MULTI-OBJECTIVE SHORTEST PATH APPROACH FOR ROUTING AND SCHEDULING

Lilla Beke

Submitted in partial fulfillment of the requirements
of the Degree of Doctor of Philosophy

School of Engineering and Materials Science

Queen Mary University of London

April 2022

Statement of originality

I, Lilla Beke, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without the prior written consent of the author.

Signature: Lilla Beke

Date: 28/04/2022

Details of collaboration and publications:

Chapter 5 is based on material previously published in the journal SN Computer Science (Beke et al. 2021). The design of the local search operator used in Chapters 6 and 7 was done in collaboration with researchers from the Instituto Politécnico Nacional, Mexico: Dr Lourdes Uribe, Dr Adriana Lara and Prof. Carlos A. Coello Coello. This research was published in (Beke et al. 2023).

Abstract

Customers and goods are being transported at an ever-increasing rate, through complex and interconnected transportation systems. The need for efficiency in terms of economic and environmental aspects, to name a few, gives rise to optimisation problems that often include finding shortest paths. This is why shortest path problems are among the most studied combinatorial optimisation problems. The research presented in this thesis focuses on a class of shortest path problems that are multi-objective and where time window constraints are present. The thesis argues that such problems are best modelled through multigraphs, leading to the Multi-objective Shortest Path Problem on Multigraphs with Time Windows (MSPPMTW). The multigraph allows more detailed modelling of such problems, which is key to accessing the untapped potential for improved efficiency.

Multiple aspects of the problem are investigated. The MSPPMTW has increased search space compared to the simple graph Multi-objective Shortest Path Problem, which is already NP-hard. Firstly, the increased computational complexity is investigated empirically, and a model is proposed for predicting the computational effort required, based on easily measurable metrics describing the problem instance, such as the number of parallel arcs or the size of the network. Secondly, a benchmark generation method is proposed for the MSPPMTW, based on the observations of the predictive model. Lastly, a memetic algorithm (with multiple variants based on different representation methods) is proposed to address the problem of finding solutions in short time budgets and scaling to higher numbers of parallel arcs. The memetic algorithm is tested on the proposed benchmark set and a real-world application, the airport ground movement problem.

The thesis finds that the proposed memetic algorithm is comparable to the state of the art solution methods. The metaheuristic approaches also have higher promise for future applications

in optimising broader transportation systems as a whole.

There are no solutions; there are only trade-offs.

THOMAS SOWELL

Acknowledgements

Firstly, I would like to thank my supervisor Dr Jun Chen for the expert advice, patience and all the support he provided throughout the four years of study. I would also like to thank Dr Michal Weiszer, who helped me greatly, especially in the initial stages of my studies, which is always the hardest part.

I also profited from collaborating with researchers from the Instituto Politécnico Nacional, Mexico, where I spent a month that will always be a memory dear to me. I would like to thank Dr Lourdes Uribe and Dr Adriana Lara for their interest in the research topic I brought, the enlightening discussions and lastly their kind hospitality.

I would like to thank my family, without their support, I could not be where I am today. Last but not least, I wish to thank Zoltán for his understanding, encouragement and companionship especially in these last few months, but also throughout these four years, and the journey leading up to it.

Contents

Statement of originality 1
Abstract 2
Acknowledgements 5

List of Figures **10**

List of Tables **13**

List of Abbreviations **17**

1 Introduction **21**
1.1 Background and motivation 21
1.2 Research Goals 25
1.3 Contributions 26
1.4 Publication List 27
1.5 Thesis Outline 28

2 Background **29**
2.1 Preliminaries 29
2.1.1 Notions of Graph Theory 29
2.1.2 Multi-objective optimisation 33
2.2 Shortest path problems 34
2.3 Enumerative algorithms to solve the shortest path problem 39
2.4 Metaheuristic algorithms for the shortest path problem 41
2.4.1 Genetic Algorithms to solve shortest path problems 43

2.5 Real world problems 50
2.5.1 Airport ground movement problem 50
2.5.2 Other real-world applications 51
2.6 Evaluation 53
2.6.1 Benchmark sets 53
2.6.2 Performance measures 53

3 Exploring the difficulty of Multigraph Multi-objective Shortest Path Problem instances 55

3.1 Introduction 55
3.2 Network structure generation 56
3.2.1 Benchmark sets for shortest path problems 57
3.2.2 Network generation outside of shortest path problems 59
3.3 The proposed predictive model 63
3.3.1 Overview of the predictive model 63
3.3.2 Employed artificial network models 63
3.3.3 Cost generation 68
3.3.4 Overview of the generated set of instances 69
3.3.5 Features considered 71
3.3.6 Measuring difficulty of instances 75
3.4 Model building and results 77
3.4.1 Domination checks performed as the difficulty measure 77
3.4.2 Number of Pareto optimal solutions as the difficulty measure 82
3.5 Discussion 83

