A) as the random node in Parent (A) and then the crossover operation is performed as as described in Figure 4.5.

- Example:- Consider Parent(A) in Figure 4.5 belongs to a P_{sp} group and n_3 is the randomly selected node. Node n_2 is preceding n_3, and assume n_2 has direct edges with nodes n_1, n_3, n_{10} and n_{11}. From these connected nodes, one is selected randomly

(n_{11}) and replaces n_3 as R(A) in Parent (A). Using n_{11} Parent(A) is divided into $S \rightarrow n_1 \rightarrow n_2 \rightarrow n_{11}$ (p_{A1}) and $n_4 \rightarrow n_5 \rightarrow T$ (p_{A2}). Similarly, using n_9 Parent(B) is divided into $S \rightarrow n_6 \rightarrow n_7 \rightarrow n_8 \rightarrow n_9$ (p_{B1}) and $n_{10} \rightarrow n_{11} \rightarrow n_{12} \rightarrow n_{13} \rightarrow T$ (p_{B2}). Finally, two new paths will be generated, one by combining p_{A1} with p_{B2} (partial route between n_{11} and n_9) and second by combining p_{B1} with p_{A2} (partial route between n_9 and n_{11}).

4.3.9 AvgIslandFit: A multi-step selection process

MIBGA approach utilises two selection strategies for different purposes. First, a weighted selection process, as defined by Equation 4.6, is employed to select parents from the P_{sp} group of each island in the offspring generation process. Second, a multi-step selection process referred to as AvgIslandFit is used to determine the solutions that will be carried forward to the next generation. Steps to perform the selection process using AvgIslandFit are summarised as follows:

1. Perform selection on the child population of each island by removing individuals with fitness values below their respective island's average fitness.

2. Divide each island into two groups, P_{sp} and P_{cp}, based on the selection threshold parameter explained in Section 4.3.4.

3. Combine the two groups of the parent population (P_{sp} and P_{cp}) with the two groups of the child population (P_{sp} and P_{cp}) for each island.

4. Perform selection on the combined P_{sp} and P_{cp} groups of each island by removing individuals with fitness values below their respective group's average fitness.

5. Remove a random number of individuals with the worst fitness values from the P_{cp} set of each island, based on the size of the group. For instance, if size of Centralised Population is less than 5 than no solutions gets removed. If the size is between 5 and 12 than either one or two solutions with the worst fitness value(s) are removed. If the size is between 5 and 17 than either one, two or three solutions with the worst fitness values are removed, and so on.

This combination of selection strategies enhances MIBGAs performance by optimising parent selection, maintaining selection pressure, and dynamically adjusting the number of evaluations per generation based on changes in each subgroups average path length.

4.3.10 Illustration of the overall MIBGA approach

The input is the topological database of the graph network from NetworkX (Hagberg et al. 2008) generated using the process detailed in Chapter 3. Next, the user provides input parameters, including the Source node (S) and Target node (T) to define the path endpoints. The user also specifies the Number of Paths (K) to consider and the Length threshold (ϵ) to limit the acceptable path lengths. Following this, initial set of paths referred as Initial population are created and evaluated using random initialisation described in Section 4.3.1

and the fitness function given by Equation 4.5, respectively. The process referred as Island Formation Strategy (Section 4.3.4) is used to create islands containing two groups each. The computation time (C_T) of 120 seconds for each simulation is manually set to terminate the algorithm if the required number of paths (K) are not found within the specified time. After the initial population, each iteration of the process starts from the method given in Section 4.3.5 (Migration), followed by Section 4.3.6 (Offspring Generation), Section 4.3.3 (Fitness Evaluation), and Section 4.3.9 (AvgIslandFit). At the completion of this iteration this accounts for one generation of the proposed MIBGA. Upon termination, a set of optimal paths is returned by the algorithm. The novel aspects of our study are explicitly highlighted using green blocks in the flowchart presented in Figure 4.6, ensuring a clear distinction of the innovative components within MIBGA.

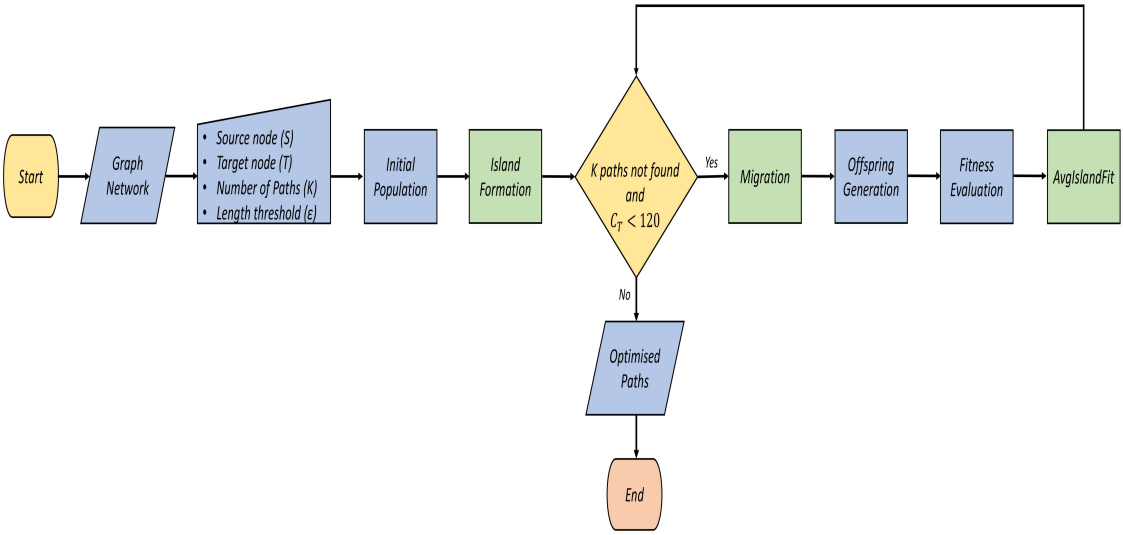

Figure 4.6: Flowchart illustrating the overall process of MIBGA.

4.4 Algorithm description and configuration

This section outlines the algorithms utilised in the work, detailing their corresponding parameters, the rationale for parameter selection, and the tuning methodology applied. We use three different real-world road network topologies of varying sizes and a self-coded version of the Dijkstra's algorithm to generate the single shortest path for each test case. User defined parameters, the number of near shortest paths (K) to generate ranges from 2 to 5, with length thresholds (ϵ) set at 0.01, 0.05, 0.10, 0.15, and 0.20. For each network, all 20 possible combinations of K and ϵ are evaluated, with 100 independent simulations per combination, resulting in a total of $2,000$ simulations per network. For each simulation, two distinct nodes are randomly selected as the source (S) and target (T) nodes, ensuring S must not be equal to T and both S and T are integers from the networks node set.

The widely used GAs proposed by Ahn & Ramakrishna (2002) and Qiongbing & Lixin (2016) are included for comparison. The components of the GAs in (Ahn & Ramakrishna 2002, Qiongbing & Lixin 2016) and our MultiPath Island-Based Genetic Algorithm

(MIBGA) are summarised in Table 4.1. All three algorithms used variable length chromosomes to represent valid paths. These chromosomes were initialised using the random encoding method described in Section 4.3.1. In all three algorithms, crossover serves as the primary genetic operator for generating new paths for evaluation, with each algorithm using a different crossover method. The GA introduced by Ahn & Ramakrishna (2002) utilised the node based crossover method whereas the GA from Qiongbing & Lixin (2016) employed the same adjacency crossover method, detailed in Chapter 2. In contrast, MIBGA employed the LFPC operator to generate new paths, as described in Section 4.3.7. Additionally, the GAs from Ahn & Ramakrishna (2002) and Qiongbing & Lixin (2016) applied path mutation as described in Chapter 2, Section 2.7.5, while MIBGA employed the LFPC operator (used as both a mutation and crossover operator), as detailed in Section 4.3.8. The GA by Ahn & Ramakrishna (2002) employed pairwise Tournament Selection (without replacement) (Eiben & Smith 2015), while the GA from Qiongbing & Lixin (2016) utilised Roulette Wheel Selection (Eiben & Smith 2015). In contrast, MIBGA applied the novel selection operator AvgIslandFit, which is introduced and described in Section 4.3.9. It is important to note that although each algorithm utilised a mutation operator, the bias introduced by it can be considered negligible (Ahn & Ramakrishna 2002).

GAs/Authors	Chromosome Length	Encoding Method	Crossover Operator	Mutation Operator	Selection Operator
Ahn & Ramakrishna (2002)	Variable	Random	Node Based	Path Mutation	Tournament
Qiongbing & Lixin (2016)	Variable	Random	Same Adjacency	Path Mutation	Roulette Wheel
MIBGA	Variable	Random	LFPC	LFPC	AvgIslandFit

Table 4.1: The components of the GAs proposed by Ahn & Ramakrishna (2002), Qiongbing & Lixin (2016), and MultiPath Island-Based Genetic Algorithm (MIBGA).

For a fair comparison, all algorithms are initialised with the same initial population of variable length chromosomes, generated using the random initialisation process described in Section 4.3.1. A population size of 250 is used, as using a population size equal to the number of nodes, as proposed by Ahn & Ramakrishna (2002), becomes impractical for large networks, such as those used in this work. The crossover probability is set to 1.00, as the best results are achieved when all paths undergo crossover, minimising duplicates within the population. The mutation probability for each algorithm is set to 0.05, a commonly used value (Ahn & Ramakrishna 2002). Given the nature of the problem at hand, termination conditions used are, (i) If K paths satisfying the length threshold (ϵ) have been found. If so terminate, if not continue, and (ii) If computation time has exceeded 120 seconds. If so terminate, if not continue.

The second termination condition is used to ensure a practical time frame and stay within the computational budget. Upon completion (excluding timeouts), each algorithm returns a set P_A of K or more paths. From this set, the K-Most Diverse Near-Shortest Paths (KMDNSP) P_{KMDNSP} is selected using Equation 4.3. A limit of 30 paths for the

set P_A was imposed to constrain the computational budget. If the size of P_A exceeds 30, a random subset of 30 paths was selected. Thus, for all algorithms, the maximum number of path combinations to be evaluated for a single test case using Equation 4.3, will be $\binom{30}{5}$, when K is set to 5.

In addition to the parameters discussed above, MIBGA requires three additional parameters, selection threshold, minimum island size, and maximum island size for the islands. Selection threshold is a parameter that is introduced to enhance the convergence capabilities of MIBGA. Smaller selection threshold tend to reduce diversity within the population, leading to premature convergence, while larger values can reduce the algorithms ability to converge efficiently. Minimum island size and maximum island size are used to maintain an adequate sub-population within each island. We used a combination of Latin Hypercube Sampling (LHS) (Stein 1987) and random search to tune these three parameters. The rationale for using LHS was its ability to efficiently explore a large and multidimensional parameter space with relatively few samples. Given that our experimental setup required testing the algorithm on three networks, each with 20 combinations of ϵ and K, 100 independent runs per combination (each with a minimum of 250 evaluations), and varying the selection threshold (0.10–0.20), minimum island size (4–12), and maximum island size (12–20), LHS provided a stratified and computationally efficient means to ensure representative coverage of the search space. After tuning, the selection threshold, minimum island size, and maximum island size were set to 0.10, 5, and 15, respectively. These parameters are employed in Algorithm 1 (see page 61).

4.5 Experimental results

In this chapter, we compare the performance of our MultiPath Island-Based Genetic Algorithm (MIBGA) on the K-Most Diverse Near-Shortest Paths (KMDNSP) problem with the genetic algorithms proposed by Ahn & Ramakrishna (2002) and Qiongbing & Lixin (2016). As mentioned and detailed in Chapter 3, we use three road networks of varying sizes and topologies in order to give a suitable range for overall road network size. Kansas is the largest network with $2,224$ unique nodes and $4,257$ edges while Arizona is the smallest with only 834 nodes and $1,547$ edges. The network of Washington had $1,564$ nodes, and $2,784$ edges. A data repository providing all three graphical representations in the edge list format, is included in the GitHub profile: MIBGA. The code to convert these edge lists into NetworkX (Hagberg et al. 2008) format graphs is also provided.

User defined parameters, the number of near shortest paths (K) to generate ranges from 2 to 5, with length threshold (ϵ) set at 0.01, 0.05, 0.10, 0.15, and 0.20. For each network, all 20 possible combinations of K and ϵ are evaluated, with 100 independent simulations per combination, resulting in a total of $2,000$ simulations per network. We assess the average runtime of each algorithm with respect to K and ϵ to evaluate their performance. The analysis in this research is conducted using a consistent device featuring a 12-core CPU, an 18-core GPU, and a 16-core Neural Engine. A 120 seconds timeout was enforced, with $Div(P)$ set to zero if exceeded.

4.5.1 Comparative study on the proportion of non-timeout instances

We begin our discussion by comparing the performance of MIBGA against the GA by Ahn & Ramakrishna (2002) and Qiongbing & Lixin (2016) in terms of identifying the K near shortest paths for a given pair of source and target nodes on the three test networks. Figure 4.7 display the percentage (rounded) of cases where an algorithm successfully identified the K Near-Shortest Paths within the specified time. Each cell indicates the percentage of cases where an algorithm successfully identified K Near-Shortest Paths within 120 seconds. Results are aggregated across three networks, with each percentage rounded from 300 test cases (100 per network). For instance, when K is set to 5 and ϵ is set to 0.01, MIBGA successfully identified 5 or more paths within the specified length in 89% of the test cases, compared to 81% and 75% for the GAs from Ahn & Ramakrishna (2002) and Qiongbing & Lixin (2016), respectively.

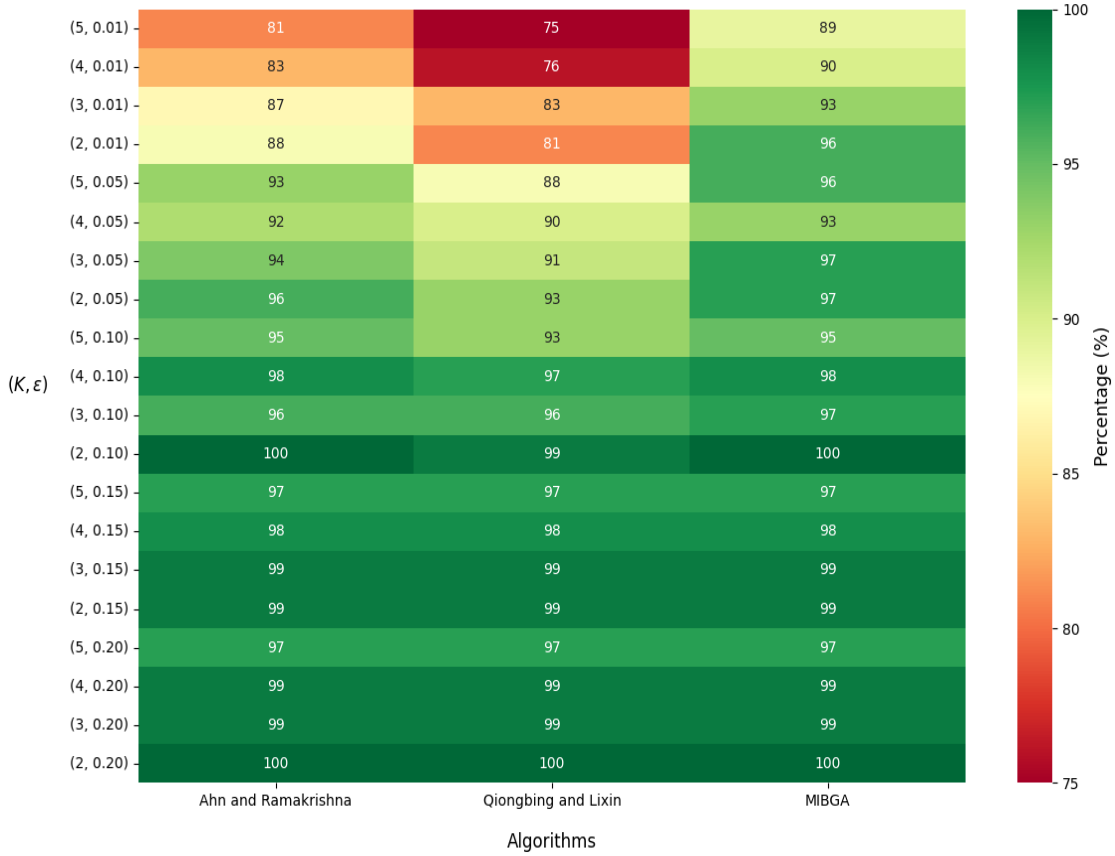

Figure 4.7: Success percentage of each algorithm in identifying the K Near-Shortest Paths. Algorithms, including Ahn & Ramakrishna (2002), Qiongbing & Lixin (2016), and MIBGA are shown on the x-axis. All possible combinations of (K, ϵ) are shown on the y-axis. The values in each cell represent the percentage of cases where an algorithm identified K Near-Shortest Paths within 120 seconds. Results are aggregated across three networks, with each cell reflecting the rounded percentage based on 300 test cases (100 per network).

The success percentage of each algorithm in identifying the K Near-Shortest Paths is shown in Figure 4.7. The x-axis represents the algorithms, including Ahn & Ramakrishna (2002), Qiongbing & Lixin (2016), and MIBGA, while the y-axis displays all possible (K, ϵ) combinations. MIBGA consistently achieves high success rates (above 90%) across all

(K, ϵ) combinations, except for (5, 0.01). In contrast, the GA by Ahn & Ramakrishna (2002) starts with lower success rates, such as 81% at (5, 0.01), but improves significantly as ϵ increases. The GA by Qiongbing & Lixin (2016) performs the worst at smaller ϵ values, with success rates dropping to 75% and 76% at (5, 0.01) and (4, 0.01), respectively. With the relaxation of length constraints (ϵ increasing), success rates improve across all algorithms. For $\epsilon \geq 0.10$, the performance stabilises, with all three algorithms achieving at least 95% success. Significant differences were observed only at lower ϵ values, where MIBGA outperformed both GAs in (Ahn & Ramakrishna 2002, Qiongbing & Lixin 2016).

4.5.2 Comparative study on the generation of most diverse path sets

We report the results obtained when we compare the set of paths generated by MIBGA with those generated by the GAs from Ahn & Ramakrishna (2002) and Qiongbing & Lixin (2016) based on the diversity metric Div(P), as shown in Equation 4.3. Div(P) measures the diversity of a set P as the minimum pairwise dissimilarity among the contained paths (Häcker et al. 2021). The set with the maximum value of Div(P) is selected as the set of K-Most Diverse Near-Shortest Paths (KMDNSP) P_{KMDNSP}.

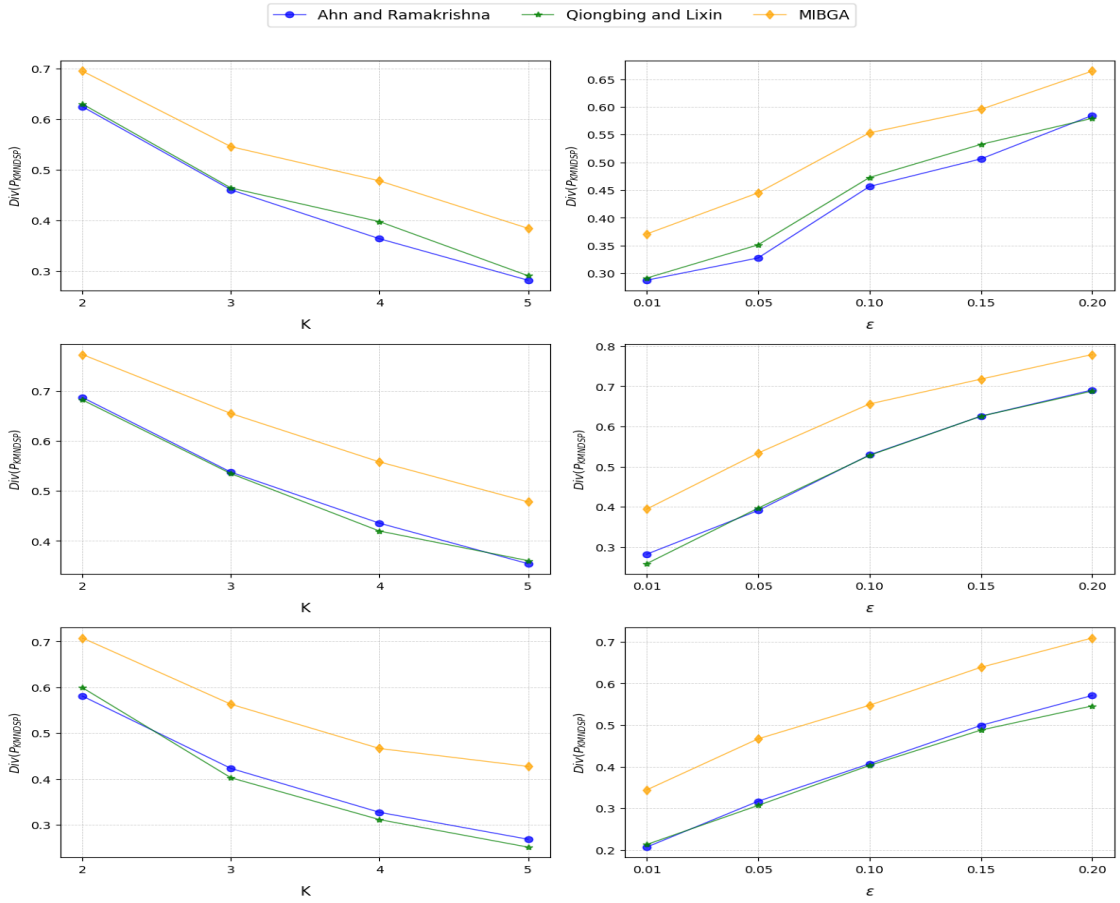

Figure 4.8: The variation in average Div(P_{KMDNSP}) by MIBGA and the GAs detailed in (Ahn & Ramakrishna 2002, Qiongbing & Lixin 2016) with respect to the user defined parameters, the number of near shortest paths to generate (K) and the length threshold (ϵ) is shown across the road networks of Arizona (834 nodes and 1,547 edges), Washington (1,564 nodes and 2,784 edges), and Kansas (2,224 nodes and 4,257 edges).

Algorithm	ϵ	Mean $Div(P_{KMDNSP})$ ± Standard Deviation			
		$K = 2$	$K = 3$	$K = 4$	$K = 5$
Arizona (834 **nodes and** 1,547 **edges)**					
Ahn & Ramakrishna (2002)	0.01	0.47 ± 0.35	0.28 ± 0.29	0.26 ± 0.28	0.13 ± 0.19
Qiongbing & Lixin (2016)		0.46 ± 0.36	0.26 ± 0.29	0.30 ± 0.29	0.14 ± 0.19
MIBGA		**0.55 ± 0.33**	**0.37 ± 0.30**	**0.34 ± 0.29**	**0.22 ± 0.23**
Ahn & Ramakrishna (2002)	0.05	0.53 ± 0.31	0.35 ± 0.33	0.23 ± 0.26	0.20 ± 0.22
Qiongbing & Lixin (2016)		0.54 ± 0.33	0.39 ± 0.32	0.27 ± 0.26	0.20 ± 0.23
MIBGA		**0.63 ± 0.31**	**0.48 ± 0.30**	**0.35 ± 0.27**	**0.32 ± 0.25**
Ahn & Ramakrishna (2002)	0.10	0.67 ± 0.30	0.50 ± 0.33	0.39 ± 0.29	0.27 ± 0.25
Qiongbing & Lixin (2016)		0.69 ± 0.26	0.50 ± 0.32	0.42 ± 0.28	0.28 ± 0.24
MIBGA		**0.73 ± 0.24**	**0.58 ± 0.29**	**0.50 ± 0.28**	**0.40 ± 0.27**
Ahn & Ramakrishna (2002)	0.15	0.67 ± 0.31	0.55 ± 0.29	0.42 ± 0.31	0.38 ± 0.28
Qiongbing & Lixin (2016)		0.70 ± 0.27	0.54 ± 0.29	0.48 ± 0.30	0.42 ± 0.28
MIBGA		**0.75 ± 0.25**	**0.61 ± 0.28**	**0.55 ± 0.29**	**0.47 ± 0.28**
Ahn & Ramakrishna (2002)	0.20	0.79 ± 0.23	0.62 ± 0.30	0.51 ± 0.33	0.42 ± 0.30
Qiongbing & Lixin (2016)		0.76 ± 0.26	0.63 ± 0.30	0.52 ± 0.31	0.41 ± 0.29
MIBGA		0.81 ± 0.21	**0.69 ± 0.26**	**0.64 ± 0.27**	**0.52 ± 0.28**
Washington (1,564 **nodes and** 2,784 **edges)**					
Ahn & Ramakrishna (2002)	0.01	0.45 ± 0.34	0.30 ± 0.28	0.22 ± 0.25	0.15 ± 0.21
Qiongbing & Lixin (2016)		0.43 ± 0.37	0.26 ± 0.29	0.19 ± 0.24	0.15 ± 0.20
MIBGA		**0.57 ± 0.31**	**0.42 ± 0.28**	**0.34 ± 0.28**	**0.24 ± 0.24**
Ahn & Ramakrishna (2002)	0.05	0.59 ± 0.33	0.40 ± 0.31	0.32 ± 0.28	0.26 ± 0.27
Qiongbing & Lixin (2016)		0.62 ± 0.34	0.44 ± 0.30	0.27 ± 0.27	0.25 ± 0.26
MIBGA		**0.72 ± 0.27**	**0.57 ± 0.28**	**0.45 ± 0.29**	**0.39 ± 0.29**
Ahn & Ramakrishna (2002)	0.10	0.74 ± 0.28	0.56 ± 0.30	0.43 ± 0.32	0.39 ± 0.30
Qiongbing & Lixin (2016)		0.73 ± 0.29	0.57 ± 0.31	0.44 ± 0.32	0.38 ± 0.31
MIBGA		**0.83 ± 0.22**	**0.67 ± 0.26**	**0.59 ± 0.30**	**0.53 ± 0.31**
Ahn & Ramakrishna (2002)	0.15	0.80 ± 0.25	0.71 ± 0.28	0.57 ± 0.30	0.42 ± 0.33
Qiongbing & Lixin (2016)		0.76 ± 0.28	0.71 ± 0.28	0.57 ± 0.31	0.47 ± 0.30
MIBGA		0.83 ± 0.24	**0.81 ± 0.18**	**0.66 ± 0.26**	**0.57 ± 0.29**
Ahn & Ramakrishna (2002)	0.20	0.86 ± 0.21	0.71 ± 0.28	0.64 ± 0.27	0.55 ± 0.30
Qiongbing & Lixin (2016)		0.87 ± 0.22	0.70 ± 0.28	0.63 ± 0.29	0.56 ± 0.29
MIBGA		**0.91 ± 0.16**	**0.80 ± 0.21**	**0.74 ± 0.20**	**0.66 ± 0.25**
Kansas (2,224 **nodes and** 4,257 **edges)**					
Ahn & Ramakrishna (2002)	0.01	0.30 ± 0.29	0.27 ± 0.25	0.17 ± 0.19	0.10 ± 0.14
Qiongbing & Lixin (2016)		0.31 ± 0.31	0.27 ± 0.25	0.16 ± 0.20	0.12 ± 0.17
MIBGA		**0.48 ± 0.28**	**0.38 ± 0.26**	**0.27 ± 0.23**	**0.25 ± 0.23**
Ahn & Ramakrishna (2002)	0.05	0.48 ± 0.31	0.34 ± 0.26	0.24 ± 0.23	0.21 ± 0.22
Qiongbing & Lixin (2016)		0.53 ± 0.33	0.30 ± 0.28	0.22 ± 0.21	0.18 ± 0.20
MIBGA		**0.65 ± 0.28**	**0.50 ± 0.26**	**0.39 ± 0.25**	**0.34 ± 0.23**
Ahn & Ramakrishna (2002)	0.10	0.65 ± 0.30	0.39 ± 0.28	0.30 ± 0.28	0.29 ± 0.23
Qiongbing & Lixin (2016)		0.64 ± 0.31	0.41 ± 0.29	0.31 ± 0.26	0.25 ± 0.24
MIBGA		0.71 ± 0.25	**0.58 ± 0.27**	**0.49 ± 0.27**	**0.40 ± 0.26**
Ahn & Ramakrishna (2002)	0.15	0.70 ± 0.33	0.51 ± 0.29	0.44 ± 0.25	0.34 ± 0.27
Qiongbing & Lixin (2016)		0.73 ± 0.28	0.49 ± 0.28	0.42 ± 0.25	0.31 ± 0.27
MIBGA		**0.83 ± 0.20**	**0.64 ± 0.27**	**0.54 ± 0.26**	**0.54 ± 0.25**
Ahn & Ramakrishna (2002)	0.20	0.78 ± 0.28	0.61 ± 0.30	0.49 ± 0.29	0.40 ± 0.26
Qiongbing & Lixin (2016)		0.78 ± 0.28	0.55 ± 0.32	0.46 ± 0.30	0.40 ± 0.25
MIBGA		**0.87 ± 0.18**	**0.72 ± 0.25**	**0.65 ± 0.26**	**0.60 ± 0.25**

Table 4.2: The performance of MIBGA and the GAs by Ahn & Ramakrishna (2002) and Qiongbing & Lixin (2016) on the road networks of Arizona, Washington, and Kansas, is presented. The performance is evaluated using the metrics $Div(P)$ and $Dis(p, p')$ detailed in Equation 4.3 and Equation 4.4, respectively. **Mean Div(P) ± Standard Deviation** represents the average $Div(P)$ observed across 100 test cases for each combination of K (the number of near shortest paths to generate) and ϵ (length threshold). A higher value of Mean Div(P) indicates more diverse (less overlapping) solution set, reflecting better performance. Bold numbers indicate better and statistically significant results (p-value < 0.0025).

The dissimilarity between two paths p and p' is evaluated using $Dis(p, p')$ using Equation 4.4. The complete process is detailed in Chapter 2, Section 4.2. Figure 4.8 illustrates the variation in average Div(P_{KMDNSP}) by MIBGA and the GAs by Ahn & Ramakrishna (2002) and Qiongbing & Lixin (2016) with respect to the user defined parameters, the number of near shortest paths to generate (K) and the length threshold (ϵ), across all networks in this study. All algorithms are highlighted in different colours, emphasising the superior performance of MIBGA. They all followed the same trend, and the variations in Figure 4.8 unveil, Div(P_{KMDNSP}) increases with the increase in ϵ and Div(P_{KMDNSP}) decreases with the increase in K.

The experimental results of this study are shown in Table 4.2, average $Div(P_{KMDNSP})$ observed across 100 test cases for each combination of K and ϵ are represented using **Mean** $Div(P_{KMDNSP})$ $\pm$ **Standard Deviation**, reported separately for the road networks of Arizona, Washington, and Kansas. Equations 4.3 and 4.4, respectively, provide the metrics $Div(P)$ and $Dis(p, p')$ used for evaluation. A higher value of Mean $Div(P)$ indicates more diverse (less overlapping) solution set, reflecting better performance. The results shown demonstrate the superiority of MIBGA compared to the GAs by Ahn & Ramakrishna (2002) and Qiongbing & Lixin (2016) in generating a more diverse set of paths. For all 20 combinations of the user defined parameters K and ϵ, on average the set of paths generated by MIBGA captured better $Div(P_{KMDNSP})$, with best performance of 0.91 ± 0.16 observed on the Washington network, where K and ϵ were set to 2 and 0.20, respectively. Better performance and significance are indicated using the bold font in Table 4.2, with only a few statistically insignificant combinations, where K was set to 2.

The Wilcoxon signed-rank test, a non-parametric statistical hypothesis with Bonferroni correction (Souza et al. 2018) was used to assess the statistical differences between the results of MIBGA and the GAs by Ahn & Ramakrishna (2002) and Qiongbing & Lixin (2016). Statistical significance was evaluated with a Bonferroni-corrected p-value of 0.0025, accounting for the 20 different combinations of K and ϵ. The test was performed in pairs, MIBGA vs Ahn & Ramakrishna (2002) and MIBGA vs Qiongbing & Lixin (2016), with the hypothesis that the distribution of differences between the two results is stochastically greater than a distribution symmetric about zero (Virtanen et al. 2020). Using the SciPy library in Python (Virtanen et al. 2020), significance was established only if both comparisons yielded significant results.

$$r = Z/\sqrt{N} \tag{4.7}$$

Furthermore, we measured the effect size between the compared methods to provide a more interpretable and quantitative assessment of the magnitude of the effect (Fritz et al. 2012). Cohen (2013) proposed the method shown in Equation 4.7 to measure the effect size for non-parametric data. In this approach, the effect size (r) is calculated using Equation 4.7, where Z is derived from the statistic returned by the Wilcoxon signed-rank test, and N is the total number of observations in the sample. According to Cohens guidelines, Effect size (r) is considered, Large (when $r \geq 0.50$), Medium (when $r \geq 0.30$), or Small (when $r \geq 0.10$).

Our analysis reveals that for most combinations where $K = 5$ and $\epsilon = 0.01$, the effect size is small across all networks. This suggests that while statistically significant, the practical advantages of the compared methods under these parameters is minimal. This outcome is expected, as a length threshold of $\epsilon = 0.01$ is very small, which restricts the number of admissible paths. Within such a narrow margin, the paths tend to be few and highly similar, leading to small effect sizes. However, as K decreases ($K = 2, 3$) and ϵ increases ($\epsilon = 10, 15, 20$), the effect sizes increases particularly for the Arizona and Washington networks, indicating a more substantial and practically significant difference between the methods. This implies that under relaxed constraints (lower K, higher ϵ), the proposed MIBGA demonstrates a more pronounced advantage in generating diverse and efficient paths. In contrast, for the Kansas network, the effect sizes remain small across most parameter combinations. This is likely due to its relatively uniform and less complex topology, which inherently offers fewer path variations and less opportunity for one method to outperform the other. This highlights the importance of testing on diverse network topologies, as both network structure and parameter settings can significantly influence the algorithms effectiveness. The results of both the significance and effect size testing are available in a Jupyter notebook on our GitHub profile: MIBGA.

4.5.3 Comparative study on execution times of the algorithms

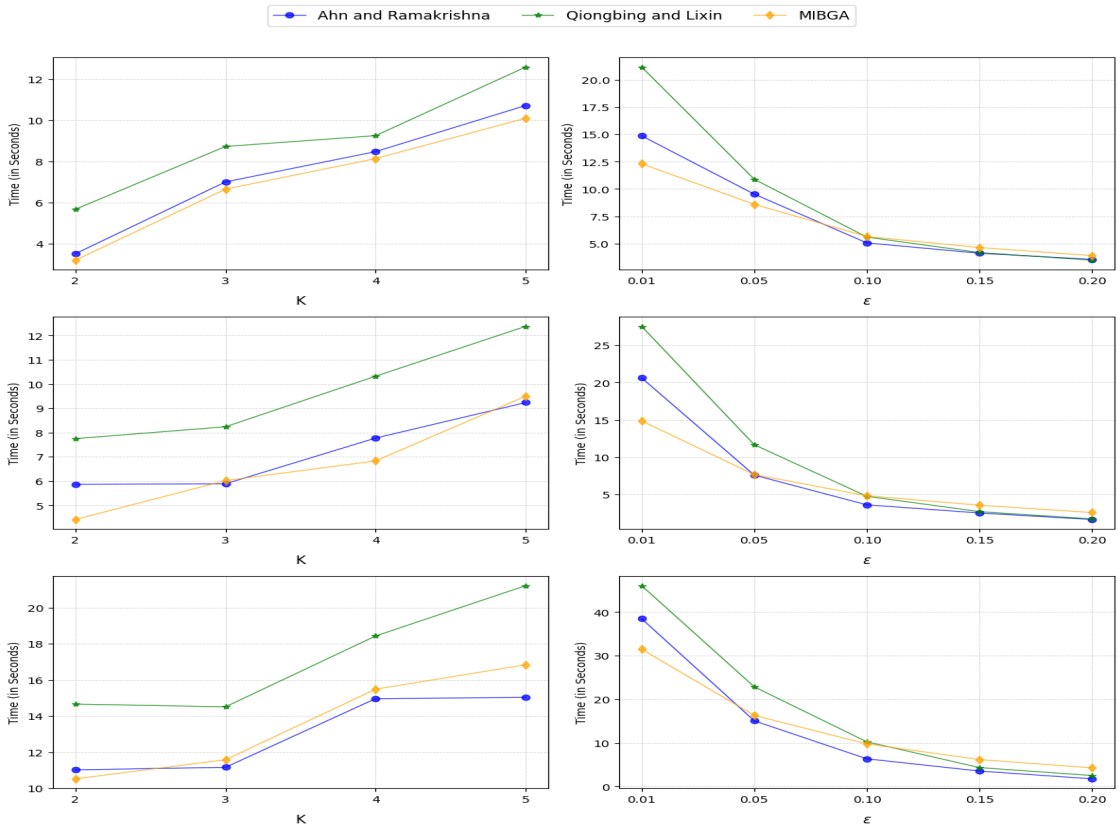

Figure 4.9: The variation in average execution time (in Seconds) by MIBGA and the GAs detailed in (Ahn & Ramakrishna 2002, Qiongbing & Lixin 2016) with respect to the user defined parameters, the number of near shortest paths to generate (K) and the length threshold (ϵ) is shown across the road networks of Arizona, Washington, and Kansas.

We report the execution time of our MultiPath Island-Based Genetic Algorithm (MIBGA) and the GAs by Ahn & Ramakrishna (2002) and Qiongbing & Lixin (2016) with respect to the user defined parameters, the number of near shortest paths to generate (K) and the length threshold (ϵ). A routing algorithms time complexity, such as Dijkstra algorithm, is dependent on the networks node and vertex counts. Using Figure 4.9, we report the computation time on each network independently because we employed a variety of network topologies with different numbers of nodes and vertices.

The results in Figure 4.9 reveal, the average execution time for all algorithms decreases with the higher values of ϵ. This highlights the effective exploration capabilities of GAs in a larger solution space. One possible reason for this behaviour is the random initialisation procedure employed for path construction, as discussed in Section 4.3.1. A relaxed length threshold (ϵ), enabled the algorithms to explore more optimal candidate paths that satisfy the threshold. Notably, the same initialisation procedure was applied across all three GAs, ensuring a consistent basis for comparison. Among the algorithms considered, MIBGA outperformed or matched the others for smaller values of ϵ, while the GA proposed in (Ahn & Ramakrishna 2002) performed slightly better for larger values, particularly on the Kansas network. With respect to K, MIBGA majorly outperforms both GAs on the network of Arizona and Washington across all values of K, while the GA from Ahn & Ramakrishna (2002) performed better on the Kansas network. The GA from Qiongbing & Lixin (2016) exhibits the longest execution time on all networks, indicating it may not scale well with larger networks.

4.5.4 Discussion on experimental results

In Section 4.5.2, we demonstrated the superiority of MIBGA over the genetic algorithms presented in (Ahn & Ramakrishna 2002, Qiongbing & Lixin 2016), as it generated the most diverse (less overlapping) path sets across all combinations of the number of near shortest paths to generate (K) and the length threshold (ϵ). The results were further validated as statistically significant through the Wilcoxon signed-rank test, with a Bonferroni-corrected p-value of 0.0025. The main reason for these results is shown in Figure 4.10, where average number of paths produced by each algorithm across all 20 combinations of (K, ϵ) are displayed.

Coloured bars represent the rounded average derived from 300 test cases (100 per combination on each network). The standard deviation is shown by black bars, with red highlights denoting high variability when it surpasses the average. MIBGA consistently produced more pathways inside the length threshold than the GAs by Ahn & Ramakrishna (2002) and Qiongbing & Lixin (2016). These additional paths produced by MIBGA provided more options for creating a more diverse set of paths. As a result, MIBGA outperforms the other GAs (Ahn & Ramakrishna 2002, Qiongbing & Lixin 2016) by consistently identifying more diverse and effective path sets. We acknowledge that the additional paths increased MIBGAs computation time when evaluating diversity, as shown in Figure 4.9 with K set to 4 and 5. However, its faster convergence, demonstrated by the lower portion of timeout instances in Figure 4.7 compensated for the added overhead. Examining

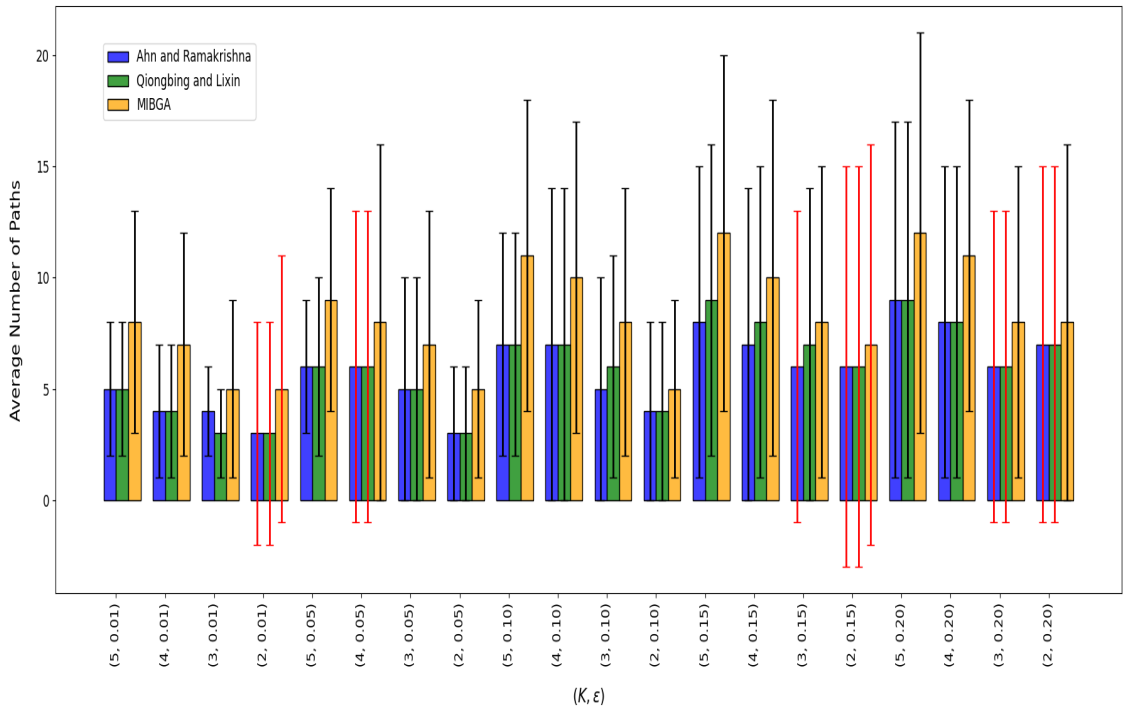

Figure 4.10: Average number of paths found by each algorithm across all 20 combinations of (K, ϵ). Distinct coloured bars represent the averages for each algorithm, while the black bars indicate the standard deviation. Combinations where the standard deviation exceeds the average are highlighted in red to indicate high variability. Results are aggregated across three networks, with each bar reflecting the rounded average based on 300 test cases (100 on each network).

these factors enabled us to interpret the data more effectively and draw well informed conclusions in Chapter 6.

4.6 Practical applications of MIBGA

In this research, MIBGA is introduced as an effective approach for solving the KMDNSP problem on real-world road networks. The proposed method offers valuable advancements in vehicle routing, alternate routing, and decision making for intelligent transportation systems. The key takeaways, in relation to management of these real-world applications, are as follows under a number of different headings, illustrated using real-world scenarios:

- **Balancing path length and route diversity:** Our study emphasises the trade-off between travel distance and path diversity, highlighting that alternative routes can be both feasible and adequately distinct. This balance can be very useful in real-world scenarios such as transportation of hazardous material, emergency response routing for ambulances or fire trucks, and wildfire evacuation planning, where having multiple effective routes enhances reliability and safety (Liu, Jin, Yang & Zhou 2017). It also benefits logistics operations (Wang et al. 2020), where diversified routing can improve delivery flexibility and service quality.

- **Enhanced decision making for route selection:** Traditional K-Shortest Path

(KSP) algorithms (Eppstein 1998) prioritises distance minimisation but fail to consider diverse (less overlapping) and practical alternatives necessary for real-world applications. In contrast, MIBGA efficiently generates multiple structurally diverse routes within a near-shortest distance range, giving users a variety of different paths to choose from. This makes the approach useful in systems like school bus routing, where specific routes can assist in navigating everyday traffic issues, bicycle routing, where low-traffic alternatives are the main focus, and smart city or GIS based navigation platforms, which aim to customise routing based on user constraints and preferences.

- **Implications for traffic management and sustainability:** By providing multiple near-optimal paths, MIBGA supports traffic decongestion strategies by distributing vehicles across alternative routes rather than concentrating them on one or two optimal paths. This has direct benefits such as, reducing traffic congestion, fuel consumption, and CO2 emissions, thereby contributing to more sustainable urban mobility. Practical use cases include tourist navigation systems, ride sharing platforms, and public transport planning, where load balancing across diverse routes enhances overall efficiency.

- **Optimisation strategies for Parallel Genetic Algorithms:** The proposed island-based evolutionary framework introduces an efficient migration and selection strategy that accelerates convergence while preserving solution diversity. This approach can be extended beyond transportation, to other combinatorial optimisation problems where diverse solutions are required.

Overall, this research highlights the practical significance of MIBGA for modern transportation systems, demonstrating that GAs can effectively enhance route planning and operational efficiency in complex real-world networks.

4.7 Chapter summary

This chapter presents the design, implementation, and evaluation of our MultiPath Island-Based Genetic Algorithm (MIBGA) for the K-Most Diverse Near-Shortest Paths problem on real-world road networks. Unlike traditional shortest path (EW 1959) and K-shortest path (Eppstein 1998) algorithms, which primarily focus on identifying paths based solely on their length, MIBGA emphasises on generating a set of k-near shortest paths while ensuring maximum structural dissimilarity among them. For this, MIBGA leverages parallel genetic algorithms within an island-based framework, ensuring efficient path optimisation while preserving solution diversity. By employing a variable-length chromosome representation 4.3.1, a specialised mending function 4.3.2, and adaptive selection mechanisms 4.3.9, MIBGA is capable of generating multiple optimal paths tailored to real-world constraints and user preferences.

Experimental analysis was organised into several sections, where we compare the performance of our proposed MIBGA against the two well-known GAs by Ahn & Ramakrishna

(2002) and Qiongbing & Lixin (2016), in the context of path routing. First in Section 4.5.1, we showcase the ability of all algorithms to identify K or more paths within the specified threshold for length determined by ϵ. MIBGA outperformed both GAs in (Ahn & Ramakrishna 2002, Qiongbing & Lixin 2016) with higher non-timeout instances on all networks, specifically with the smaller ϵ values, such as 0.01 and 0.05. Then in Section 4.5.2, we demonstrated the superiority of MIBGA against the GAs in (Ahn & Ramakrishna 2002, Qiongbing & Lixin 2016) on generating more diverse (less overlapping) path sets. The Wilcoxon signed-rank test with Bonferroni-corrected p-value of 0.0025 and effect size based on Cohen (2013) method, was used to highlight the statistical differences between the results and quantify their magnitude.

In Section 4.5.3, we reported the variation in runtime of each algorithm with respect K and ϵ. MIBGA majorly outperformed both GAs on the Arizona and Washington networks with respect to K, while the GA from Ahn & Ramakrishna (2002) excelled on the Kansas network. Similarly, for smaller values of ϵ, MIBGA matched or outperformed other algorithms, whereas the GA in (Ahn & Ramakrishna 2002) showed slightly better performance for larger ϵ values. The GA from Qiongbing & Lixin (2016) had the longest execution times, indicating poor scalability.

Taken together, these findings show that the island-based genetic design of MIBGA is well-suited to the KMDNSP problem, it maintains solution diversity, scales to large real-world networks, and achieves competitive runtimes under varying values of K and ϵ. In doing so, this chapter demonstrates the effectiveness of an Island-Based Genetic Algorithm as a practical computational strategy for generating diverse near-shortest paths, thereby fulfilling the objective that motivated the research question at the outset. The chapter concludes by discussing the practical applications of MIBGA, emphasising the key advancements in vehicle routing, alternate routing, and decision making for intelligent transportation systems.

In **Chapter 5**, we introduce our Parallel Optimal-route Search (POS) algorithm for multi-criteria vehicle routing on real-world road network. Designed to enhance routing efficiency while addressing general public needs, POS aims to generate multiple optimal paths, optimised across multiple objectives. To define the objective space for our POS algorithm, 25% of the total nodes in each network were randomly assigned as amenities (Charging stations, Hotels, and Pubs), with an equal distribution of nodes among the three amenities. Code to create these multi-feature networks is made available in the Jupyter Notebook Multi-feature Networks on the GitHub profile: POS. Our goal is to maximise these amenities using a fitness metric that has been specifically designed for this purpose, explained in Chapter 5.2.1.

Chapter 5

Parallel Optimal-route Search for Multi-Criteria Vehicle Routing

Related publication: Sharma, H., Galván, E. and Mooney, P., 2025. A parallel genetic algorithm for multi-criteria path routing on complex real-world road networks. Applied Soft Computing, 170, p.112559, Link to paper.

5.1 Introduction

A Parallel Genetic Algorithm (PGA) specifically designed for multi-criteria vehicle routing is described in this chapter. The algorithm aims to enhance the existing routing methods by offering users the ability to choose their preferred path from a set of optimal paths optimised across multiple objectives. These objectives are defined at the path level and integrate both path length and accessibility to relevant Points of Interest (POIs), such as hotels, charging stations, and other amenities, allowing the algorithm to generate routes that are not only efficient but also contextually practical for real-world navigation applications intended for the general public. While this work illustrates the approach using amenities as POIs, the framework is flexible and can be extended to incorporate any quantitative criterion measurable along a path. This adaptability enables the generation of more realistic and user-centric routes than existing methods, with potential applications in areas such as tourist trip planning, shared mobility systems, and intelligent transportation services. Major contributions from this chapter include:

- The developed approach, called the Parallel Optimal-route Search (POS), is a novel PGA that employs a hybrid model combining Global Parallelisation and Island-Based strategies for multi-criteria vehicle routing (described in Section 5.2).
- In POS, new paths for evaluation are generated using the Loop-Free Path-Composer (LFPC) operators (described in Chapters 4.3.7 and 4.3.8). A comparative study in Section 5.5.1 demonstrates its superior performance over the traditional NBCPM operators (discussed in Chapter 2), producing 93.61% unique solutions compared to 67.14% with NBCPM.

- We introduce a novel island formation strategy that describes the process of creating sub-populations from the centralised population in Section 5.2.2. The use of a centralised population from the global parallelisation model eliminates the need for a migration strategy.
- We propose a novel fitness metric called Positional Count that prioritises minimising path length while also maximising access to specific POIs, such as Pubs, Hotels, and Charging stations (described in Section 5.2.1).
- Proposed the Cross-Population Ratio of Non-Dominated Individuals (cpRNI), a modified formulation of the classical RNI metric that enables fair cross-algorithm comparison by constructing a shared non-dominated population and proportionally crediting overlapping solutions. This extends RNI from a within-population diagnostic to a robust comparative performance indicator.
- To demonstrate the insensitivity of our approach to network topology and scale. We use highly complex real-world road networks (described in Chapter 3) instead of the simpler synthetic networks commonly adopted by the community. The four networks used contain between $3,000$ and $10,800$ nodes, compared to the few hundred nodes typically found in related work.
- We conducted comprehensive empirical experiments using well-known and robust evolutionary algorithms NSGA-II and NSGA-III (described in Chapter 2, Section 2.6.2) to evaluate the performance of our method (described in Section 5.5.2).

In the following section, we present a detailed description of our POS algorithm. First, we would like to highlight that similar to MIBGA (described in Chapter 4), POS employs a variable-length chromosome representation (Chapter 4.3.1) and a mending function (Chapter 4.3.2) to align with the problem requirements and prevent undesirable loops. The overall process is outlined in Section 5.2.5.

5.2 Parallel Optimal-route Search (POS)

5.2.1 Fitness evaluation metric

In GAs, the fitness function is a problem-specific assessment measure that evaluates potential solutions in the search space with the goal to be optimised. In this work, we propose a multi-criteria vehicle planning strategy that simultaneously optimises four objectives: route length, and the Positional Count, a metric designed to reduce the impact of long routes with high amenity counts for three distinct amenities, referred to as Pubs, Hotels, and Charging stations encountered along the route. Route length is to be minimised while the Positional Count of all amenities is to be maximised. Thus, we use the inverse of the route length as the fitness metric in Equation 4.5, making this a maximisation problem for simplicity. Given the impact on the GAs performance and convergence, it is very crucial

to precisely define the fitness function. Fitness evaluation metrics used in our work are described as follows:

- To find a route with the smallest distance from a source node (S) to a target node (T), the fitness function is defined in Equation 5.1, where f_p denotes the fitness and $|p|$ represents the length of path p, calculated using Equation 5.2. l_i represents the total number of nodes in the ith route, with $g_i(j)$ and $g_i(j+1)$ representing pairs of consecutive nodes and W representing the weight (distance) linked between these two nodes. It is noteworthy that Equations 5.1 and 5.2 are the same as Equations 4.5 and 2.5, and they are reproduced here for convenience.

$$f_p = \frac{1}{|p|} \tag{5.1}$$

$$|p| = \sum_{j=1}^{l_i-1} W_{g_i(j),g_i(j+1)} \tag{5.2}$$

- The positional count, which represents the fitness metric for three objectives (amenities), has been carefully designed by taking into account the position and density of each node of a particular amenity type along the route. The weight assigned to each amenity node can be calculated using Equations 5.3, 5.4, and 5.5. For an ith route in our population, the positional count (fitness value) for a specific amenity (objective) is represented by f_i in Equation 5.6. The count of occurrence of that amenity is denoted by n. The assigned positional weight, $pd_i(j)$, in Equation 5.3 is calculated using the position factor, $c_i(j)$, from Equation 5.5 and the density factor, $d_i(j)$, from Equation 5.4 for the jth amenity node. The distance of the jth amenity node from the starting node is represented by $s_i(j)$, as given in Equations 5.4 and 5.5. In Equation 5.3, we specify k (set to 5) as a factor to penalise each amenity node based on the overall path length.

$$pd_i(j) = \left\{ \begin{array}{ll} \dfrac{c_i(j) + \sqrt{d_i(j)}}{|p|_i^{k/2}}, & \text{if } n \geq 1 \\ 0, & \text{otherwise} \end{array} \right\} \tag{5.3}$$

$$d_i(j) = \left\{ \begin{array}{ll} s_i(j+1) - s_i(j-1), & \text{if } n > 1 \\ |p|_i, & \text{if } n = 1 \\ 0, & \text{Otherwise} \end{array} \right\} \tag{5.4}$$

$$c_i(j) = \left\{ \begin{array}{ll} |p|_i - s_i(j), & \text{if } n \geq 1 \\ 0, & \text{Otherwise} \end{array} \right\} \tag{5.5}$$

$$f_i = \begin{cases} \sum_{j=1}^{n} pd_i(j), & \text{if } n \geq 1 \\ 0, & \text{Otherwise} \end{cases} \tag{5.6}$$

In order to offer a comprehensive grasp of our fitness evaluation metric, we employ the illustrative example depicted in Figure 5.1. This visual representation incorporates nodes of distinct types: Ordinary nodes (depicted in white), Pubs (depicted in blue), Hotels (depicted in red), and Charging Stations (depicted in green). Within this context, a valid path ($S \rightarrow 4 \rightarrow 10 \rightarrow 19 \rightarrow T$) originating from source node S and terminating at target node T is represented through highlighted edges (in orange). This path consists of five nodes: two representing Pubs, one representing the source node S, one representing the target node T, and one representing a Hotel. Notably, there were no charging stations along the route. For the given objectives, the fitness evaluations for the corresponding path are shown in Table 5.1. Fitness value associated with the distance between the source node S and target node T is calculated as the reciprocal of the Path Length. The fitness value referred to as Positional count for all amenities is computed using Equation 5.6. In more general terms, it signifies the cumulative weight assigned to each encountered amenity node along the route. This weight is denoted as Positional Density and is determined using Equation 5.3. The Position Factor is calculated with Equation 5.5, which adjusts the weight of an amenity based on its proximity to the target node (resulting in higher weights for amenities closer to the source node). Meanwhile, the Density Factor is calculated using Equation 5.4, assigning weight to each amenity node based on its distance from the preceding and subsequent amenity nodes. Fitness value for the amenity Charging Station is zero due to the absence of charging stations along the route.

Objectives	**Path Length**	**Pubs**		**Hotels**	**Charging Stations**
Nodes		4	10	19	–
Position Factor	$L = (57 + 66 + 230 + 72) = 425$	$c_1(4) = (425 - 57)$	$c_1(10) = (425 - (57 + 66))$	$c_1(19) = (425 - (57 + 66 + 230))$	0
Density Factor		$d_1(4) = (57 + 66)$	$d_1(10) = (425 - 57)$	$d_1(19) = 425$	0
Positional Density		$pd_1(4) = (368 + \sqrt{123})/425^{5/2}$	$pd_1(10) = (302 + \sqrt{368})/425^{5/2}$	$pd_1(19) = (72 + \sqrt{425})/425^{5/2}$	0
Fitness Value	$f_1(\text{Length}) = 1/425$	$f_1(\text{Pubs}) = pd_1(4) + pd_1(10)$		$f_1(\text{Hotels}) = pd_1(19)$	$f_1(\text{Charging Stations}) = 0$

Table 5.1: Fitness evaluation for the four objectives considered in this chapter is presented for the path, $S \rightarrow 4 \rightarrow 10 \rightarrow 19 \rightarrow T$, highlighted with orange edges in Figure 5.1. Positional density is computed using Equation 5.3, with k set to 5. f_1(Charging Stations) is evaluated as 0, because no nodes representing charging stations are present in the path.

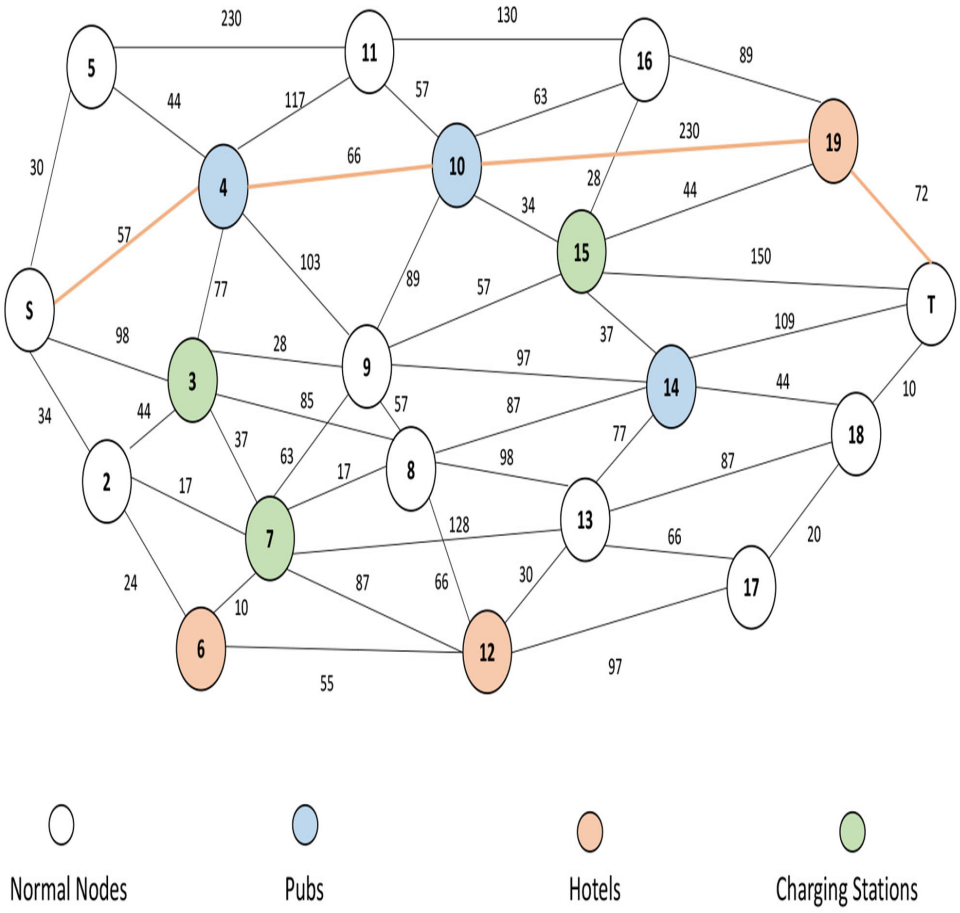

Figure 5.1: Graph used to illustrate the fitness metric in Table 5.1 is shown. Each node is assigned a unique number as an ID, and nodes representing amenities Pubs, Hotels, and Charging stations are depicted in blue, red, and green, respectively. The sample path from S to T is highlighted using orange edges.

5.2.2 Island formation strategy

Many wildlife species exhibit tendencies to live in groups with others of their kind, providing them with security, comfort, and a greater number of members for foraging (Barta et al. 1997, Li et al. 2002, Ward & Webster 2016). From an evolutionary perspective, this behaviour supports the notion of "survival of the fittest" by affording more opportunities for the strongest individuals to pass on their genes and secure their lineage. Taking inspiration from this idea of group living and elitism, we attempt to replicate this process by organising candidate solutions (possible solution paths) into groups consisting of a few strong candidates and other more general candidates. To provide strong candidate paths the opportunity to produce offspring with the other strong candidate paths, a replacement strategy allowing strong candidates to be considered in both Superior Population and Centralised Population groups is employed. The pseudocode for the island formation process, along with an example, is presented in Algorithm 4 and Figure 5.2, respectively. The island formation strategy is outlined as follows:

Algorithm 4 Island Formation Strategy

1: **Input:** Centralised population (Pop), Selection Threshold (E), (S_m-M_m) and (S_f-M_f) representing size range for PSP and PCP groups respectively.
2: **Output:** Collection of Islands consisting sub population (G).

3: Sort population Pop using Non-Dominating Sorting with Improved crowding distance;
4: *Superior Population*: s = Top E% of Pop;
5: *Centralised Population*: $r = Pop$;
6: Let $IslandCollection$ be an empty group: $G = \{\}$.
7: **while** $|r| \neq 0$ **do**
8: Let $CurrentIsland$ be an empty group: $c = \{\}$;
9: Let PSP be an empty group: $m = \{\}$;
10: Let PCP be an empty group: $f = \{\}$;
11: Preset the length of group m and f;
12: $|m| \leftarrow$ a random number in the range $[S_m, M_m]$;
13: $|f| \leftarrow$ a random number in the range $[S_f, M_f]$;
14: **if** $|f| \leq |r|$ **then**
15: Randomly extract $|m|$ elements (With Replacement) from group s and add to m: $m \leftarrow m \cup \{x_1, \ldots, x_{|m|}\}$;
16: Randomly extract $|f|$ elements (Without Replacement) from group r and add to f: $f \leftarrow f \cup \{x_1, \ldots, x_{|f|}\}$;
17: Add groups m and f to group c: $c \leftarrow c \cup \{m, f\}$;
18: Add group c to group G: $G \leftarrow G \cup \{c\}$;
19: **else if** $|r| \geq 1$ **then**
20: Randomly extract $|m|$ elements (With Replacement) from group s and add to m: $m \leftarrow m \cup \{x_1, \ldots, x_{|m|}\}$;
21: Extract the remaining elements from group r and add to f: $f \leftarrow f \cup \{x_1, \ldots, x_{|f|}\}$;
22: Add groups m and f to group c: $c \leftarrow c \cup \{m, f\}$;
23: Add group c to group G: $G \leftarrow G \cup \{c\}$;
24: **end if**
25: **end while**
26: **return** G

1. The initial population of solutions must first be sorted based on the fitness metrics explained in Section 5.2.1 using the Non-Dominating Sorting with Improved Crowding Distance (ICD), described in Chapter 2 (see line 4 in Algorithm 4).

2. Once sorted, the population is categorised into two groups: the Superior Population and Centralised Population, using a parameter called the Selection Threshold. This refers to the portion of current best solutions that we shall consider as Superior Population. For instance, if the Selection Threshold is 0.25 or 25% this means that the top 25% of solutions from a generation will be Superior Population while the Centralised Population will consist the entire population of the current generation (see line 5 and 6 in Algorithm 4).

3. We set the minimum and maximum sizes for two groups: PSP: Parents from Superior Population and PCP: Parents from Centralised Population.

4. We randomly select a number within the specified range of the minimum and maximum group sizes to determine the size of each group (see line 13 and 14 in Algorithm 4). The island size is the total number of individuals within the island, including both groups mentioned earlier. The PSP group comprises solutions from

the Superior Population while the PCP group comprises solutions from the Centralised Population.

5. After determining the sizes of the PSP and PCP groups, we proceed to randomly select the specified number of solutions from the Superior Population and the Centralised Population respectively. These selected solutions are then added to their respective groups (see line 16-18 in Algorithm 4).

6. To promote elitism throughout the algorithm, the selection from the Superior Population is made using a with replacement strategy while the selection from the Centralised Population is made with out the replacement strategy (see lines 16, 17, 21 in Algorithm 4).

If it is not possible to extract the specified number of PCP from the Centralised Population group, the process follows one of the following steps based on the situation:

- If the Centralised Population group have one or more solutions, a new island is created with the groups PSP (from Superior Population) and PCP (remaining solutions from the Centralised Population), and the process is terminated.

- If the Centralised Population group has no solutions, the process is terminated.

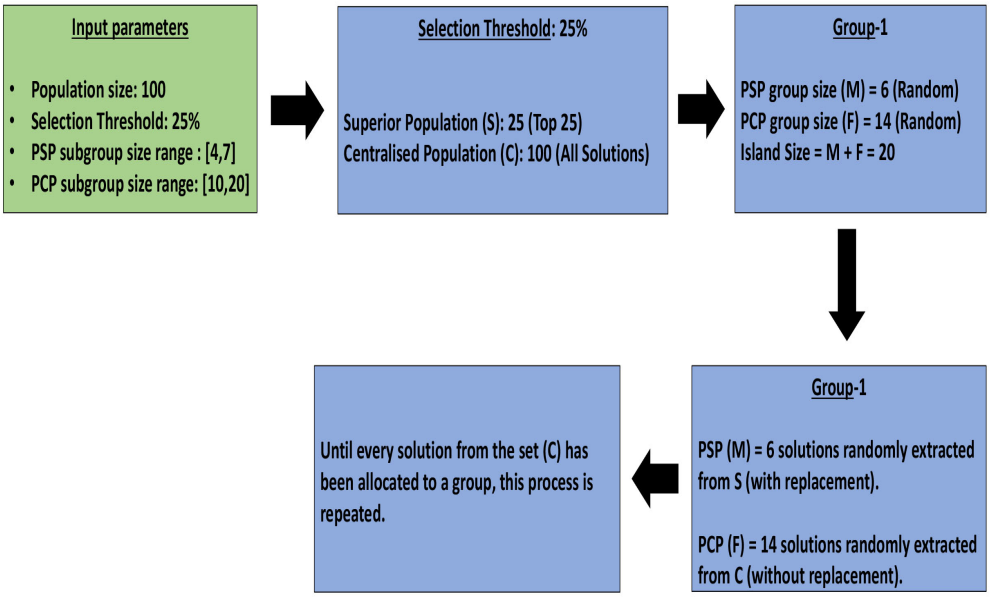

Figure 5.2: An example illustrating the Island formation process is shown in this Figure. Using a Selection Threshold of 25%, two population groups, Superior Population (S) and Centralised Population (C) are created. Group-1 (or Island-1) is then formed, consisting of one group of solutions extracted from S termed PSP and another group of solutions extracted from C termed PCP. The process continues until every member of C has been assigned to a group.

In our work, Figure 5.2 illustrates how islands are formed. At each generation, the process may yield different number of islands and each island may have a different size.

Minimum and Maximum group sizes are used to maintain selection pressure in the ordinal ranking (Goldberg 1989) scheme explained in Section 5.2.4 for the parent selection process to generate offsprings.

5.2.3 Offspring generation strategy

New candidate paths for evaluation are generated through an island-wise offspring generation process that confines the operation to members of the same island within the solution space. Offspring produced by each island are stored separately and added to the centralised population for the selection process described in Section 5.2.4. A Loop-Free Path-Composer (LFPC) is used to generate new paths in the offspring generation process. LFPC can generate a new path in two ways. The primary means is via the crossover operator, while the second approach is to apply the mutation operator on top of the crossover operator, which is only used for a limited number of paths. The crossover and mutation operator used to generate new paths are briefly described in Sections 4.3.7 and 4.3.8, respectively. It is worth noting that for the parent selection process, the selection of solutions from both the PSP and PCP groups at each island will be done using the Ordinal-based selection (Goldberg 1989) explained in Section 5.2.4. The pseudocode to perform the offspring generation process is explained in Algorithm 5.

Algorithm 5 Offspring Generation Strategy

Input: G, Collection of Islands with 2 groups of population within each island, Mutation Probability (M_p).
Output: C, Newly generated solution set for Evaluation.

Let $ChildPopulation$ be an empty set: $C = \{\}$.
for i in G **do**
  $CurrentIsland$: $I_c \leftarrow i$;
  Solutions from PSP group: $m \leftarrow$ PSP group from I_c;
  Solutions from PCP group: $f \leftarrow$ PCP group from I_c;
  Let New solutions from current island be an empty group: $o = \{\}$;
  while $|o| \neq |f|$ **do**
    PCP_{Parent}: $P_f \leftarrow$ a solution drawn from f using weights assigned through ordinal selection;
    PSP_{Parent}: $P_m \leftarrow$ a solution drawn from m using weights assigned through ordinal selection;
    $|r| \leftarrow$ a random number in the range $[1, 100]$;
    if $|r| \leq M_p$ **then**
      //Generate new solutions using LFPC (Crossover and Mutation).//
      $Child_1, Child_2 \leftarrow$ CrossoverWithMutation (P_f, P_m)
    else
      //Generate new solutions using only LFPC (only Crossover).//
      $Child_1, Child_2 \leftarrow$ Crossover (P_f, P_m)
    end if
    Add $Child_1$ and $Child_2$ to o: $o \leftarrow o \cup \{Child_1, Child_2\}$;
  end while
  Add o to C: $C \leftarrow C \cup \{o\}$;
end for
return C

5.2.4 Selection process

Our proposed approach incorporates two distinct selection strategies, each with a specific purpose. The first strategy is based on ordinal ranking (Goldberg 1989), which involves ordering the individuals in the population based on their fitness value. This approach is used to select parents for the offspring generation process based on their rank, rather than their actual fitness level. To assign ranks to candidate solutions, we employ Non-Dominating Sorting with Improved Crowding Distance (ICD) (Yue et al. 2021), which prefers solutions from smaller rank fronts over those from larger rank fronts. When two solutions belong to the same front, the solution with higher ICD is considered better. The ranking is assigned in ascending order, with the top-performing individual receiving the first rank, the second-best receiving the second rank, and so on. The selection of individuals for reproduction is determined by using selection probability proportional to their rank, meaning individuals with better (small) ranks have a greater chance of being selected for reproduction. This selection strategy is utilised for selecting solutions from both the PSP and PCP groups at each island during the offspring generation process and as explained in Section 5.2.3. The offspring generation process is applied to each island individually so this selection strategy entails ranking the solutions of both groups at each island separately. Second, a Non-Dominating Selection process determines the solutions that will be carried forward to the next generation.

The steps to perform Non-Dominating Selection are summarised as follows:

1. Add the newly generated solutions to the Centralised Population.
2. Calculate the Pareto front and ICD for each solution in the Centralised Population.
3. Sort the Centralised Population with Pareto front in ascending order.
4. Sort the solutions belonging to the same Pareto front in descending order of ICD.
5. Select the top N solutions to be carried forward to the next generation as the Centralised Population.

The combination of selection strategies employed in this POS enhances its overall performance by optimising the parent selection process, maintaining selection pressure throughout the algorithm, and dynamically adjusting the number of evaluations for each island based on the size of its PCP group.

5.2.5 Outline of the overall process

The general structure for our proposed POS is represented in Figure 5.3. The overall process is presented in a step-by-step manner, as follows:

- The input is the topological database of the graph network from NetworkX (Hagberg et al. 2008) generated using the process described in Chapter 3.

- Initial set of paths, referred as Initial population, are created and evaluated using random initialisation described in Section 4.3.1.
- The Initial population is evaluated using the fitness evaluation metrics given in Section 5.2.1.
- The process referred to as the Island Formation Strategy in Section 5.2.2 is used to create islands containing two further groups each.
- Each iteration of the process ends by updating the Centralised Population using the selection process described in Section 5.2.4.
- Maximum generation count for the algorithm is set manually to terminate the algorithm when maximum generation count equals current generation count.
- After the initial population, each iteration of the process starts with the formation of new islands from the updated Centralised Population (Section 5.2.2), followed by Section 5.2.3 (Offspring generation process), Section 5.2.1 (fitness evaluation), and Section 5.2.4 (selection).
- At the completion of this iteration this accounts for one generation of the proposed GA. Upon termination, a set of optimal paths is returned by the algorithm.

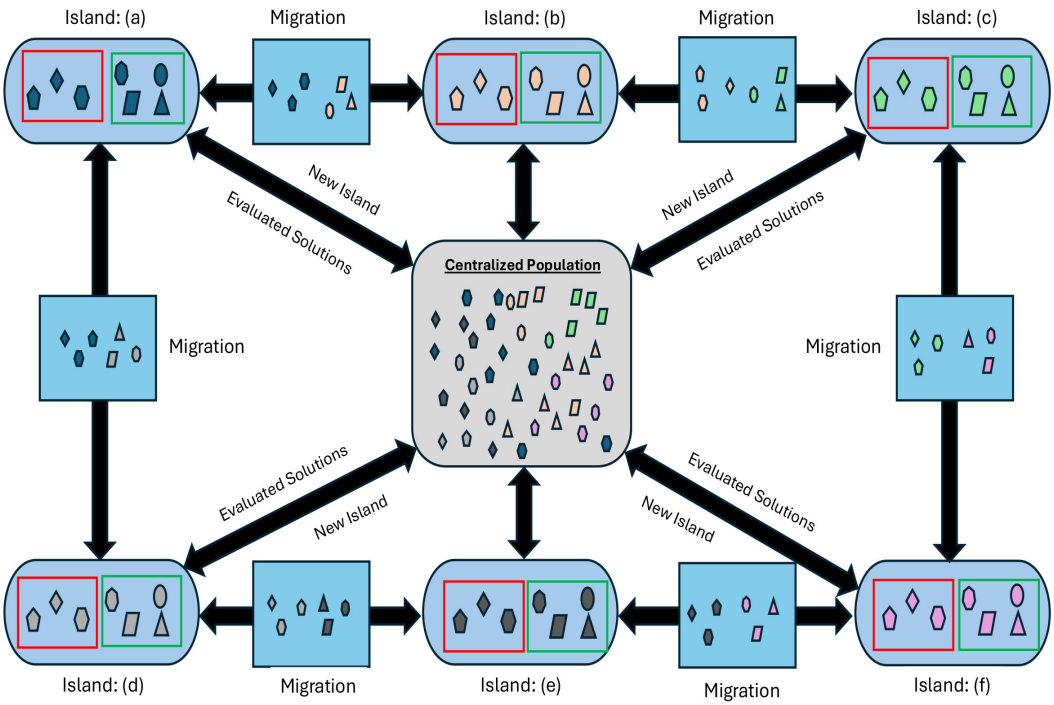

Figure 5.3: Proposed POS algorithm following a hybrid model where each slave node functions as an island, consisting of two groups: the red bounding box represents PSP (Parents from Superior Population), and the green bounding box represents PCP (Parents from Centralised Population).

5.3 Performance and evaluation metrics

The algorithms used in this chapter are compared using the metrics described in this section. These metrics are widely recognised for their capacity to demonstrate how well a multi-objective evolutionary algorithm balances the different objectives. By systematically applying these metrics, we aim to provide a comprehensive analysis of each algorithms advantages and disadvantages in optimising the multiple objectives within our study. The performance and evaluation metrics used are summarised as follows:

- **GAP (%):** This is an application specific evaluation metric that represents the percentage difference between the shortest path generated by a respective algorithm (under comparison) and the shortest possible path founded using Dijkstras algorithm. In vehicle routing cases where decreasing travel distance is a priority, a reduced GAP indicates that the paths generated by an algorithm are closer in length to the best shortest paths. When making comparisons, an algorithm with a lower GAP (%) is considered better than another algorithm with a higher GAP (%) (Di Placido et al. 2022).

- **Cross-Population Ratio of Non-Dominated Individuals (cpRNI):** The Ratio of Non-dominated Individuals (RNI) in a population (Tan et al. 2002), given as

$$RNI = \frac{\text{Number of Non-dominated Individuals}}{\text{Total Number of Individuals}} \tag{5.7}$$

  RNI as a metric to compare the performance of different multi-objective algorithms has been reported in several studies (Yang & Natarajan 2010, Subashini & Bhuvaneswari 2012, Rajeswari et al. 2021, Hasan et al. 2023). However, it is important to recognise that the number of non-dominated solutions within a population inherently influenced by the overall quality of solutions in that population. For instance, a population with many high-quality solutions, may produce fewer non-dominated solutions in comparison to a population with many low-quality solutions.

  Consequently, directly comparing RNI values across independently evolved populations can be misleading. Therefore, to accurately assess the performance of competing algorithms in terms of RNI, a shared population allowing the solutions from different approaches to compete against each other is necessary. To address this limitation, we introduce the Cross-Population Ratio of Non-Dominated Individuals (cpRNI), an adaptation of the classical RNI (Tan et al. 2002) designed for cross-algorithm comparative evaluation within a unified non-dominated population. In this comparative study, we are dealing with six different simulations (2 simulations each of NSGA-II, NSGA-III and POS) for each test scenario. Each simulation produces its own set of optimal solutions at the end of its run. Steps performed to calculate cpRNI in are outlined as follows,

  - We combined the optimal sets returned by NSGA-II, NSGA-III, and POS (2 simulations each) into a single optimal set.

- Using the non-domination criterion explained in Section 2.6.2, we create a subset of individuals belongings to the first front in the Pareto optimal set. This subset of individuals is our population.

- For each algorithm, we determine the number of solutions it contributed towards the population.

  * If a solution in the population is generated by more than one algorithm, then the count of all algorithms that generated the respective solution is incremented by one, and the size of the population is incremented by $(n-1)$, where n is the number of algorithms that contributed to the solution.

  * For example, suppose we are dealing with two algorithms Alg. 1 and Alg. 2. The total number of non-dominated solutions are 10. Out of the 10 non-dominated solutions, 5 are generated by only Alg. 1, 3 are generated by only Alg. 2 and 2 are generated by both Alg. 1 and Alg. 2. Then the cpRNI value for Alg. 1 will be 0.58 $(\frac{7}{12})$ while for Alg. 2 it will be 0.42 $(\frac{5}{12})$.

- cpRNI value for each algorithm will be between $0 < cpRNI \leq 1$, where $cpRNI = 1$ meaning all the solutions in the population are generated by the respective algorithm. $cpRNI = 0$ means no solution in the population is generated by the respective algorithm.

- When making comparisons, an algorithm with a higher cpRNI value is considered better than another algorithm with a lower cpRNI value.

- **The overall Hypervolume (HV) captured by non-dominated population:** The hypervolume indicator (Zitzler & Thiele 1998) measures the quality of a set P = $\{p_1, p_2, \ldots, p_n\}$ containing n non-dominated objective vectors produced by a multi-objective optimizer, such as a multi-objective evolutionary algorithms (Fonseca et al. 2006). For a minimization problem with d objectives, the hypervolume indicator measures the area simultaneously dominated by the set P and bounded above by a reference point $\mathbf{r} \in R^d$, where $\mathbf{r} \geq (\max_p p_1, \ldots, \max_p p_d)$. Here, each objective vector $p = (p_1, \ldots, p_d) \in P \subset R^d$, and the comparison $(\geq)$ is performed componentwise (Fonseca et al. 2006, Coello et al. 2007). Higher hypervolume values reflect a more accurate representation of the Pareto front. In this study, hypervolume is calculated using the hypervolume calculator from the Python library Pymoo (Blank & Deb 2020). Each objective function in the Pymoo library is expected to be minimised, and each constraint must be specified in the manner of ≤ 0 (Blank & Deb 2020). As a result, we multiplied each objective value by -1 and minimised $-f_i$, where f_i is the fitness value for the ith objective. To capture the volume of the objective space dominated by the set of solutions, a reference point specifying the lower bound for all objectives is also provided.

- **Uniform Distribution (UD) of non-dominated population:** As the name suggests, the Uniform Distribution (UD) metric assesses the degree to which the solutions constituting the Pareto front have a uniform distribution. It quantifies the distribution of non-dominated individuals in the population (Srinivas & Deb 1994, Tan et al. 2002) to measure their spread across the Pareto front. Mathematically, $UD(X')$ for a given subset of non-dominated individuals X' in a population X, where $X' \subseteq X$, is defined as

$$UD(X') = \frac{1}{1 + S_{nc}} \tag{5.8}$$

where S_{nc} refers to the standard deviation of niche count (Tan et al. 2002) for the set X'. The niche count is then formulated as follows:

$$S_{nc} = \sqrt{\frac{\sum_i^{N_{x'}} (nc(x'_i) - \overline{nc}(X'))^2}{N_{x'} - 1}} \tag{5.9}$$

where $N_{x'}$ is the size of the set X', $\text{nc}(x'_i)$ is the niche count of ith individual x'_i where $x'_i \in X'$, and $\overline{nc}(X')$ is the mean of $nc(x'_i)$, $\forall i = 1, 2, ..., N_{x'}$, represented using the following Equations (Tan et al. 2001, 2002),

$$nc(x'_i) = \sum_{j,j\neq 1}^{N_{x'}} f(i,j), \text{ where } f(i,j) = \left\{ \begin{array}{ll} 1, & dis(i,j) < \sigma_{\text{share}} \\ 0, & else \end{array} \right\} \tag{5.10}$$

$$\sigma_{\text{share}} = \alpha \frac{(max_k - min_k)}{\sqrt{N_{x'}}} \tag{5.11}$$

$$\overline{nc}(X') = \frac{\sum_i^{N_{x'}} nc(X'_i)}{N_{x'}} \tag{5.12}$$

where $dis(i,j)$ is the Euclidean distance between the individual i and the individual j in the single objective domain. The σ_{share} parameter is the adaptive method proposed in (Fonseca & Fleming 1998), where max_k and min_k represents the maximum and minimum fitness of the solutions for the kth objective, respectively. The constant α is set to 1 in our work. In terms of performance comparison, an algorithm with a higher $UD(X')$ value suggests a more thorough exploration of the Pareto front and search space. This means that the algorithm is considering a broader range of solutions in comparison to the algorithm with a lower $UD(X')$ value.

5.4 Algorithm description and configuration

In this section, we outline the algorithms used in the study along with their respective parameters, including the reasoning behind the parameter selection and the tuning process

employed. For each simulation on the respective real-world road network we randomly generate two numbers to represent the source (S) and target (T) nodes, ensuring S must not be equal to T and both S and T are integers that belong to the node set of the network graph. As stated in Section 2.6.2, NSGA-III uses a population size of N, which is the smallest multiple of 4 that is greater than or equal to H (Deb & Jain 2013). In this work, we are dealing with four objectives namely path length and Positional Count of Pubs, Hotels, and Charging stations. Based on empirical observations, number of Divisions p in Equation 2.8 is set to 7 (Palakonda & Mallipeddi 2020), resulting in a population size of 120. This ensures that the average computation time of each algorithm remains within one minute while maintaining solution quality. The candidate paths for each algorithm were initialised using the process described in Section 4.3.1, and to ensure a fair comparison, all algorithms under comparison were initialised using the same initial population and the same number of generations, set to 10. For all simulations in this work, crossover is used as the predominant genetic operator to generate new paths for evaluation, with the Crossover probability set to 1.00 (100 %). The Mutation Probability for each algorithm was determined through parameter tuning, with possible values ranging from 0.01 to 0.10. Complete list of parameters and the algorithms in which they are utilised is presented in Table 5.2.

- **Non-Dominated Sorting Genetic Algorithm (NSGA) II and III:** For each testing scenario, two simulations of both NSGA-II and NSGA-III were conducted. The first simulation uses the genetic operators Node Based Crossover and Path Mutation (NBCPM), while the second simulation employs the Loop-Free Path-Composer (LFPC) operators proposed in this work. In both NSGA-II and NSGA-III, solutions for the next generation are selected from the combined offspring and parent populations using non-dominated sorting. The key difference arises when selecting solutions within the same front. NSGA-II uses the original crowding distance metric, while NSGA-III relies on the perpendicular distance of a solution from a preset reference point in the normalised objective space. In the offspring generation process, NSGA-II employs tournament selection, while NSGA-III randomly selects two solutions from the solution space to generate offspring. For NSGA-III, the Mutation Probability was the only parameter requiring tuning. We employed a grid search (Bergstra & Bengio 2012), testing 10 possible values, and identified the optimal Mutation Probability as 0.03. In contrast, for NSGA-II, both the Mutation Probability and Tournament Size required tuning. While the original study by NSGA-II (Deb et al. 2002) used binary tournament selection, we extended our exploration with Tournament Size values from 2 to 10. This produced 90 possible combinations, all of which were assessed using a grid search (Bergstra & Bengio 2012). As a result, we determined the optimal settings for NSGA-II to be a Tournament Size of 5 and a Mutation Probability of 0.05. The number of offspring produced during each generation equals the size of the initial population. After the solutions are sorted, only the top N solutions are moved to the next generation, where N refers to the size of the initial population.

Parameters	Value	NSGA-II	NSGA-III	POS
Number of tests on each network	50	✓	✓	✓
Population Size	120	✓	✓	✓
Number of generations	10	✓	✓	✓
Crossover probability	1.00	✓	✓	✓
Divisions	7	×	✓	×
Selection Threshold	0.20	×	×	✓
Mutation probability	0.05, 0.03, 0.03	✓	✓	✓
Tournament Size	5	✓	×	×
Minimum size of PSP group	5	×	×	✓
Maximum size of PSP group	9	×	×	✓
Minimum size of PCP group	5	×	×	✓
Maximum size of PCP group	9	×	×	✓

Table 5.2: List of parameters used for NSGA-II, NSGA-III, and the POS algorithm.

- **Parallel Optimal-route Search (POS):** Like the NSGA-II and NSGA-III approach, two simulations of POS are conducted. The first simulation uses the genetic operator NBCPM, while the second simulation employs LFPC operator. Parent selection is performed using ordinal ranking (Goldberg 1989) explained in Section 5.2.4. Table 5.2 displays five additional parameters that are specific to the island formation strategy used in this approach. Selection threshold determines the selection of superior solutions, as well as the Minimum and Maximum group size for both the PSP and PCP groups on each island. These additional parameters, along with the Mutation Probability (set to 0.03), were tuned using Latin Hypercube Sampling (LHS) (Stein 1987). The Selection threshold was varied between 0.20 and 0.30, the minimum and maximum sizes of the PSP group vary between 3 and 10, while the minimum and maximum sizes of the PCP group range from 5 to 12. The tuning process was implemented using the Python library pyDOE2, a well-recognised tool for efficiently sampling parameter spaces. The number of offsprings produced during each generation equals the size of the initial population. The solutions for the next generation are selected from the centralised population consisting of both parent and offspring populations using non-dominating sorting with the Improved Crowding Distance (ICD) metric (Yue et al. 2021) metric, explained in Section 2.6.2. Once the solutions are sorted using the Improved Crowding Distance metric, only the top N solutions, where N refers to the size of the initial population, are selected and moved to the next generation. It is note worthy that the use of different parame-

ters does not affect the fairness of comparisons, as all simulations generate an equal number of solutions for each test case.

5.5 Experimental results

In this section, we present the experimental analysis conducted to demonstrate the effectiveness of our Parallel Optimal-route Search (POS) algorithm in solving the multi-criteria vehicle routing problem on real-world road networks. First, we present a comparative study to highlight the superiority of our Loop-Free Path-Composer (LFPC) over the commonly used Node-Based Crossover and Path Mutation (NBCPM) operators (Ahn & Ramakrishna 2002, Lin et al. 2021, Zhu et al. 2014, Qiongbing & Lixin 2016) in GA based routing problems. Following this, in Section 5.5.2 we compare the performance of our POS algorithm against the well known Non-Dominated Sorting Genetic Algorithm (NSGA) II and III. The parameter tuning of each algorithm is provided in Section 5.4.

The evaluation metric, GAP (%), assesses the simulations based on their ability to generate shorter paths, which is crucial for a Vehicle Routing Problem (VRP). The evaluation metrics Cross-Population Ratio of Non-Dominated Individuals (cpRNI), Uniform Distribution (UD), and Hypervolume (HV) assess the non-dominated set of solutions, providing a comprehensive evaluation of overall performance and solution quality. Details on these evaluation metrics can be found in Section 5.3. We conducted 50 independent test runs on each network, with the results averaged to produce the findings presented in Table 5.3. For each test case, each simulation generated 1200 solutions, and the proportion of unique solutions (fitness evaluations) produced by each simulation is presented as Mean Unique Solutions (MUS), expressed as percentage in Table 5.3. The computation time for each simulation is reported as Mean Computation Time (in seconds) in Table 5.3.

5.5.1 Comparative studies: Genetic operators NBCPM vs. LFPC

The results obtained when we compare the solutions generated by simulations employing the NBCPM operators against the simulations employing the LFPC operator are presented. As outlined in Section 5.4, we conducted two simulations for NSGA-II, NSGA-III, and the POS algorithm using the operators collectively referred to as NBCPM (Section 2.7.4) and LFPC (Section 4.3.7).

We begin our discussion by comparing the performance of the NBCPM operator with the proposed LFPC operator in terms of their ability to converge to the shortest path, given a pair of source and target nodes. This is evaluated using the GAP (%) evaluation metric described in Section 5.3, and the results for this comparison metric are summarised in the column labelled MGAP $\pm$ STD (%) in Table 5.3. Smallest MGAP is considered superior, and the results in Table 5.3, indicate that, on average the simulations employing the LFPC operator outperforms the simulations employing the NBCPM operator across all the networks in the study, achieving the smallest MGAP $\pm$ STD (%) of 4.41 $\pm$ 5.30 on the Oklahoma road network.

Algorithm	MGAP ± STD	MUD ± STD	McpRNI± STD	MHV ± STD	MUS ± STD	MCT ± STD
Oklahoma (3,351 nodes and 6,903 edges)						
NSGA-II (NBCPM)	7.02 ± 8.88	0.10 ± 0.06	0.22 ± 0.16	0.28 ± 0.14	61.06 ± 8.27	2.44 ± 0.35
NSGA-II (LFPC)	**4.41 ± 5.30**	0.09 ± 0.03	**0.26 ± 0.15**	0.34 ± 0.13	90.22 ± 6.14	7.09 ± 1.26
NSGA-III (NBCPM)	9.95 ± 12.06	0.11 ± 0.06	0.03 ± 0.03	0.16 ± 0.11	68.24 ± 9.70	2.52 ± 0.39
NSGA-III (LFPC)	8.38 ± 10.24	**0.12 ± 0.5**	0.04 ± 0.04	0.16 ± 0.10	**94.87 ± 5.78**	7.23 ± 1.10
POS (NBCPM)	6.88 ± 8.25	0.10 ± 0.05	0.21 ± 0.12	0.28 ± 0.15	61.13 ± 9.41	**2.30 ± 0.39**
POS (LFPC)	4.67 ± 6.00	0.11 ± 0.05	0.24 ± 0.13	**0.35 ± 0.15**	91.79 ± 5.74	7.38 ± 1.41
Arkansas (4,370 nodes and 8,983 edges)						
NSGA-II (NBCPM)	8.30 ± 8.11	0.10 ± 0.07	0.25 ± 0.16	0.30 ± 0.14	63.50 ± 7.15	3.17 ± 0.40
NSGA-II (LFPC)	7.69 ± 8.21	0.10 ± 0.05	0.19 ± 0.12	0.29 ± 0.10	91.61 ± 3.80	9.10 ± 1.34
NSGA-III (NBCPM)	14.48 ± 12.03	0.13 ± 0.14	0.02 ± 0.02	0.15 ± 0.07	70.54 ± 9.58	3.24 ± 0.44
NSGA-III (LFPC)	12.55 ± 11.50	**0.14 ± 0.14**	0.03 ± 0.03	0.15 ± 0.07	**96.22 ± 2.88**	9.45 ± 1.26
POS (NBCPM)	9.14 ± 9.27	0.10 ± 0.08	**0.27 ± 0.16**	0.29 ± 0.13	63.13 ± 8.68	**3.00 ± 0.37**
POS (LFPC)	**6.59 ± 7.70**	0.11 ± 0.05	0.23 ± 0.10	**0.33 ± 0.10**	93.11 ± 3.33	9.64 ± 1.42
Louisiana (6,981 nodes and 16,682 edges)						
NSGA-II (NBCPM)	16.03 ± 18.03	0.09 ± 0.06	0.24 ± 0.16	0.26 ± 0.15	64.40 ± 8.58	5.41 ± 0.89
NSGA-II (LFPC)	**10.31 ± 12.62**	0.10 ± 0.06	0.23 ± 0.12	0.31 ± 0.16	91.33 ± 5.41	15.25 ± 2.54
NSGA-III (NBCPM)	20.12 ± 20.00	0.12 ± 0.07	0.02 ± 0.03	0.14 ± 0.09	73.03 ± 10.22	5.77 ± 0.97
NSGA-III (LFPC)	18.53 ± 19.31	**0.13 ± 0.06**	0.03 ± 0.04	0.13 ± 0.07	**96.35 ± 5.03**	15.81 ± 2.44
POS (NBCPM)	14.33 ± 16.17	0.10 ± 0.06	0.21 ± 0.14	0.26 ± 0.14	64.51 ± 9.58	**5.15 ± 0.97**
POS (LFPC)	11.70 ± 14.47	0.10 ± 0.04	**0.26 ± 0.15**	**0.32 ± 0.15**	92.94 ± 5.64	15.89 ± 2.82
Texas (10,886 nodes and 24,464 edges)						
NSGA-II (NBCPM)	16.44 ± 13.96	0.08 ± 0.05	**0.25 ± 0.15**	0.30 ± 0.13	68.73 ± 8.25	9.10 ± 1.33
NSGA-II (LFPC)	13.67 ± 12.43	0.10 ± 0.06	0.24 ± 0.15	0.36 ± 0.14	93.14 ± 3.38	31.20 ± 6.65
NSGA-III (NBCPM)	21.73 ± 18.39	0.10 ± 0.05	0.02 ± 0.03	0.15 ± 0.07	77.99 ± 9.22	9.46 ± 1.39
NSGA-III (LFPC)	19.00 ± 17.34	**0.13 ± 0.09**	0.03 ± 0.04	0.17 ± 0.10	**97.61 ± 2.13**	32.05 ± 6.13
POS (NBCPM)	17.82 ± 16.53	0.09 ± 0.04	0.22 ± 0.12	0.29 ± 0.11	69.40 ± 9.01	**8.62 ± 1.36**
POS (LFPC)	**12.82 ± 12.80**	0.10 ± 0.04	**0.25 ± 0.14**	**0.41 ± 0.14**	94.16 ± 3.12	32.91 ± 7.35

Table 5.3: The performance of NSGA-II, NSGA-III, and the POS algorithm using the NBCPM and LFPC operator is presented for the road networks of Oklahoma, Arkansas, Louisiana, and Texas, as described in Section 3.4. The performance is evaluated using the metrics, Mean GAP ± Standard Deviation (MGAP ± STD (%)), Mean Uniform Distribution ± Standard Deviation (MUD ± STD), Mean Hypervolume ± Standard Deviation (MHV ± STD), Mean Cross-Population Ratio of Non-Dominated Individuals ± Standard Deviation (McpRNI ± STD), Mean Unique Solutions ± Standard Deviation (MUS ± STD (%)), and Mean Computation Time ± Standard Deviation (MCT ± STD (sec)), detailed in Section 5.3. Lower values of MGAP and MCT indicate better performance, while higher values of MHV, MUD, McpRNI, and MUS are preferred. The bold numbers highlight the best results for each metric on the respective network.

The distribution of non-dominated solutions across the Pareto front is assessed using the UD metric described in Section 5.3. The results for this comparison metric are summarised in the column labelled MUD ± STD in Table 5.3. A higher MUD value is considered better, and the results in Table 5.3 indicate that, on average the simulations employing the LFPC operator produced non-dominated solutions that were more uniformly distributed across the Pareto front. Figure 5.4 illustrates an example, where NSGA-II employing the LFPC operator had a better UD compared to the NSGA-II simulation using the NBCPM operator. As a result, the non-dominated set from the LFPC operator simulation is more evenly distributed across the objective space.

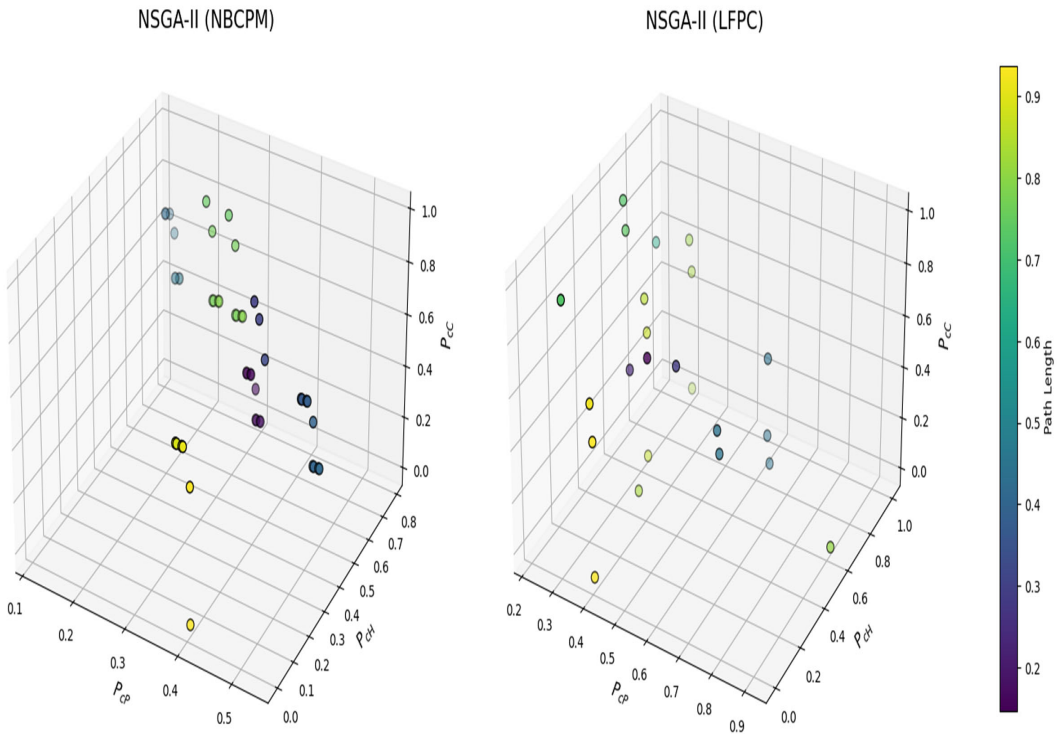

Figure 5.4: Example of a test cases, illustrating the distribution of non-dominated solutions generated by the two NSGA-II simulations on the Oklahoma network. X-axis, Y-axis, and Z-axis are represented by P_{CP} (Positional count for Pubs), P_{CH} (Positional count for Hotels) and P_{CC} (Positional count for Charging stations) respectively. Path length (4th objective) is represented using node colours, with dark yellow representing the highest fitness (smallest length) and dark blue representing the smallest fitness (highest length). The simulation using the LFPC operator had a better UD across the objective space compared to the simulation using the NBCPM operator.

The quality of solutions produced by each algorithmic simulation is assessed using a shared population of non-dominated solutions, allowing solutions from different algorithms to compete. A new non-dominated set is then derived from this shared population, and the Cross-Population Ratio of Non-Dominated Individuals (cpRNI) contributed by each simulation to the new set is computed. Complete procedure for calculating cpRNI is described in Section 5.3, and the results for this metric are summarised in the McpRNI ±

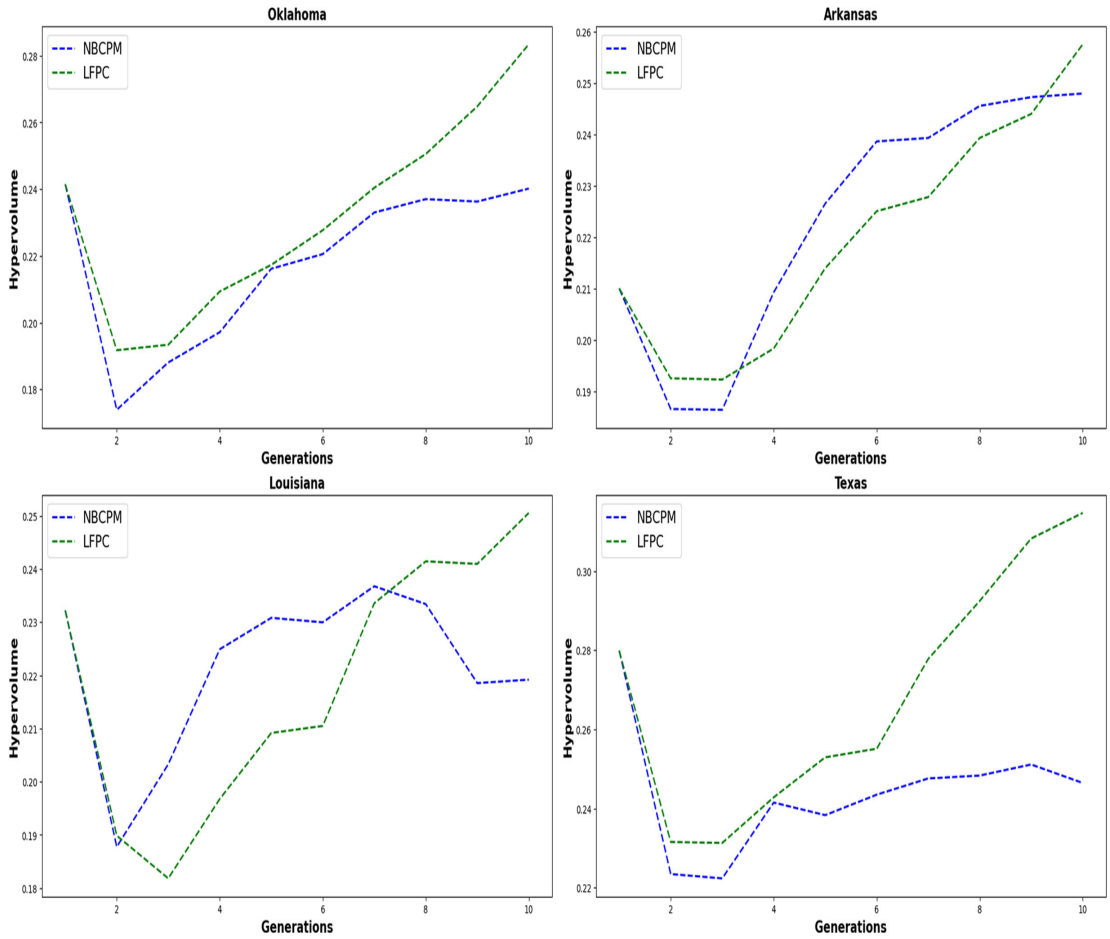

Figure 5.5: Variation in average hypervolume across 10 generations is examined for simulations employing the NBCPM and the LFPC operator. This analysis is based on aggregating the hypervolume data from all simulations employing the NBCPM operator and those employing the LFPC operator, across 50 test cases for each network, with respect to the number of generations. The results are presented in four distinct subplots, each illustrating the variation in average hypervolume for a specific network. Generation count is plotted on the x-axis, while captured hypervolume is on the y-axis.

STD column of Table 5.3. The value of cpRNI ranges from 0 to 1, with 1 representing the best outcome and 0 the worst. Table 5.3 highlights the superior performance of simulations using the LFPC operator compared to those using the NBCPM operator for the cpRNI metric. Except for the Arkansas network, simulations employing the LFPC operator consistently achieved higher cpRNI values than those using the NBCPM operator.

Hypervolume (HV) (Zitzler & Thiele 1998) is used to assess the simulations based on their ability to capture the volume within the objective space covered by the set of non-dominated solutions. We used the HV calculator from the Python library Pymoo (Blank & Deb 2020) for the assessment and the results for this metric are summarised in the MHV $\pm$ STD column of Table 5.3. A higher HV indicates better performance, and the results in Table 5.3 highlights that the simulations using the LFPC operator captured greater volume as compared to the simulations using NBCPM. Figure 5.5 illustrates the variation in average Hypervolume (HV) across 10 generations for simulations employing both the NBCPM and LFPC operator. The variation across all four networks is shown,

with simulations using the LFPC operator capturing a higher average HV at generation 10 on each network.

As stated in Chapter 3, time complexity of a routing algorithm depends upon the number of nodes and vertices in the network. As we used multiple network topologies with varying numbers of nodes and vertices, we report the computation time for each network separately. Mean Computation Time (MCT) represents the average time an algorithm takes to evaluate a test case on a given network. In genetic algorithms, computation time is directly correlated with the number of fitness evaluations performed (Yuen & Chow 2008). For each test case, although each simulation generates a total of $1,200$ paths, the fitness is evaluated only for the unique solutions produced. The results in Table 5.3 clearly show that simulations using the NBCPM operators outperform those using the LFPC operator in terms of MCT across all networks. However, this difference is largely due to the significant disparity in the number of unique solutions (fitness evaluations) generated by the simulations employing the NBCPM operator compared to those employing the LFPC operator, as shown in Table 5.3. As stated in Section 2.7.4, the requirement of a shared node between both parents introduces bias and reduces diversity. This resulted in an average of 67.14% (approx. 809) fitness evaluations per test case compared to 93.61% (approx. $1,123$) with the LFPC operator across all networks. Consequently, the LFPC based simulations outperformed the NBCPM based simulations on most metrics, albeit with higher computational cost.

The results presented in this section demonstrate the clear superiority of the LFPC operator over the NBCPM operator across all networks analysed in this work. Simulations employing the LFPC operator generated solutions that were shorter in length (MGAP (%)), captured a larger hypervolume (MHV), were more uniformly distributed across the Pareto front (MUD), and exhibited higher quality (McpRNI) compared to those generated using the NBCPM operator. While the simulations with the NBCPM operator had a shorter (better) computation time (MCT), this advantage was primarily due to significantly fewer fitness evaluations (MUS (%)) compared to the LFPC simulations. In the following section, we will present a detailed comparison between NSGA-II, NSGA-III, and the POS algorithm proposed in our study.

5.5.2 Comparative studies: Algorithms NSGA-II, NSGA-III, POS

This section aims to compare the performance of our Parallel Optimal-route Search (POS) algorithm with the well-known NSGA-II and NSGA-III in addressing the path routing problem. This comparison is used to demonstrate the effectiveness of the POS algorithm in solving the vehicle routing problem and to benchmark its performance against the well-known multi-objective optimisation techniques. As described in Section 5.5.1, we conducted two simulations for NSGA-II, NSGA-III and the POS algorithm. One simulation utilised the NBCPM operators, while the other employed the LFPC operator, which are explained in Sections 2.7.4 and 4.3.7 respectively.
The results of all six simulations, evaluated using the metrics discussed in Section 5.3, are summarised in Table 5.3. The data shows that the POS algorithm consistently outper-

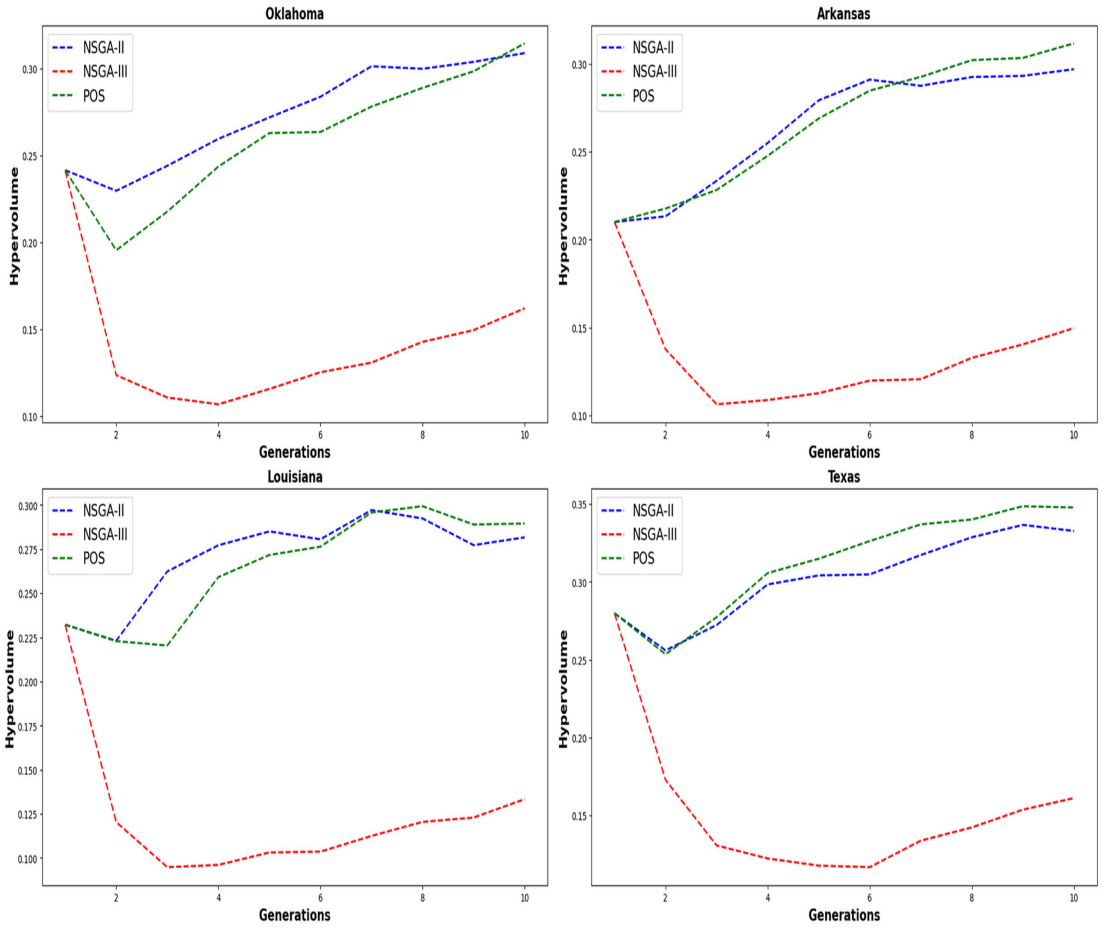

Figure 5.6: Variation in average hypervolume across 10 generations for the simulations of NSGA-II, NSGA-III, and proposed POS algorithm is shown in this plot. This analysis is based on aggregating the hypervolume data from all simulations of each algorithm, using both the NBCPM and LFPC operator, across 50 test cases per network, relative to the number of generations. The results are presented in four distinct subplots, each illustrating the variation in average hypervolume for a specific network. Generation count is plotted on the x-axis, while captured hypervolume is on the y-axis.

formed NSGA-II and NSGA-III across the four networks (Oklahoma, Arkansas, Louisiana, and Texas), with the best performance observed when using the LFPC operator. The key highlights include,

- **Mean GAP ± Standard Deviation (MGAP ± STD (%)):-** The POS algorithm produced lower MGAP values than NSGA-II and NSGA-III on the Arkansas and Texas networks, while NSGA-II performed better on the other two. NSGA-III consistently had the worst performance across all networks, indicating that the POS algorithm and NSGA-II converged more accurately to the optimal solutions compared to NSGA-III.

- **Mean Uniform Distribution ± Standard Deviation (MUD ± STD):-** NSGA-III excelled in generating uniformly distributed solutions across the Pareto front, outperforming both NSGA-II and the POS algorithm. This advantage is attributed to NSGA-III ability to maintain a diverse (less overlapping) set of solutions using

predefined reference points in the objective space. Across all four networks, the proposed POS approach also demonstrated better or comparable UD compared to NSGA-II.

- **Mean Cross-Population Ratio of Non-Dominated Individuals ± Standard Deviation (McpRNI ± STD):-** The POS algorithm outperformed both NSGA-II and NSGA-III in generating a higher proportion of non-dominated solutions in 3 out of the 4 networks, demonstrating its effectiveness in generating a better solution set of non-dominated individuals.

- **Mean Hypervolume ± Standard Deviation (MHV ± STD):-** The POS algorithm frequently achieved higher hypervolume values, particularly with the LFPC operator, indicating a better approximation of the Pareto front and capturing a greater hypervolume space. Figure 5.6 presents the variation in average hypervolume across 10 generations for NSGA-II, NSGA-III, and the POS algorithm. The variation across all four networks is shown, with POS algorithm capturing a higher average HV at generation 10 on each network. NSGA-III demonstrated the poorest performance among the three algorithms under evaluation, attributed to its limited convergence capabilities (Cui et al. 2019). To further examine the slow convergence of NSGA-III, 50 additional tests were conducted on the Oklahoma Network, with the algorithms initialised using 50 generations instead of 10. The variation in average hypervolume for NSGA-II, NSGA-III, and the POS algorithm across 50 generations on the Oklahoma Network is illustrated in Figure 5.7, clearly showing slower convergence in NSGA-III compared to both NSGA-II and the POS algorithm.

- **Mean Unique Solutions ± Standard Deviation (MUS ± STD (%)):-** NSGA-III consistently outperform NSGA-II and the proposed POS approach in terms of generating more unique solutions on all network, highlighting its ability to explore the solution space effectively. However, this increased diversity came at the expense of very slow convergence, as evident in the hypervolume plot in Figures 5.6 and 5.7.

- **Mean Computation Time ± Standard Deviation (MCT ± STD (sec)):-** Regarding computation time, NSGA-II had a slight advantage over the proposed POS approach when using the LFPC operator. However, the superior solution quality of the POS algorithm often justified the additional computational effort.

Overall Summary: The POS algorithm using the LFPC operator demonstrated strong performance across all networks. It excelled in achieving lower MGAP values, higher MHV, and higher McpRNI on most networks. Notably, the POS algorithm with the LFPC operator achieved the highest hypervolume (MHV) across all networks, reflecting its effectiveness in capturing better Pareto fronts. However, this advantage came with increased computation time compared to other configurations. Despite this, the POS algorithm proved to be a robust alternative to NSGA-II and NSGA-III, especially in large and complex networks, consistently delivering high-quality results while maintaining reasonable computational efficiency.

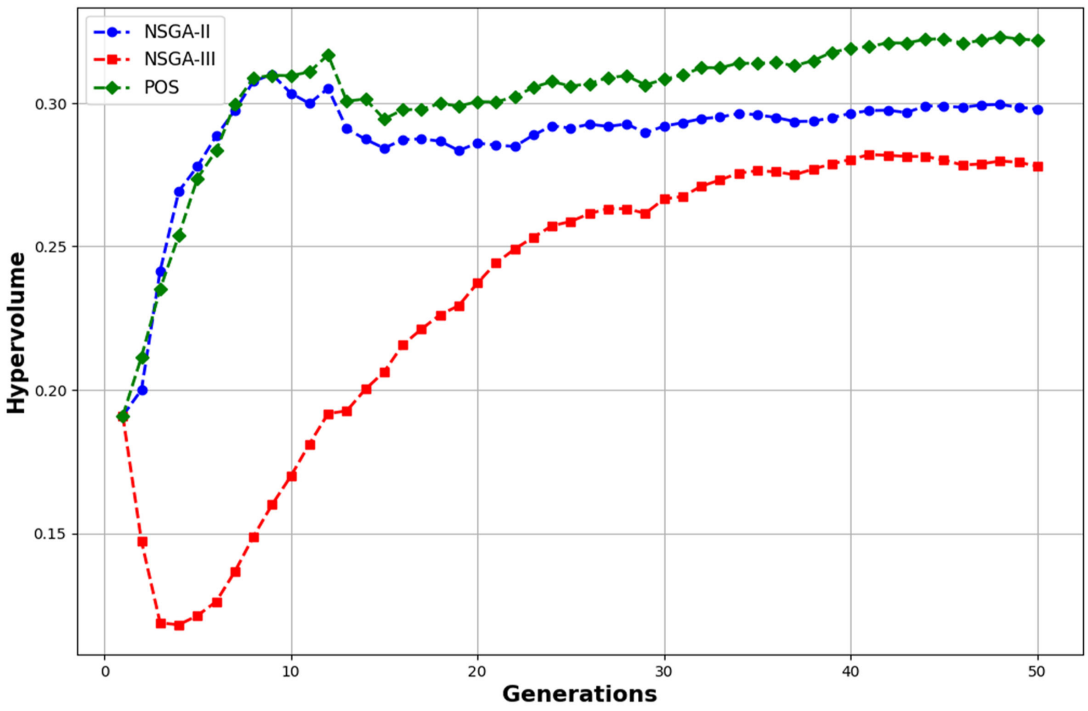

Figure 5.7: Variation in average hypervolume across 50 generations is examined for the simulations of NSGA-II, NSGA-III, and the POS algorithm proposed in our study. This analysis aggregates hypervolume data from the simulations of each algorithm, using both the NBCPM and LFPC operator across 50 test cases on the Oklahoma network, relative to the number of generations. In this plot, the generation count is plotted on the x-axis, while the captured hypervolume is plotted on the y-axis.

5.5.3 Statistical analysis

Next, we discuss the significance of our results, examining the pros and cons of the genetic operators and algorithms explored in this work, with a focus on their effectiveness in solving the multi-criteria Vehicle Routing Problem (VRP). The results from all six simulations (two each for NSGA-II, NSGA-III, and POS) were summarised in Table 5.3, highlighting their performance across four evaluation metrics (as described in Section 5.3). Additionally computational budget, measured in terms of computation time and the number of unique solutions (fitness evaluations) across four real-world road networks (outlined in Section 3.4), was also considered in the analysis.

We employed Friedman's chi-square test (Friedman 1937) alongside the Nemenyi post-hoc test to evaluate the significance of our results. The Friedman test was conducted using the Python library SciPy (Virtanen et al. 2021), while the Nemenyi post-hoc test was performed with the Python library scikit (Terpilowski 2019). The tests were conducted with a significance level of 0.05. The results for these tests across the four networks are available in the Jupyter notebooks on the GitHub profile: POS. The key highlights from these tests are summarised as follows:

- The results between the simulations using the NBCPM and LFPC operator were mostly significant across the evaluation metrics in all networks in our study.

- The results from the POS simulation using the LFPC operator on the GAP (%), HV, Unique Solutions, and cpRNI evaluation metrics were significant across all networks when compared to both NSGA-III simulations. The results for UD were significant only on the Louisiana network.
- The results from the NSGA-II simulation using the LFPC operator on the GAP (%), HV, Unique Solutions, and cpRNI evaluation metrics were significant across all networks when compared to NSGA-III simulations.
- The results between the POS simulation using the LFPC operator and the NSGA-II simulation using the LFPC operator were generally insignificant, except for computation time.

The analysis reveals that the choice of operator between NBCPM versus LFPC significantly impacts the performance across most evaluation metrics in all networks. This suggests that the LFPC operator generally offers a more effective approach in optimising the Vehicle Routing Problem (VRP) across different scenarios. The NSGA-II and POS algorithm combined with the LFPC operator consistently outperformed the NSGA-III simulations in terms of key metrics like GAP (%), HV, and cpRNI across all networks. However, NSGA-III had a significant advantage over NSGA-II and the POS algorithm in generating a higher number of unique solutions. When directly comparing POS and NSGA-II using the LFPC operator, the results were generally similar, with no significant differences in most metrics.

5.6 Impact of our proposed fitness metric

In the current world, navigating frequently requires balancing multiple objectives. Innovative strategies have been developed for path routing to address the difficulty of both minimising travel length and maximising encountered amenities. However, a traditional approach encountered a significant hurdle that longer roads naturally had more amenities, which reduced the effectiveness of the overall system.

Our investigation into a more complex strategy was prompted by the contradiction between path length and amenity count. Our aim was to create a system that maximised the amount of facilities experienced while also taking into account their location and significance along the route. This was no easy task since finding the ideal balance among the inherent trade-off was challenging. To address this challenge, we introduced a fitness metric called Positional Count described in Section 5.2.1. This metric assigns each amenity a dynamic weight based on where it is located and how densely it is distributed along the path. By accounting for these factors, our fitness metric goes beyond a basic count and tries to maximise the positional count of amenities, leading to more insightful results.

To demonstrate the efficacy of our fitness metric, the optimal solutions generated with respect to the Positional Count of Pubs, Hotels, and Charging stations for each test case were examined. It is note worthy that an optimal solution for a specific amenity is determined by the highest Positional Count value observed among all algorithmic simulations

in this work. Therefore, an optimal solution could originate from any of the algorithmic simulations conducted in this work. Then, we evaluate the path length of those optimal paths and compare them with the shortest paths produced by Dijkstras algorithm as the bench mark. The average error and standard deviation (in percentage) observed across the three amenity objectives Pubs, Hotels, and Charging stations is 36.25 $\pm$ 28.64, 38.56 $\pm$ 51.90, and 33.81 $\pm$ 23.90, respectively. These numbers, while showcasing the inherent complexity of the problem, also underscore the significant advancements our fitness metric has brought to the table. Given these results, we believe that further enhancing the precision of our fitness metric to reduce these errors would serve as a natural extension of this pioneering work. This endeavour promises to improve the optimisation procedure and make a substantial contribution to the area by creating a more reliable framework for multi-criteria vehicle routing solutions.

5.7 Practical applications of POS

In this chapter, the Parallel Optimal-route Search (POS) algorithm is introduced as a multi-criteria vehicle routing method on real-world road networks. The proposed method offers valuable advancements in vehicle routing, alternate routing, and multi-objective intelligent transportation systems. The key takeaways are as follows:

- **Evaluation metric for vehicle routing:-** As mentioned earlier, longer routes naturally encounter more amenities. Therefore, we propose a fitness metric that minimises path length while maximising the occurrence of amenities along the route. This metric balances the trade-off between travel distance and key attributes such as Pubs, Hotels, Charging stations, and more. This balance is particularly useful for problems like Tourist Trip Design (Choachaicharoenkul et al. 2022), which aim to maximise tourist destination visits while minimising travel distance.

- **Enhanced decision making for route selection:-** Conventional routing systems generally optimise paths based on length and/or travel but often overlook other important attributes, such as amenities, tourist destinations, and additional factors, when determining optimal routes. In contrast, POS efficiently generates multiple paths optimised across multiple objectives, not only minimising path length but also incorporating other important attributes, providing users with a diverse (less overlapping) set of route options. This makes the approach attractive for smart city planning, intelligent transportation systems, Geographic Information Systems (GIS) based navigation systems, and others.

- **Implications for traffic management and sustainability:-** like our MultiPath Island-Based Genetic Algorithm (MIBGA), POS returns multiple optimal paths and supports traffic decongestion strategies by distributing vehicles across alternative routes rather than concentrating them on a single path. This has direct implications such as, reducing traffic congestion, fuel consumption, and CO2 emissions, contributing to sustainable development.

- **Optimisation strategies for Parallel Genetic Algorithms:-** The proposed hybrid framework incorporates global parallelisation and island-based evolutionary strategies to accelerate convergence while preserving solution diversity. Beyond transportation, this approach can be applied to other single or multi-objective optimisation problems where faster convergence is essential.

Overall, our work highlights the practical significance of POS for modern transportation systems, demonstrating that GAs can be effectively employed to design advanced multi-criteria routing frameworks that enhance route planning and operational efficiency in complex real-world networks.

5.8 Chapter summary

This chapter presents the design, implementation, and evaluation of our Parallel Optimal-route Search (POS) for multi-criteria vehicle routing on real-world road networks. The proposed algorithm seeks to upgrade the current path routing methods by allowing the end user to select their favourite or preferred path from a set of multiple paths optimised on multiple objectives. The proposed algorithm incorporates parallel computing techniques through a hybrid model utilising Global Parallelisation and Island-Based approaches. The algorithm aims to optimise path routing by considering four objectives: minimising path length while maximising the Positional Count of Pubs, Hotels, and Charging stations. The experiments were designed to formulate a comparative study between the proposed POS, NSGA-II and NSGA-III, as well as between the proposed LFPC and the conventional NBCPM operators.

The results in Table 5.3, highlighted the superiority of our proposed LFPC operator against the traditional NBCPM operators in terms of its ability to generate solutions with better quality, capturing higher Hypervolume (HV) space, following better uniform distribution across the Pareto front. Another major advantage of employing the LFPC operator over the NBCPM operators is its ability to explore a wider range of solution possibilities by maintaining diverse (less overlapping) set of solutions throughout the algorithm. This was clearly demonstrated using Mean Unique Solutions (MUS) in Table 5.3, where simulations employing the LFPC operator had an average 93.61% unique solutions test case in contrast to just 67.14% for the simulations utilising the NBCPM operators. The significance of results was confirmed using the Friedman's chi-square (Friedman 1937) and Nemenyi post-hoc tests.

We also compared the performance of our POS algorithm with the well-known Non-Dominated Sorting Genetic Algorithms NSGA-II and NSGA-III. To formulate this comparative study, the results obtained from the two simulations of NSGA-II, NSGA-III, and the proposed POS algorithm were compared against each other. It is noteworthy that the comparison were made between the simulations using the same genetic operators. The results presented in Table 5.3, highlight the superior performance of the POS and NSGA-II algorithms using the LFPC operator compared to NSGA-III with the LFPC operator

across the majority of evaluation metrics, and this was majorly due to the poor convergence capabilities of NSGA-III (Cui et al. 2019), evident in Figure 5.7. The significance of results was confirmed using the Friedman's chi-square (Friedman 1937) and Nemenyi post-hoc tests.

The POS simulation using the LFPC operator outperformed the NSGA-II simulation with the same operator in terms of MUS and HV, though NSGA-II had slightly better computation time. The results on other metrics were generally similar and insignificant for most metrics. The results presented in this study offer valuable insights into the comparative performance of the POS algorithm, NSGA-II, and NSGA-III for the VRP. The fast convergence of NSGA-II underscores its popularity as a preferred algorithm for multi-objective VRP (Garside et al. 2024).

Taken together, the experimental findings demonstrate that the proposed fitness formulation effectively combines spatial information and amenity-based objectives within a mathematically grounded optimisation framework, addressing the core research objectives introduced in Chapter 1. In addition, the strong performance of the POS algorithm—particularly when paired with the LFPC operator shows that a parallel, island-based MOGA can serve as an efficient and scalable solution for multi-criteria vehicle routing on real-world networks. Overall, the POS algorithm emerged as a strong alternative to existing multi-objective routing methods, demonstrating its effectiveness across a wide range of networks. It consistently produced high-quality solutions while maintaining reasonable computational efficiency, highlighting its robustness and versatility.

Next, in **Chapter 6**, we summarise our findings and elaborate on the application of Evolutionary Algorithms (EAs) for path routing. We also discuss the challenges and potential avenues for enhancing alternate and multi-criteria routing systems, based on the insights gained from our experiments.

Chapter 6

Conclusions and Future work

6.1 Thesis overview

This thesis presents routing frameworks designed to generate diverse (less overlapping) alternative paths/routes that adapt to varying user needs and situational contexts. Contributing to the field of Intelligent Transportation Systems, these frameworks demonstrate how population based algorithms, such as Genetic Algorithms (GAs) can be effectively leveraged to solve complex routing scenarios involving multiple and often conflicting objectives. To evaluate the effectiveness of the proposed frameworks, we employed multiple real-world graph networks, as detailed in Chapter 3. The MultiPath Island-Based Genetic Algorithm (MIBGA) detailed in Chapter 4, was introduced to demonstrate the capability of Parallel Genetic Algorithms (PGAs) in generating a diverse set of near-optimal paths for the K-Most Diverse Near-Shortest Paths (KMDNSP) (Häcker et al. 2021) problem. Experimental findings underscored MIBGAs competitive convergence speed and improved scalability across multiple complex graph networks. Following the promising results achieved with MIBGA, we introduced our Parallel Optimal-route Search (POS) algorithm in Chapter 5, incorporating three additional objectives to enable a flexible, multi-criteria routing mechanism. The performance of MIBGA is evaluated against well-established GAs for routing introduced in (Ahn & Ramakrishna 2002, Qiongbing & Lixin 2016), while POS is compared with two of the most widely used multi-objective GAs, NSGA-II (Deb et al. 2002) and NSGA-III (Deb & Jain 2013). Chapter 6 is the final chapter in this thesis and it provides a summary of the key contributions of our work (section 6.2, a discussion of the computational performance of the developed approaches (section 6.3) and the computational challenges encountered in the research (section 6.4). We identify some limitations of the work in Section 6.5 before providing our overall conclusions and outline of future work in the final section of the thesis (section 6.6).

6.2 Key contributions from this research

This thesis **makes several significant contributions to the field of routing systems, with a particular emphasis on PGAs, a specialised class of Evolutionary**

Algorithms (EAs). It presents novel optimisation frameworks that incorporated new operators and strategies designed to address distinct NP-hard problems, with a strong emphasis on enhancing convergence speed and scalability. The main contributions are summarised below:

- **Developed a geospatial data pipeline that transformed ESRI Shapefiles into an edge list format suitable for graphical networks, enriched with computed edge attributes**. The workflow included data gathering from reliable open-source sources, feature selection, spatial data preprocessing, edge attribute computation, and graph construction. The complete pipeline and its applications are detailed in a **peer-reviewed journal publication Sharma et al. (2023)**.

- **Created the GeoGraphNetworks data repository, which provided comprehensive and publicly accessible spatial network resources, including road and rail networks of the United States (57 networks) and road and river networks of Great Britain (53 networks)**. The repository offered 110 networks in total, enabling detailed network representations across multilingual programming environments. The datasets are hosted on Figshare at GeoGraphNetworks, and the accompanying Python code for generating graph structures using NetworkX is available on GitHub at GeoGraphNetworks. Visual previews of each network were also included to illustrate their scale and structure.

- **Developed the MIBGA to solve the KMDNSP (Häcker et al. 2021) problem, with the aim of providing end users with a diverse (less overlapping) set of near-optimal paths**. The approach leveraged the inherent diversity preserving capabilities of EAs and enhanced convergence by introducing a selection threshold parameter to promote elitism. To support the effectiveness of MIBGA, several new operators and strategies were designed and integrated into the algorithm. These components were specifically tailored to enhance solution diversity, maintain feasibility, and accelerate convergence. The key strategies and operators are outlined below, and the complete methodology and evaluation of MIBGA have been presented in a **peer-reviewed journal publication Sharma et al. (2025a)**.

  - **Island formation strategy:** As described in Chapter 4.3.4, this strategy partitioned each islands population into two segments, a superior population (high fitness group) and a centralised population (average fitness group). Greater opportunities of generating offspring were given to candidates from the superior population, thereby encouraging the propagation of high quality solutions while maintaining genetic diversity.
  - **Migration strategy:** As described in Chapter 4.3.5, this strategy promoted and maintained diversity within the algorithm by exchanging superior population solutions between two randomly selected islands at each generation, prior to generating new solutions for fitness evaluation.

- **Loop-Free Path-Composer (LFPC):** As described in Chapter 4.3.6, two variants of the LFPC operator were introduced, the first using only crossover, and the second integrating mutation with crossover. However, as emphasised throughout this thesis, the LFPC variant using only crossover is predominantly employed, while the version integrating mutation is used to generate only a small number of candidates. This operator was introduced to overcome the limitation of the Node Based Crossover (NBC), as detailed in Chapter 2.7.4, which required a common node to perform a valid crossover operation.
- **AvgIslandFit (A multi-step selection process):** Because the KMDNSP problem prioritised shorter paths to satisfy the primary criterion of near-optimality, this operator (described in Chapter 4.3.9) was introduced to preserve the best solutions from each island and reduce the risk of losing high quality solutions during the evolutionary process.

- **Developed the POS, a PGA, that followed a hybrid model combining global parallelisation with the island model approach, to address multi-criteria vehicle routing by providing users with a set of optimal paths tailored to multiple objectives.** These objectives were specifically designed to develop a vehicle routing system suitable for the general public. This was supported by a novel fitness metric,

  - **Positional Count (PC) fitness metric:** As described in Chapter 5.2.1, the metric was designed to maximise the number of amenities encountered, while also considering their location and contextual significance along the route. Each amenity is assigned a weight based on its location and density, allowing the metric to prioritise strategically placed amenities. This addressed a major shortcoming of traditional methods, where longer paths were unfairly favoured due to the higher likelihood of encountering more features.

  Non-dominated Sorting with Improved Crowding Distance (ICD) (Yue et al. 2021) was applied to the centralised population in the global parallelisation model, while new solutions for evaluation were generated using the island model approach, as described in Chapter 5.2.2. The full methodology and evaluation of POS have been presented in a **peer-reviewed journal publication Sharma et al. (2025b)**.

- **Cross-Population Ratio of Non-Dominated Individuals (cpRNI):** As described in Chapter 5.3, cpRNI extends the classical RNI metric to enable fair cross-algorithm comparative evaluation by assessing all candidate solutions within a shared non-dominated population, thereby eliminating the population-specific bias present in traditional RNI method.

- **Demonstrated the scalability and convergence performance of the proposed algorithms** across multiple real-world transportation networks, as detailed in Chapters 4.5 and 5.5.

6.3 Comprehensive empirical insights from key contributions

This section presents an empirical analysis of the key contributions described in the previous section, focusing on their performance across multiple evaluation criteria and use cases.

- **GeoGraphNetworks:** The generated networks spanned a wide range of sizes, with multiple instances surpassing $100,000$ unique nodes and edges. The most extensive, the TQ RoadLink network from Great Britain (GB), featured $373,079$ nodes and $445,192$ edges underscoring the methods robustness at scale. Formal validation of these networks (detailed in Chapter 3.5) was performed using ESRI Shapefiles containing junction geometries officially defined by Ordnance Survey (OS). This ensured a higher level of reliability and structural accuracy.

- **LFPC:** In Chapter 5.5.1, we demonstrated the superiority of the proposed LFPC operator over the traditional NBC and Path Mutation (PM) operators collectively referred to as NBCPM. The key advantage of LFPC was in its ability to maintain diversity throughout the algorithm, enabling it to explore a broader range of the solution space. This advantage was reflected in the Mean Unique Solutions (MUS) metric reported in Table 5.3, where LFPC based simulations achieved an average of 93.61% unique solutions, compared to just 67.14% for those using NBCPM. The statistical significance of these results was confirmed using Friedmans chi-square test (Friedman 1937), followed by the Nemenyi post-hoc test.

- **PC fitness metric:** When we compared the lengths of the paths produced by Dijkstra's algorithm (used as a benchmark) with those of the optimal paths generated using the PC of pubs, hotels, and charging stations, the average percentage deviations and standard deviations were 36.25 ± 28.64, 38.56 ± 51.90, and 33.81 ± 23.90, respectively. These deviations highlighted the complexity of this problem and also underscored the ability of our fitness metric to generate solutions that balanced path length with amenity relevance.

- **MIBGA:** As shown in Chapter 4.5, MIBGA consistently outperformed the GAs by Ahn & Ramakrishna (2002) and Qiongbing & Lixin (2016) across multiple evaluations,

  - In Chapter 4.5.1, MIBGA achieved higher non-timeout instances on all networks, specifically with the smaller ϵ values $(0.01-0.10)$, as shown in Figure 4.7.
  - In Chapter 4.5.2, MIBGA generated significantly more diverse (less overlapping) path sets. This was confirmed using the Wilcoxon signed-rank test with a Bonferroni-corrected p-value of 0.0025 and effect size based on Cohen (2013) method.

  - In Chapter 4.5.3, MIBGA generally outperformed both GAs on the Arizona and Washington networks with increasing K, while the GA by Ahn & Ramakrishna (2002) performed best on the Kansas network. For smaller ϵ values, MIBGA matched or exceeded the others, whereas the GA by Ahn & Ramakrishna (2002) showed slightly better performance at larger ϵ values. The GA by Qiongbing & Lixin (2016) had the longest runtimes, indicating limited scalability.

- **POS:** As discussed in Chapter 5.5.1, simulations using the LFPC operator consistently outperformed those based on the NBCPM operator. However, when comparing the POS, NSGA-II, and NSGA-III algorithms in Chapter 5.5.2, where each employed the LFPC operator

  - The POS and NSGA-II demonstrated superior performance across most evaluation metrics. This was primarily due to the slow convergence behaviour of NSGA-III (Cui et al. 2019), as shown in Figure 5.7.
  - The POS algorithm outperformed both NSGA-II and NSGA-III in generating superior non-dominated solution sets across three of the four networks (Arkansas, Louisiana, and Texas), as evidenced by higher cpRNI values.
  - POS outperformed NSGA-II in terms of Mean Unique Solutions (MUS) and Mean Hypervolume (MHV), although NSGA-II had slightly better computation time. Differences in other metrics were generally minor and statistically insignificant.

6.4 Description of auxiliary computational challenges in this research

As mentioned at the beginning of this chapter, the research has been very successful and has delivered several key contributions. However, it is worth noting that during the work several auxiliary or tangential challenges presented situations where we needed to think very carefully about solution design or measurement of the quality or performance of the developed approaches. We list several of the most pressing challenges we encountered as follows:

- **Network Selection for Algorithm Evaluation:** A key challenge in the experimental design was selecting appropriate types of networks to effectively demonstrate the performance of our GAs. As stated before, the time complexity of routing algorithms is directly influenced by the number of nodes and edges, so it was essential to use real-world networks that were complex enough to reflect practical scenarios, yet appropriately sized to allow feasible computation. Striking this balance required careful consideration. Additionally, we opted for undirected networks over directed ones to enable our GAs to more thoroughly explore the search space and uncover diverse (less overlapping) routing options, which is crucial for multi-path and multi-criteria routing.

- **Justifying the use of EAs in path routing:** While EAs offer flexibility and adaptability, they typically incur higher computational costs than traditional algorithms such as Dijkstras (EW 1959), largely due to the need to evaluate fitness values and manage evolving populations of solutions. However, given that the problem is NP-hard, EAs provide a viable approach where exact methods fall short. Therefore, one major challenge was to effectively demonstrate the key advantages of EAs such as their ability to generate and optimise multiple solutions simultaneously, produce diverse path sets, and handle multiple objectives, despite these higher computational demands.

- **Balancing solution quality with computational efficiency:** Given the nature of vehicle routing applications, fast path retrieval is often as critical as retrieving the most optimal path. While exhaustive methods like grid search can eventually identify the best possible solution, they are often computationally infeasible for large-scale or real-time applications. In contrast, EAs strike a balance between computational speed and solution quality. Therefore, alongside maintaining population diversity, special attention was paid to promoting faster convergence in our GAs to ensure timely yet high quality results.

- **Parameter tuning in MIBGA and POS:** As discussed in Chapters 4.4 and 5.4, both MIBGA and POS required careful tuning of additional parameters that significantly influenced performance. For example, the selection threshold, used to promote elitism, had to be balanced carefully because values too low could led to premature convergence, while overly high values reduced selection pressure. Similarly, the minimum island size ensured that each island retained enough solutions to generate new candidates for evaluation. Determining effective parameter settings was a non-trivial task and posed a substantial implementation challenge.

- **Difficulty in demonstrating statistical significance in path routing:** EAs are metaheuristic algorithms, and their performance is commonly validated using statistical significance testing. However, in the context of path routing, demonstrating statistical significance can be challenging. For instance, an algorithm may consistently produce paths that are slightly shorter (for example: 0.1 or 0.2 km), which is a significant improvement in real-world situations. However, such small absolute differences may not be statistically significant, particularly when working with a limited number of test instances. This highlights a gap between practical value and statistical validation in routing applications.

6.5 Identified Limitations

It is both expected and acceptable for a research project of this scale and complexity to have some limitations. While the methods proposed in this work have shown significant and impactful results, certain limitations have been identified that may affect the broader

applicability or performance of these methods in specific scenarios. These often reflect the inherent challenges of this type of real-world research rather than shortcomings in the research design or execution. Recognising these constraints is essential for interpreting the results accurately and guiding future research. We identify some key limitations as follows:

- **Static network assumptions:** The performances of MIBGA and POS were evaluated under the assumption of static road networks without any dynamic variables, primarily due to computational limitations. These networks did not account for real-world dynamics such as traffic signals, congestion, or temporal variations. As a result, the effectiveness of the algorithms under real-time constraints, remains an open area for further investigation.

- **Sensitivity to small ϵ values:** As discussed in Chapter 4.5.1, the current version of MIBGA performed well for ϵ values greater than 0.10, with fewer than 5% timeout instances. However, for smaller ϵ values $(0.01 - 0.10)$, the proportion of timeout instances increased noticeably. This suggests MIBGAs limited efficiency under stricter constraints, and points to the need for further optimisation, potentially through integration with heuristic methods or adaptive strategies.

- **Computational overhead of the LFPC operator:** As stated in Chapter 5.5.1, the LFPC operator consistently outperformed the NBCPM operator by generating more unique and diverse solutions, thereby enhancing exploration. However, its reliance on repeated calls to the path, initialisation method resulted in increased computational cost. This was evident in Table 5.3, where simulations using LFPC had higher Mean Computation Time (MCT). A potential improvement would be to use the NBCPM operator during the initial generations and switch to LFPC once the search begins producing duplicate paths.

- **Generalised assumptions in positional weighting:** The positional weighting strategy assumed general user preferences, such as favouring amenities located earlier in the route or in less dense areas. However, this may not reflect all real-world scenarios. For instance, some users may prefer to access certain amenities only after travelling a specific distance, such as stopping for a meal halfway through a long trip or looking for a hotel after covering a substantial portion of their journey. Accounting for such personalised preferences remains an area for future refinement.

- **Trade-off between optimality and desirable features:** Although the PC fitness metric significantly improved over traditional count based strategies by incorporating desirable features (such as amenities used in this thesis), the resulting paths still showed considerable deviation from the shortest paths, averaging 33-38%. In applications where strict adherence to shortest routes is critical (such as emergency routing), these deviations may be unacceptable. Therefore, further tuning or hybridisation with distance sensitive objectives may be necessary to reduce this gap.

6.6 Conclusions and Future Work

To conclude, this work presented novel algorithms and metrics for addressing complex routing problems, and the results achieved are both promising and impactful. The key contributions of this research outlined as follows:

- **This work demonstrated that Island-Based Genetic Algorithms (IBGAs), a specialised class of EAs are highly effective in solving the complex KMDNSP problem efficiently.** The multi-island structure of IBGAs proved particularly well suited for applications that require the generation of multiple diverse (less overlapping) and optimal paths, reinforcing their practical value in real-world vehicle routing systems.

- **This work demonstrated how spatial information, including geographic coordinates and distances, can be used to mathematically formulate a fitness metric that balances path length and amenity exposure.** The proposed PC fitness metric incorporates both spatial proximity and geographic distribution of amenities into the optimisation process. By assigning higher weights to amenities located in less dense areas and by prioritising those closer to the starting point, the metric penalises unnecessarily long paths while still encouraging relevant amenity encounters. This formulation addresses the limitations of traditional count based approaches by shifting the focus from the sheer number of Points of Interest (POIs) to their spatial relevance and contextual importance of features along the route.

- **This work demonstrated that Multi-Objective Genetic Algorithms (MOGAs) can be effectively employed to address multi-criteria VRPs on large and complex real-world networks, as evidenced by the performance of the proposed POS algorithm.** The proposed POS algorithm offers a computationally effective alternative to NSGA II and III (Deb et al. 2002, Deb & Jain 2013), performing particularly well on large and complex networks. It consistently generated high quality solutions with reasonable computational demands, highlighting its applicability to real-world routing problems.

- **This work demonstrated that the proposed PGAs achieve faster convergence and greater scalability by computationally outperforming several state-of-the-art GAs on the selected routing problems.** The MIBGA was benchmarked against the approaches of Ahn & Ramakrishna (2002) and Qiongbing & Lixin (2016), showing superior convergence behaviour and reduced execution time across multiple networks. Similarly, the POS algorithm was compared against NSGA-II and NSGA-III (Deb et al. 2002, Deb & Jain 2013), consistently producing high-quality solutions with lower computational demands. Together, these results confirm the scalability and efficiency advantages of the introduced PGAs.

- **To support reproducibility and encourage further research, the GeoGraph-Networks data repository was developed as part of this work.** It contains

all the processed road network data along with the necessary code to generate and visualise the networks used throughout this study, making it a valuable resource for future experimentation and benchmarking.

Opportunities for future work

Despite the demonstrated effectiveness of the proposed methods and metrics, several promising directions remain for further exploration. Building on the findings and challenges discussed above, the following future work avenues are suggested:

- **Hybridisation:** Future work could explore the potential of combining different algorithms and strategies to leverage their respective strengths, such as
  - **Integrating MIBGA with heuristic approaches:** As a potential enhancement for solving the KMDNSP problem, heuristic methods could be employed for smaller ϵ values $(0.01 - 0.10)$, where MIBGA currently struggles with timeout instances. As ϵ increases, control could be gradually shifted to MIBGA, which has demonstrated faster convergence and improved runtime performance at higher ϵ values, as detailed in Chapter 4.5.3. This hybrid strategy could significantly improve scalability and efficiency across varying problem constraints.
  - **Integrating MIBGA with POS:** Several integration strategies could be explored to combine the strengths of MIBGA and POS. One approach involves incorporating the dissimilarity metric from MIBGA as an additional objective in the POS routing system, alongside the existing length threshold criteria. Another promising direction is to embed components of MIBGA such as the migration strategy described in Chapter 4.3.5 into the POS framework to enhance its exploration capabilities. Such integration could lead to more robust and diverse (less overlapping) path generation.
  - **Using both LFPC and NBCPM operators within a single GA:** Develop adaptive mechanisms to alternate or combine these operators during evolution for a better balance of exploration and exploitation.
- **Refinement of the PC fitness metric for different application domain:** The PC fitness metric holds significant potential for adaptation and refinement across diverse application domains. By tailoring this metric to specific contexts, domain relevant preferences and constraints can be incorporated, thereby enhancing the relevance and quality of the generated solutions. Potential domains for such refinements include,
  - **Tourist Trip Design Problem (TTDP):** The PC metric could be adapted to TTDP by aiming to minimise travel distance or time spent on the road while maximising the exposure to valuable tourist attractions. Attractions may be classified into categories (Ruiz-Meza & Montoya-Torres 2022) including historical sites, museums, parks, and others, with weights assigned according to

characteristics such as path length and user generated ratings. This would help tailor route recommendations not only to include more attractions, but to emphasise those that are timely, relevant, and highly rated by users.

  - **University campus shared mobility systems:** One promising application of the PC fitness metric lies in designing personalised campus exploration routes for new students. By assigning higher weights to buildings based on their relevance to a student's academic program or orientation schedule, the system can generate personalised paths that prioritise visits to the most important facilities such as, department buildings, libraries, student centres, and others. An additional advantage is that orientation organisers can generate diverse sets of routes for different student groups, helping to distribute foot traffic more evenly across the campus. This not only enhances the tour experience but also reduces the likelihood of overcrowding in specific campus zones, which is particularly beneficial during large-scale orientation events involving multiple academic programs.
  - **Disaster relief logistics systems:** In disaster scenarios, the PC fitness metric can be leveraged to allocate emergency response teams to distinct evacuation sites via separate, efficient routes. By prioritising specific areas such as, densely populated or vulnerable zones, the approach enables effective coverage and route diversity. This helps minimise congestion and ensures smoother traffic flow for both emergency responders and the general public.

- **Personalised dynamic routing systems:** Future extensions could focus on enhancing the adaptability of the routing framework by incorporating user-specific preferences alongside real-time contextual information. This involves integrating a level of personalisation, where users can select and prioritise routing criteria such as scenic value, road type, or amenity relevance based on their individual needs, with dynamic routing capabilities that respond to real-time factors like traffic conditions, road closures, or amenity availability. Together, these enhancements would enable highly responsive, context-aware navigation that adapts seamlessly to both user preferences and evolving real-world conditions. To realise this vision, several factors require careful consideration,

  - **What real-time information should be incorporated?:** The type and granularity of real-time information depend heavily on the specific application and user context. Some systems may only need to account for major disruptions such as road closures or accidents, while others could benefit from incorporating fine-grained updates like minor delays or temporary congestion to further improve route relevance and responsiveness.
  - **How should real-time contextual information be integrated into the routing framework?:** Determining the most effective method for integrating real-time information remains an open research challenge. However, a practical approach could involve penalising features located on affected roads. For

example, in the case of minor traffic delays, a small penalty could be applied to the corresponding edges, whereas in the event of serious accidents or closures, features along those routes could be assigned strongly negative weights, effectively discouraging their selection during route generation.

- **How can the framework adapt to changing real-time information?:** Adapting the routing framework to real-time information requires mechanisms for continuous monitoring and flexible recalibration of routes. Sudden disruptions, such as accidents or unexpected road closures pose particular challenges, as they may demand immediate re-routing with limited available alternatives. To handle such scenarios, the system must be capable of quickly identifying viable substitute paths, prioritising safety and feasibility, even if it occasionally requires overriding user preferences.

All the aforementioned future research directions could ultimately lead to practical deployment and real-world implementation of the proposed methods. Incorporating feedback from end users, such as transport planners, GIS professionals, and local authorities would further strengthen these methodologies by providing insights into usability, scalability, and operational feasibility. While the present validation primarily focused on technical and topological accuracy, future work that combines these developments with end-user evaluation would support the transition from experimental research to practical application, ensuring that the resulting frameworks are both scientifically robust and operationally relevant.

References

Adamo, T., Gendreau, M., Ghiani, G. & Guerriero, E. (2024), 'A review of recent advances in time-dependent vehicle routing', European Journal of Operational Research .

Ahn, C. W. & Ramakrishna, R. S. (2002), 'A genetic algorithm for shortest path routing problem and the sizing of populations', IEEE transactions on evolutionary computation **6**(6), 566–579.

Aibinu, A., Salau, H. B., Rahman, N. A., Nwohu, M. & Akachukwu, C. (2016), 'A novel clustering based genetic algorithm for route optimization', Engineering Science and Technology, an International Journal **19**(4), 2022–2034.

Al Theeb, N., Abu-Aleqa, M. & Diabat, A. (2024), 'Multi-objective optimization of two-echelon vehicle routing problem: Vaccines distribution as a case study', Computers & Industrial Engineering **187**, 109590.

Amgoth, T. & Jana, P. K. (2015), 'Energy-aware routing algorithm for wireless sensor networks', Computers & Electrical Engineering **41**, 357–367.

Anderson, E. J. & Ferris, M. C. (1990), A genetic algorithm for the assembly line balancing problems, Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Antonio, L. M. & Coello, C. A. C. (2013), Use of cooperative coevolution for solving large scale multiobjective optimization problems, in '2013 IEEE Congress on Evolutionary Computation', IEEE, pp. 2758–2765.

Barta, Z., Flynn, R. & Giraldeau, L.-A. (1997), 'Geometry for a selfish foraging group: a genetic algorithm approach', Proceedings of the Royal Society of London. Series B: Biological Sciences **264**(1385), 1233–1238.

Basiri, A., Amirian, P. & Mooney, P. (2016), 'Using crowdsourced trajectories for automated osm data entry approach', Sensors **16**(9), 1510.

Bellman, R. (1958), 'On a routing problem', Quarterly of applied mathematics **16**(1), 87–90.

Bergstra, J. & Bengio, Y. (2012), 'Random search for hyper-parameter optimization', The journal of machine learning research **13**(1), 281–305.

Bhardwaj, A. & El-Ocla, H. (2020), 'Multipath routing protocol using genetic algorithm in mobile ad hoc networks', IEEE Access **8**, 177534–177548.

Blank, J. & Deb, K. (2020), 'pymoo: Multi-objective optimization in python', IEEE Access **8**, 89497–89509.

Boeing, G. (2017), 'Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks', Computers, Environment and Urban Systems **65**, 126–139.

Bogyrbayeva, A., Meraliyev, M., Mustakhov, T. & Dauletbayev, B. (2024), 'Machine learning to solve vehicle routing problems: A survey', IEEE Transactions on Intelligent Transportation Systems **25**(6), 4754–4772.

Borgatti, S. P. & Everett, M. G. (2006), 'A graph-theoretic perspective on centrality', Social networks **28**(4), 466–484.

Bui, X.-N., Boissonnat, J.-D., Soueres, P. & Laumond, J.-P. (1994), Shortest path synthesis for dubins non-holonomic robot, in 'Proceedings of the 1994 IEEE International Conference on Robotics and Automation', IEEE, pp. 2–7.

Butts, C. T. (2008), 'network: a package for managing relational data in r', Journal of statistical software **24**, 1–36.

Cadger, F., Curran, K., Santos, J. & Moffett, S. (2012), 'A survey of geographical routing in wireless ad-hoc networks', IEEE Communications Surveys & Tutorials **15**(2), 621–653.

Cantú-Paz, E. et al. (1998), 'A survey of parallel genetic algorithms', Calculateurs paralleles, reseaux et systems repartis **10**(2), 141–171.

Chen, S. & Nahrstedt, K. (1998), 'An overview of quality of service routing for next-generation high-speed networks: problems and solutions', IEEE network **12**(6), 64–79.

Cheng, D., Gkountouna, O., Züfle, A., Pfoser, D. & Wenk, C. (2019), Shortest-path diversification through network penalization: A washington dc area case study, in 'Proceedings of the 12th ACM SIGSPATIAL International Workshop on Computational Transportation Science', pp. 1–10.

Chitra, C. & Subbaraj, P. (2012), 'A nondominated sorting genetic algorithm solution for shortest path routing problem in computer networks', Expert systems with applications **39**(1), 1518–1525.

Choachaicharoenkul, S., Coit, D. & Wattanapongsakorn, N. (2022), 'Multi-objective trip planning with solution ranking based on user preference and restaurant selection', IEEE Access **10**, 10688–10705.

Chondrogiannis, T., Bouros, P., Gamper, J. & Leser, U. (2015), Alternative routing: k-shortest paths with limited overlap, in 'Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems', pp. 1–4.

Coello, C. A. C., Lamont, G. B., Van Veldhuizen, D. A. et al. (2007), Evolutionary algorithms for solving multi-objective problems, Vol. 5, Springer.

Cohen, J. (2013), Statistical power analysis for the behavioral sciences, routledge.

Conceição, M. A., Monteiro, M. M., Kasraian, D., van den Berg, P., Haustein, S., Alves, I., Azevedo, C. L. & Miranda, B. (2023), 'The effect of transport infrastructure, congestion and reliability on mental wellbeing: a systematic review of empirical studies', Transport reviews **43**(2), 264–302.

Cordeau, J.-F., Laporte, G. & Mercier, A. (2001), 'A unified tabu search heuristic for vehicle routing problems with time windows', Journal of the Operational research society **52**(8), 928–936.

Cordeau, J.-F., Laporte, G., Savelsbergh, M. W. & Vigo, D. (2007), 'Vehicle routing', Handbooks in operations research and management science **14**, 367–428.

Csardi, G. & Nepusz, T. (2006), 'The igraph software', Complex syst **1695**, 1–9.

Csardi, M. G. (2013), 'Package 'igraph'', Last accessed **3**(09), 2013.

Cui, Z., Chang, Y., Zhang, J., Cai, X. & Zhang, W. (2019), 'Improved nsga-iii with selection-and-elimination operator', Swarm and Evolutionary Computation **49**, 23–33.

Dabia, S., Ropke, S., Van Woensel, T. & De Kok, T. (2013), 'Branch and price for the time-dependent vehicle routing problem with time windows', Transportation science **47**(3), 380–396.

Dantzig, G., Fulkerson, R. & Johnson, S. (1954), 'Solution of a large-scale traveling-salesman problem', Journal of the operations research society of America **2**(4), 393–410.

Darwish, S. M., Elmasry, A. & Ibrahim, S. H. (2020), Optimal shortest path in mobile ad-hoc network based on fruit fly optimization algorithm, in 'The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2019) 4', Springer, pp. 91–101.

Deb, K. & Jain, H. (2013), 'An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints', IEEE transactions on evolutionary computation **18**(4), 577–601.

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002), 'A fast and elitist multiobjective genetic algorithm: Nsga-ii', IEEE transactions on evolutionary computation **6**(2), 182–197.

Deep, K. & Mebrahtu, H. (2011), 'Combined mutation operators of genetic algorithm for the travelling salesman problem', International Journal of Combinatorial Optimization Problems and Informatics **2**(3), 1–23.

Del Ser, J., Osaba, E., Molina, D., Yang, X.-S., Salcedo-Sanz, S., Camacho, D., Das, S., Suganthan, P. N., Coello, C. A. C. & Herrera, F. (2019), 'Bio-inspired computation: Where we stand and what's next', Swarm and Evolutionary Computation **48**, 220–250.

Di Placido, A., Archetti, C. & Cerrone, C. (2022), 'A genetic algorithm for the close-enough traveling salesman problem with application to solar panels diagnostic reconnaissance', Computers & Operations Research **145**, 105831.

Dominico, G. & Parpinelli, R. S. (2021), 'Multiple global optima location using differential evolution, clustering, and local search', Applied Soft Computing **108**, 107448.

Donati, A. V., Montemanni, R., Casagrande, N., Rizzoli, A. E. & Gambardella, L. M. (2008), 'Time dependent vehicle routing problem with a multi ant colony system', European journal of operational research **185**(3), 1174–1191.

Dong-liang, L., Bei, L. & Hai-hua, W. (2023), 'The importance of nature-inspired meta-heuristic algorithms in the data routing and path finding problem in the internet of things', International Journal of Communication Systems **36**(10), e5450.

Eiben, A. & Smith, J. (2015), Introduction to Evolutionary Computing, Springer.

Eksioglu, B., Vural, A. V. & Reisman, A. (2009), 'The vehicle routing problem: A taxonomic review', Computers & Industrial Engineering **57**(4), 1472–1483.

Eppstein, D. (1998), 'Finding the k shortest paths', SIAM Journal on computing **28**(2), 652–673.

Erdoğan, S. & Miller-Hooks, E. (2012), 'A green vehicle routing problem', Transportation research part E: logistics and transportation review **48**(1), 100–114.

Euchi, J., Yassine, A. & Chabchoub, H. (2015), 'The dynamic vehicle routing problem: Solution with hybrid metaheuristic approach', Swarm and Evolutionary Computation **21**, 41–53.

Euler, L. (1741), 'Solutio problematis ad geometriam situs pertinentis', Commentarii academiae scientiarum Petropolitanae pp. 128–140.

EW, D. (1959), 'Anoteontwoproblems inconnexionwithgraphs', Numerische Mathemati **1**(1), 269–271.

Fey, M. & Lenssen, J. E. (2019), 'Fast graph representation learning with pytorch geometric', arXiv preprint arXiv:1903.02428 .

Folino, G. & Spezzano, G. (2006), P-cage: an environment for evolutionary computation in peer-to-peer systems, in 'European Conference on Genetic Programming', Springer, pp. 341–350.

Fonseca, C. M. & Fleming, P. J. (1998), 'Multiobjective optimization and multiple constraint handling with evolutionary algorithms. i. a unified formulation', IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans **28**(1), 26–37.

Fonseca, C. M., Paquete, L. & López-Ibánez, M. (2006), An improved dimension-sweep algorithm for the hypervolume indicator, in '2006 IEEE international conference on evolutionary computation', IEEE, pp. 1157–1163.

Fonte, C. C. & Martinho, N. (2017), 'Assessing the applicability of openstreetmap data to assist the validation of land use/land cover maps', International Journal of Geographical Information Science **31**(12), 2382–2400.

Forghani, M. & Delavar, M. R. (2014), 'A quality study of the openstreetmap dataset for tehran', ISPRS International Journal of Geo-Information **3**(2), 750–763.

Friedman, M. (1937), 'The use of ranks to avoid the assumption of normality implicit in the analysis of variance', Journal of the american statistical association **32**(200), 675–701.

Fritz, C. O., Morris, P. E. & Richler, J. J. (2012), 'Effect size estimates: current use, calculations, and interpretation.', Journal of experimental psychology: General **141**(1), 2.

Galván, E. & Schoenauer, M. (2019), Promoting semantic diversity in multi-objective genetic programming, in 'Proceedings of the Genetic and Evolutionary Computation Conference', GECCO '19, Association for Computing Machinery, New York, NY, USA, p. 1021–1029.
URL: *https://doi.org/10.1145/3321707.3321854*

Galván, E., Simpson, G. & Ameneyro, F. V. (2022), 'Evolving the mcts upper confidence bounds for trees using a semantic-inspired evolutionary algorithm in the game of carcassonne', IEEE Transactions on Games .

Galván, E., Trujillo, L. & Stapleton, F. (2022), 'Semantics in multi-objective genetic programming', Applied Soft Computing **115**, 108143.

Galván, E. & Stapleton, F. (2023), 'Evolutionary multi-objective optimisation in neurotrajectory prediction', Applied Soft Computing **146**, 110693.
URL: *https://www.sciencedirect.com/science/article/pii/S1568494623007111*

Garside, A. K., Ahmad, R. & Muhtazaruddin, M. N. B. (2024), 'A recent review of solution approaches for green vehicle routing problem and its variants', Operations Research Perspectives p. 100303.

Gavoille, C. (2001), 'Routing in distributed networks: Overview and open problems', ACM SIGACT News **32**(1), 36–52.

Gen, M., Cheng, R. & Wang, D. (1997), Genetic algorithms for solving shortest path problems, in 'Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)', IEEE, pp. 401–406.

Gen, M. & Lin, L. (2006), A new approach for shortest path routing problem by random key-based ga, in 'Proceedings of the 8th annual conference on genetic and evolutionary computation', pp. 1411–1412.

Gendreau, M., Ghiani, G. & Guerriero, E. (2015), 'Time-dependent routing problems: A review', Computers & operations research pp. 189–197.

Ghoseiri, K. & Ghannadpour, S. F. (2010), 'Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm', Applied Soft Computing **10**(4), 1096–1107.

Gillies, S. (2013), 'The shapely user manual', URL https://pypi. org/project/Shapely .

Gmira, M., Gendreau, M., Lodi, A. & Potvin, J.-Y. (2021), 'Tabu search for the time-dependent vehicle routing problem with time windows on a road network', European Journal of Operational Research **288**(1), 129–140.

Goldberg, D. E. (1989), Genetic Algorithms in Search, Optimization and Machine Learning, 1st edn, Addison-Wesley Longman Publishing Co., Inc., USA.

Goldberg, D. E., Richardson, J. et al. (1987), Genetic algorithms with sharing for multimodal function optimization, in 'Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms', Vol. 4149, Hillsdale, NJ: Lawrence Erlbaum, pp. 41–49.

Gonen, B. & Louis, S. J. (2006), 'Genetic algorithm finding the shortest path in networks', Reno: University of Nevada .

Gong, G., Deng, Q., Gong, X., Zhang, L., Wang, H. & Xie, H. (2018), 'A bee evolutionary algorithm for multiobjective vehicle routing problem with simultaneous pickup and delivery', Mathematical Problems in Engineering **2018**(1), 2571380.

Gong, Y.-J., Chen, W.-N., Zhan, Z.-H., Zhang, J., Li, Y., Zhang, Q. & Li, J.-J. (2015), 'Distributed evolutionary algorithms and their models: A survey of the state-of-the-art', Applied Soft Computing **34**, 286–300.

Google (n.d.), '[route from blackrock health hermitage clinic to maynooth university – google maps]', https://www.google.com/maps. Retrieved 12 June 2025.

Guck, J. W., Van Bemten, A., Reisslein, M. & Kellerer, W. (2017), 'Unicast qos routing algorithms for sdn: A comprehensive survey and performance evaluation', IEEE Communications Surveys & Tutorials **20**(1), 388–415.

Häcker, C., Bouros, P., Chondrogiannis, T. & Althaus, E. (2021), Most diverse near-shortest paths, in 'Proceedings of the 29th International Conference on Advances in Geographic Information Systems', pp. 229–239.

Hagberg, A., Swart, P. & S Chult, D. (2008), Exploring network structure, dynamics, and function using networkx, Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Hakeem, A., Gehani, N., Ding, X., Curtmola, R. & Borcea, C. (2019), Multi-destination vehicular route planning with parking and traffic constraints, in 'Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services', pp. 298–307.

Hamed, A. Y. (2010), 'A genetic algorithm for finding the k shortest paths in a network', Egyptian Informatics Journal **11**(2), 75–79.

Hanshar, F. T. & Ombuki-Berman, B. M. (2007), 'Dynamic vehicle routing using genetic algorithms', Applied Intelligence **27**, 89–99.

Harada, T. & Alba, E. (2020), 'Parallel genetic algorithms: a useful survey', ACM Computing Surveys (CSUR) **53**(4), 1–39.

Hart, P. E., Nilsson, N. J. & Raphael, B. (1968), 'A formal basis for the heuristic determination of minimum cost paths', IEEE transactions on Systems Science and Cybernetics **4**(2), 100–107.

Hasan, K. W., Ali, S. M., Paul, S. K. & Kabir, G. (2023), 'Multi-objective closed-loop green supply chain model with disruption risk', Applied soft computing **136**, 110074.

Heidari, A., Imani, D. M., Khalilzadeh, M. & Sarbazvatan, M. (2023), 'Green two-echelon closed and open location-routing problem: application of nsga-ii and mogwo metaheuristic approaches', Environment, Development and Sustainability **25**(9), 9163–9199.

Her, Y. G. & Yu, Z. (2021), 'Mapping the us census data using the tiger/line shapefiles: Ae557/ae557, 05/2021', EDIS **2021**(3).

Hidalgo, J. I. & Fernández, F. (2005), Balancing the computation effort in genetic algorithms, in '2005 IEEE Congress on Evolutionary Computation', Vol. 2, IEEE, pp. 1645–1652.

Horn, J., Nafpliotis, N. & Goldberg, D. E. (1994), A niched pareto genetic algorithm for multiobjective optimization, in 'Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence', Ieee, pp. 82–87.

Hunter, J. D. (2007), 'Matplotlib: A 2d graphics environment', Computing in science & engineering **9**(03), 90–95.

Ichoua, S., Gendreau, M. & Potvin, J.-Y. (2003), 'Vehicle dispatching with time-dependent travel times', European journal of operational research **144**(2), 379–396.

Inagaki, J., Haseyama, M. & Kitajima, H. (1999), A genetic algorithm for determining multiple routes and its applications, in '1999 IEEE International Symposium on Circuits and Systems (ISCAS)', Vol. 6, IEEE, pp. 137–140.

Jain, H. & Deb, K. (2013), 'An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach', IEEE Transactions on evolutionary computation **18**(4), 602–622.

Jebari, K. & Madiafi, M. (2013), 'Selection methods for genetic algorithms', International Journal of Emerging Sciences **3**(4), 333–344.

Jemai, J., Zekri, M. & Mellouli, K. (2012), An nsga-ii algorithm for the green vehicle routing problem, in 'Evolutionary Computation in Combinatorial Optimization: 12th European Conference, EvoCOP 2012, Málaga, Spain, April 11-13, 2012. Proceedings 12', Springer, pp. 37–48.

Ji, Z., Kim, Y. S. & Chen, A. (2011), 'Multi-objective α-reliable path finding in stochastic networks with correlated link costs: A simulation-based multi-objective genetic algorithm approach (smoga)', Expert Systems with Applications **38**(3), 1515–1528.

Jordahl, K., Van den Bossche, J., Wasserman, J., McBride, J., Gerard, J., Fleischmann, M., Tratner, J., Perry, M., Farmer, C., Hjelle, G. A. et al. (2019), 'geopandas/geopandas: v0. 6.0', Zenodo .

Jordahl, K., Van den Bossche, J., Wasserman, J., McBride, J., Gerard, J., Tratner, J., Perry, M. & Farmer, C. (2021), 'geopandas/geopandas: v0. 5.0', Zenodo .

Karduni, A., Kermanshah, A. & Derrible, S. (2016a), 'A protocol to convert spatial polyline data to network formats and applications to world urban road networks', Scientific data **3**(1), 1–7.

Karduni, A., Kermanshah, A. & Derrible, S. (2016b), 'Urban road network data', figshare https://figshare.com/articles/dataset/Urban_Road_Network_Data/2061897/1.

Karur, K., Sharma, N., Dharmatti, C. & Siegel, J. E. (2021), 'A survey of path planning algorithms for mobile robots', Vehicles **3**(3), 448–468.

Katoh, N., Ibaraki, T. & Mine, H. (1982), 'An efficient algorithm for k shortest simple paths', Networks **12**(4), 411–427.

Kaur, H., Singh, H. & Sharma, A. (2016), 'Geographic routing protocol: A review', International Journal of Grid and Distributed Computing **9**(2), 245–254.

Kaur, J., Singh, J., Sehra, S. S. & Rai, H. S. (2017), Systematic literature review of data quality within openstreetmap, in '2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS)', IEEE, pp. 177–182.

Kechagiopoulos, P. N. & Beligiannis, G. N. (2014), 'Solving the urban transit routing problem using a particle swarm optimization based algorithm', Applied Soft Computing **21**, 654–676.

Kim, G., Ong, Y.-S., Heng, C. K., Tan, P. S. & Zhang, N. A. (2015), 'City vehicle routing problem (city vrp): A review', IEEE Transactions on Intelligent Transportation Systems **16**(4), 1654–1666.

Kora, P. & Yadlapalli, P. (2017), 'Crossover operators in genetic algorithms: A review', International Journal of Computer Applications **162**(10).

Lambora, A., Gupta, K. & Chopra, K. (2019), Genetic algorithm- a literature review, in '2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)', pp. 380–384.

Laporte, G. (2007), 'What you should know about the vehicle routing problem', Naval Research Logistics (NRL) **54**(8), 811–819.

Lau, H. C., Chan, T., Tsui, W. & Pang, W. (2009), 'Application of genetic algorithms to solve the multidepot vehicle routing problem', IEEE transactions on automation science and engineering **7**(2), 383–392.

Lehner, B. & Grill, G. (2013), 'Global river hydrography and network routing: baseline data and new approaches to study the world's large river systems', Hydrological Processes **27**(15), 2171–2186.

Lehner, B., Liermann, C. R., Revenga, C., Vörösmarty, C., Fekete, B., Crouzet, P., Döll, P., Endejan, M., Frenken, K., Magome, J. et al. (2011), 'High-resolution mapping of the world's reservoirs and dams for sustainable river-flow management', Frontiers in Ecology and the Environment **9**(9), 494–502.

Leung, Y., Li, G. & Xu, Z.-B. (1998), 'A genetic algorithm for the multiple destination routing problems', IEEE Transactions on evolutionary computation **2**(4), 150–161.

Li, J.-P., Balazs, M. E., Parks, G. T. & Clarkson, P. J. (2002), 'A species conserving genetic algorithm for multimodal function optimization', Evolutionary computation **10**(3), 207–234.

Li, X., Chen, N., Ma, H., Nie, F. & Wang, X. (2024), 'A parallel genetic algorithm with variable neighborhood search for the vehicle routing problem in forest fire-fighting', IEEE Transactions on Intelligent Transportation Systems .

Lim, D., Ong, Y.-S., Jin, Y., Sendhoff, B. & Lee, B.-S. (2007), 'Efficient hierarchical parallel genetic algorithms using grid computing', Future Generation Computer Systems **23**(4), 658–670.

Lim, T. Y. (2014), 'Structured population genetic algorithms: a literature survey', Artificial Intelligence Review **41**(3), 385–399.

Lin, K., Musa, S. N. & Yap, H. J. (2024), 'Adaptive multi-objective algorithm for the sustainable electric vehicle routing problem in medical waste management', Transportation Research Record **2678**(7), 413–433.

Lin, L., Wu, C. & Ma, L. (2021), 'A genetic algorithm for the fuzzy shortest path problem in a fuzzy network', Complex & Intelligent Systems **7**, 225–234.

Liu, C., Zhou, M., Wu, J., Long, C. & Wang, Y. (2017), 'Electric vehicles en-route charging navigation systems: Joint charging and routing optimization', IEEE Transactions on Control Systems Technology **27**(2), 906–914.

Liu, H., Jin, C., Yang, B. & Zhou, A. (2017), 'Finding top-k shortest paths with diversity', IEEE Transactions on Knowledge and Data Engineering **30**(3), 488–502.

Liu, Z., Kong, Y. & Su, B. (2016), An improved genetic algorithm based on the shortest path problem, in '2016 IEEE international conference on information and automation (ICIA)', IEEE, pp. 328–332.

Lu, T. & Zhu, J. (2013), 'A genetic algorithm for finding a path subject to two constraints', Applied Soft Computing **13**(2), 891–898.

Luke, S. (2013), Essentials of metaheuristics, second edn, Lulu.

Maidana, R. G., Kristensen, S. D., Utne, I. B. & Sørensen, A. J. (2023), 'Risk-based path planning for preventing collisions and groundings of maritime autonomous surface ships', Ocean Engineering **290**, 116417.

Marinakis, Y., Iordanidou, G.-R. & Marinaki, M. (2013), 'Particle swarm optimization for the vehicle routing problem with stochastic demands', Applied Soft Computing **13**(4), 1693–1704.

McKinney, W. et al. (2011), 'pandas: a foundational python library for data analysis and statistics', Python for high performance and scientific computing **14**(9), 1–9.

Meng, F. H., Lao, Y., Wai, L. H. & Chuin, L. H. (1999), A multi-criteria, multi-modal passenger route advisory system, in 'Proceedings', Citeseer, p. No page numbers available.

Michail, D., Kinable, J., Naveh, B. & Sichi, J. V. (2020), 'Jgrapht—a java library for graph data structures and algorithms', ACM Transactions on Mathematical Software (TOMS) **46**(2), 1–29.

Mirabi, M., Ghomi, S. F. & Jolai, F. (2010), 'Efficient stochastic hybrid heuristics for the multi-depot vehicle routing problem', Robotics and Computer-Integrated Manufacturing **26**(6), 564–569.

Mohammed, M. A., Abd Ghani, M. K., Hamed, R. I., Mostafa, S. A., Ahmad, M. S. & Ibrahim, D. A. (2017), 'Solving vehicle routing problem by using improved genetic algorithm for optimal solution', Journal of computational science **21**, 255–262.

Monita, V., Irawati, I. D. & Tulloh, R. (2020), 'Comparison of routing protocol performance on multimedia services on software defined network', Bulletin of Electrical Engineering and Informatics **9**(4), 1612–1619.

Mor, A. & Speranza, M. G. (2022), 'Vehicle routing problems over time: a survey', Annals of Operations Research **314**(1), 255–275.

Moza, M. & Kumar, S. (2020), 'Finding k shortest paths in a network using genetic algorithm', Int. J. Comput. Netw. Inf. Secur.(IJCNIS) **12**(5), 56–73.

Munetomo, M., Takai, Y. & Sato, Y. (1998), A migration scheme for the genetic adaptive routing algorithm, in 'SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)', Vol. 3, IEEE, pp. 2774–2779.

Naddef, D. & Rinaldi, G. (2002), Branch-and-cut algorithms for the capacitated vrp, in 'The vehicle routing problem', SIAM, pp. 53–84.

Nanayakkara, S. C., Srinivasan, D., Lup, L. W., German, X., Taylor, E. & Ong, S. H. (2007), Genetic algorithm based route planner for large urban street networks, in '2007 IEEE Congress on Evolutionary Computation', IEEE, pp. 4469–4474.

Okulewicz, M. & Mańdziuk, J. (2013), Application of particle swarm optimization algorithm to dynamic vehicle routing problem, in 'Artificial Intelligence and Soft Computing: 12th International Conference, ICAISC 2013, Zakopane, Poland, June 9-13, 2013, Proceedings, Part II 12', Springer, pp. 547–558.

Ombuki, B., Ross, B. J. & Hanshar, F. (2006), 'Multi-objective genetic algorithms for vehicle routing problem with time windows', Applied Intelligence **24**, 17–30.

Osaba, E., Carballedo, R., Diaz, F., Onieva, E., Lopez, P. & Perallos, A. (2014), On the influence of using initialization functions on genetic algorithms solving combinatorial optimization problems: A first study on the tsp, in '2014 IEEE conference on evolving and adaptive intelligent systems (EAIS)', IEEE, pp. 1–6.

Oyola, J., Arntzen, H. & Woodruff, D. L. (2018), 'The stochastic vehicle routing problem, a literature review, part i: models', EURO Journal on Transportation and Logistics **7**(3), 193–221.

Palakonda, V. & Mallipeddi, R. (2020), 'An evolutionary algorithm for multi and many-objective optimization with adaptive mating and environmental selection', IEEE Access **8**, 82781–82796.

Park, S. & Cheng, T. (2023), 'Framework for constructing multimodal transport networks and routing using a graph database: A case study in london', Transactions in GIS **27**(5), 1391–1417.

Pebesma, E. & Bivand, R. S. (2005), 'S classes and methods for spatial data: the sp package', R news **5**(2), 9–13.

Pebesma, E. J. et al. (2018), 'Simple features for r: standardized support for spatial vector data.', R J. **10**(1), 439.

Peyer, S., Rautenbach, D. & Vygen, J. (2009), 'A generalization of dijkstra's shortest path algorithm with applications to vlsi routing', Journal of Discrete Algorithms **7**(4), 377–390.

Prasetya, D. A., Nguyen, P. T., Faizullin, R., Iswanto, I. & Armay, E. F. (2020), 'Resolving the shortest path problem using the haversine algorithm', Journal of critical reviews **7**(1), 62–64.

Purshouse, R. C. & Fleming, P. J. (2002), Why use elitism and sharing in a multi-objective genetic algorithm?, in 'Proceedings of the 4th Annual Conference on Genetic and Evolutionary computation', pp. 520–527.

Qiongbing, Z. & Lixin, D. (2016), 'A new crossover mechanism for genetic algorithms with variable-length chromosomes for path optimization problems', Expert Systems with Applications **60**, 183–189.

Qiuqi, R. et al. (2004), A gene-constrained genetic algorithm for solving shortest path problem, in 'Proceedings 7th International Conference on Signal Processing, 2004. Proceedings. ICSP'04. 2004.', Vol. 3, IEEE, pp. 2510–2513.

Rachmawati, D. & Gustin, L. (2020), Analysis of dijkstra's algorithm and a* algorithm in shortest path problem, in 'Journal of Physics: Conference Series', Vol. 1566, IOP Publishing, p. 012061.

Rajeswari, D., Prakash, M., Ramamoorthy, S. & Sudhakar, S. (2021), 'Mapreduce framework based gridlet allocation technique in computational grid', Computers & Electrical Engineering **92**, 107131.

Rao, A., Ratnasamy, S., Papadimitriou, C., Shenker, S. & Stoica, I. (2003), Geographic routing without location information, in 'Proceedings of the 9th annual international conference on Mobile computing and networking', pp. 96–108.

Razali, N. M., Geraghty, J. et al. (2011), Genetic algorithm performance with different selection strategies in solving tsp, in 'Proceedings of the world congress on engineering', number 1 in '-', International Association of Engineers Hong Kong, China, pp. 1–6.

Riget, J. & Vesterstrøm, J. S. (2002), 'A diversity-guided particle swarm optimizer-the arpso', Dept. Comput. Sci., Univ. of Aarhus, Aarhus, Denmark, Tech. Rep **2**, 2002.

Ritzinger, U., Puchinger, J. & Hartl, R. F. (2016), 'A survey on dynamic and stochastic vehicle routing problems', International Journal of Production Research **54**(1), 215–231.

Rocha, M., Sousa, P., Cortez, P. & Rio, M. (2011), 'Quality of service constrained routing optimization using evolutionary computation', Applied Soft Computing **11**(1), 356–364.

Rodriguez-Iturbe, I., Muneepeerakul, R., Bertuzzo, E., Levin, S. A. & Rinaldo, A. (2009), 'River networks as ecological corridors: A complex systems perspective for integrating hydrologic, geomorphologic, and ecologic dynamics', Water Resources Research **45**(1).

Rossi, R. A. & Ahmed, N. K. (2015a), 'The network data repository with interactive graph analytics and visualization', Network Repository https://networkrepository.com/graph500.php.

Rossi, R. A. & Ahmed, N. K. (2015b), The network data repository with interactive graph analytics and visualization, in 'AAAI'.

Ruiz-Meza, J. & Montoya-Torres, J. R. (2022), 'A systematic literature review for the tourist trip design problem: Extensions, solution techniques and future research lines', Operations Research Perspectives **9**, 100228.

Schweimer, C., Geiger, B. C., Wang, M., Gogolenko, S., Mahmood, I., Jahani, A., Suleimenova, D. & Groen, D. (2021), 'A route pruning algorithm for an automated geographic location graph construction', Scientific reports **11**(1), 11547.

Shah, R. C. & Rabaey, J. M. (2002), Energy aware routing for low energy ad hoc sensor networks, in '2002 IEEE wireless communications and networking conference record. WCNC 2002 (cat. No. 02TH8609)', Vol. 1, IEEE, pp. 350–355.

Sharma, H., Galván, E. & Mooney, P. (2025a), 'Multipath island-based genetic algorithm for the k-most diverse near-shortest paths', Information Sciences p. 122495.

Sharma, H., Galván, E. & Mooney, P. (2025b), 'A parallel genetic algorithm for multi-criteria path routing on complex real-world road networks', Applied Soft Computing **170**, 112559.

Sharma, H., Mooney, P. & Galván, E. (2023), 'A method for creating complex real-world networks using esri shapefiles.', MethodsX **11**, 102426.
URL: *https://www.sciencedirect.com/science/article/pii/S2215016123004223*

Sharma, H., Mooney, P. & Galván-López, E. (2024), 'Geographnetworks: Shapefile-derived datasets for accurate and scalable graphical representations'. Dataset.

Shukla, A., Pandey, H. M. & Mehrotra, D. (2015), Comparative review of selection techniques in genetic algorithm, in '2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)', pp. 515–519.

Solomon, M. M. (1987), 'Algorithms for the vehicle routing and scheduling problems with time window constraints', Operations research **35**(2), 254–265.

Soueres, P. & Laumond, J.-P. (1996), 'Shortest paths synthesis for a car-like robot', IEEE Transactions on Automatic Control **41**(5), 672–688.

Souza, R., Lucena, O., Garrafa, J., Gobbi, D., Saluzzi, M., Appenzeller, S., Rittner, L., Frayne, R. & Lotufo, R. (2018), 'An open, multi-vendor, multi-field-strength brain mr dataset and analysis of publicly available skull stripping methods agreement', NeuroImage **170**, 482–494.

Spliet, R. & Gabor, A. F. (2015), 'The time window assignment vehicle routing problem', Transportation Science **49**(4), 721–731.

Srinivas, N. & Deb, K. (1994), 'Muiltiobjective optimization using nondominated sorting in genetic algorithms', Evolutionary computation **2**(3), 221–248.

Stein, M. (1987), 'Large sample properties of simulations using latin hypercube sampling', Technometrics **29**(2), 143–151.

Subashini, G. & Bhuvaneswari, M. (2012), 'Comparison of multi-objective evolutionary approaches for task scheduling in distributed computing systems', Sadhana **37**(6), 675–694.

Tan, K. C., Lee, T. H. & Khor, E. F. (2001), 'Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization', IEEE Transactions on Evolutionary Computation **5**(6), 565–588.

Tan, K. C., Lee, T. H. & Khor, E. F. (2002), 'Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons', Artificial intelligence review **17**(4), 251–290.

Tarjan, R. (1972), 'Depth-first search and linear graph algorithms', SIAM journal on computing **1**(2), 146–160.

Terpilowski, M. A. (2019), 'scikit-posthocs: Pairwise multiple comparison tests in python', Journal of Open Source Software **4**(36), 1169.

Textor, J., Van der Zander, B., Gilthorpe, M. S., Liśkiewicz, M. & Ellison, G. T. (2016), 'Robust causal inference using directed acyclic graphs: the r package 'dagitty'', International journal of epidemiology **45**(6), 1887–1894.

Tian, Y., Shao, S., Xie, G. & Zhang, X. (2024), 'A multi-granularity clustering based evolutionary algorithm for large-scale sparse multi-objective optimization', Swarm and Evolutionary Computation **84**, 101453.

Toth, P. & Vigo, D. (2002a), 'Models, relaxations and exact approaches for the capacitated vehicle routing problem', Discrete Applied Mathematics **123**(1-3), 487–512.

Toth, P. & Vigo, D. (2002b), The vehicle routing problem, SIAM.

Turton, I. (2008), Geo tools, in 'Open source approaches in spatial data handling', Springer, pp. 153–169.

Van Wart, A. T., Durrant, J., Votapka, L. & Amaro, R. E. (2014), 'Weighted implementation of suboptimal paths (wisp): an optimized algorithm and tool for dynamical network analysis', Journal of chemical theory and computation **10**(2), 511–517.

Verma, S., Pant, M. & Snasel, V. (2021), 'A comprehensive review on nsga-ii for multi-objective combinatorial optimization problems', Ieee Access **9**, 57757–57791.

Virtanen, P., Gommers, R., Burovski, E., Oliphant, T. E., Weckesser, W., Cournapeau, D., Peterson, P., Reddy, T., Haberland, M., Wilson, J. et al. (2021), 'scipy/scipy: Scipy 1.6. 0', Zenodo .

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J. et al. (2020), 'Scipy 1.0: fundamental algorithms for scientific computing in python', Nature methods **17**(3), 261–272.

Walker, K. E. (2016), 'tigris: An r package to access and work with geographic data from the us census bureau'.

Wang, F., Man, Y. & Man, L. (2014), Intelligent optimization approach for the k shortest paths problem based on genetic algorithm, in '2014 10th International Conference on Natural Computation (ICNC)', IEEE, pp. 219–224.

Wang, K., Lan, S. & Zhao, Y. (2017), 'A genetic-algorithm-based approach to the two-echelon capacitated vehicle routing problem with stochastic demands in logistics service', Journal of the Operational Research Society **68**(11), 1409–1421.

Wang, M. Y. (2019), Deep graph library: Towards efficient and scalable deep learning on graphs, in 'ICLR workshop on representation learning on graphs and manifolds'.

Wang, Y., Peng, S., Zhou, X., Mahmoudi, M. & Zhen, L. (2020), 'Green logistics location-routing problem with eco-packages', Transportation Research Part E: Logistics and Transportation Review **143**, 102118.

Wang, Z. & Crowcroft, J. (1996), 'Quality-of-service routing for supporting multimedia applications', IEEE Journal on selected areas in communications **14**(7), 1228–1234.

Ward, A. & Webster, M. (2016), Sociality: the behaviour of group-living animals, Vol. 407, Springer.

Wen, W. & Hsu, S. (2005), 'A route navigation system with a new revised shortest path routing algorithm and its performance evaluation', WIT Transactions on The Built Environment **77**.

Westra, E. (2015), Python Geospatial Analysis Essentials, Packt Publishing Ltd.

Wolfinger, D., Tricoire, F. & Doerner, K. F. (2019), 'A matheuristic for a multimodal long haul routing problem', EURO Journal on Transportation and Logistics **8**(4), 397–433.

Xu, Z., Elomri, A., Pokharel, S. & Mutlu, F. (2019), 'A model for capacitated green vehicle routing problem with the time-varying vehicle speed and soft time windows', Computers & Industrial Engineering **137**, 106011.

Yang, S., Cheng, H. & Wang, F. (2009), 'Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks', IEEE

Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **40**(1), 52–63.

Yang, S. & Natarajan, U. (2010), 'Multi-objective optimization of cutting parameters in turning process using differential evolution and non-dominated sorting genetic algorithm-ii approaches', The International Journal of Advanced Manufacturing Technology **49**(5), 773–784.

Yang, W.-H., Mathur, K. & Ballou, R. H. (2000), 'Stochastic vehicle routing problem with restocking', Transportation science **34**(1), 99–112.

Yang, Y. & Wang, J. (2008), Design guidelines for routing metrics in multihop wireless networks, in 'IEEE INFOCOM 2008-The 27th Conference on Computer Communications', IEEE, pp. 1615–1623.

Yap, W. & Biljecki, F. (2023a), 'A global feature-rich network dataset of cities and dashboard for comprehensive urban analyses', Scientific data **10**(1), 667.

Yap, W. & Biljecki, F. (2023b), 'The urbanity global network dataset', figshare https://figshare.com/articles/dataset/Global_Urban_Network_Dataset/22124219.

Ye, F., Doerr, C. & Bäck, T. (2019), Interpolating local and global search by controlling the variance of standard bit mutation, in '2019 IEEE Congress on Evolutionary Computation (CEC)', IEEE, pp. 2292–2299.

Yen, J. Y. (1971), 'Finding the k shortest loopless paths in a network', management Science **17**(11), 712–716.

Yin, N. (2022), 'Multiobjective optimization for vehicle routing optimization problem in low-carbon intelligent transportation', IEEE Transactions on Intelligent Transportation Systems **24**(11), 13161–13170.

Yin, W. & Yang, X. (2013), 'A totally astar-based multi-path algorithm for the recognition of reasonable route sets in vehicle navigation systems', Procedia-Social and Behavioral Sciences **96**, 1069–1078.

Yu, Z., Yu, X., Koudas, N., Liu, Y., Li, Y., Chen, Y. & Yang, D. (2020), Distributed processing of k shortest path queries over dynamic road networks, in 'Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data', pp. 665–679.

Yue, C., Suganthan, P. N., Liang, J., Qu, B., Yu, K., Zhu, Y. & Yan, L. (2021), 'Differential evolution using improved crowding distance for multimodal multiobjective optimization', Swarm and Evolutionary Computation **62**, 100849.

Yuen, S. Y. & Chow, C. K. (2008), 'A genetic algorithm that adaptively mutates and never revisits', IEEE transactions on evolutionary computation **13**(2), 454–472.

Zhang, H., Ge, H., Yang, J. & Tong, Y. (2022), 'Review of vehicle routing problems: Models, classification and solving algorithms', Archives of Computational Methods in Engineering pp. 1–27.

Zhang, M., Li, W., Zhang, L., Jin, H., Mu, Y. & Wang, L. (2023), 'A pearson correlation-based adaptive variable grouping method for large-scale multi-objective optimization', Information Sciences **639**, 118737.

Zhang, X., Chen, Y. & Li, T. (2015), Optimization of logistics route based on dijkstra, in '2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)', IEEE, pp. 313–316.

Zhang, X., Liu, Z., Zhang, M. & Li, L. (2021), An experimental study on exact multi-constraint shortest path finding, in 'Databases Theory and Applications: 32nd Australasian Database Conference, ADC 2021, Dunedin, New Zealand, January 29–February 5, 2021, Proceedings 32', Springer, pp. 166–179.

Zhu, X., Luo, W. & Zhu, T. (2014), An improved genetic algorithm for dynamic shortest path problems, in '2014 IEEE congress on evolutionary computation (CEC)', IEEE, pp. 2093–2100.

Zitzler, E. & Thiele, L. (1998), Multiobjective optimization using evolutionary algorithms—a comparative case study, in 'International conference on parallel problem solving from nature', Springer, pp. 292–301.

Zitzler, E. & Thiele, L. (1999), 'Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach', IEEE transactions on Evolutionary Computation **3**(4), 257–271.