4 Constrained benchmark problem generation 86

4.1 Instance generation for the proposed benchmark set 88
4.1.1 Methods for instance generation 88
4.2 Experiments 94
4.2.1 Non-additivity of costs 96
4.2.2 The effect of modified costs on difficulty of unconstrained instances . . 97

4.2.3 The effect of time windows on the difficulty and feasibility 98
4.3 Discussion 107

5 Genetic Algorithm for the Unconstrained MSPP on Multigraphs 109
5.1 Introduction 109
5.2 Proposed representations and initialisation methods 111
5.2.1 Representing parallel arcs 111
5.2.2 Variation operators for the representations 117
5.2.3 Heuristic initialisation 120
5.3 Implementation details 125
5.3.1 Test instances 127
5.3.2 Evaluation of approximate solutions 127
5.4 Results 130
5.4.1 Finding exact solutions for the Multigraph MSPP instances 130
5.4.2 Initialisation methods 130
5.4.3 Comparison of variants of the proposed algorithm 131
5.5 Discussion 136

6 Genetic Algorithm for the Constrained MSPPs on Multigraphs 138
6.1 Introduction 138
6.2 Proposed modifications to the algorithm 140
6.2.1 Constraint handling and fitness evaluation 140
6.2.2 Local search process 142
6.2.3 Search based on direct variable length representation 143
6.2.4 Other representations 146
6.3 Results 146
6.3.1 Comparing variants 148
6.3.2 Smaller time budgets 154
6.3.3 Comparing to enumerative approach 155
6.3.4 Higher numbers of parallel arcs allow more solutions 158
6.3.5 Dominated costs 160

6.4 Discussion 161

7 Airport Ground Movement Problem as a Real World Application 163

7.1 Introduction 163

7.2 Airport ground movement as a combinatorial optimisation problem 165

7.2.1 Sequential routing of aircraft 166

7.2.2 The layout graph 168

7.2.3 The speed profile graph 168

7.2.4 Non-additivity of costs 170

7.2.5 Routing problem considering a single aircraft 171

7.3 The adaptation of the memetic algorithm 172

7.3.1 Direct variable length representation 172

7.3.2 Search based on priority based representations 173

7.3.3 Fitness function and constraint handling 176

7.4 Implementation details 178

7.4.1 Parameters 178

7.4.2 Problem instance 179

7.4.3 Evaluation of solutions 179

7.5 Results 180

7.5.1 Results based on the enumerative solution approach 181

7.5.2 Results based on convergence based termination 182

7.5.3 Potential for real time decision support 188

7.6 Discussion 189

8 Conclusion and Future Directions 191

8.1 Summary 191

8.2 Outlook on future work 194

Bibliography 204

List of Figures

2.1 An example of a simple graph and a path in it from node 1 to node 5, indicated by the bold line. 30

2.2 An example of a multigraph network and a path in it from node 1 to node 5, indicated by the bold lines. 31

2.3 Illustration of supported and unsupported solutions on a Pareto front. Blue circles represent supported solutions, green circles represent unsupported solutions that are not on the convex hull, and gray points are dominated solutions. 33

3.1 Illustration of the role of f_s the shrinking factor. 70

3.2 Goodness of fit shown in hexagon plots on unseen test data for the prediction of the logarithmic transformation of the number of domination tests by the model with the selected features (3.2a) and the model with the literature features (3.2b) . 81

3.3 Goodness of fit shown on hexagon plots on unseen test data for the prediction of the logarithmic transformation of the number of Pareto-optimal solutions tests by the model with the selected features (3.3a) and the model with the literature features (3.3b) 83

4.1 Cost assignment for the benchmark instances. 89

4.2 Various possibilities for cost generation with two objectives. 91

4.3 The effect of different values of the parameters p_{freq} and p_{len} on the created time windows, and an example of adjusting the number of blocked intervals based on the centrality of the arcs. 95

4.4 Illustration of non-additivity caused by time window constraints. 96

4.5 Predicted and actual difficulty of the proposed benchmark instances without time windows (blue points) against the instance set of Chapter 3. All predictions are generated with the prediction model from Chapter 3. 98
4.6 The number of domination tests for benchmark instances with an increasing time window length. 100
4.7 The number of solutions found for benchmark instances with an increasing time window length. 100
4.8 Time objective of the quickest path in the Pareto front for benchmark instances with increasing time window length. 101
4.9 Difficulty of benchmark instances with increasing time window frequency against the prediction model from the previous chapter. Infeasible instances (instance solved within 4 days, NAMOA* returned no solutions) are differentiated. 102
4.10 The number of domination tests for benchmark instances with an increasing time window frequency. 103
4.11 The umber of solutions returned by NAMOA* for benchmark instances with increasing time window frequency. Infeasible instances (instance solved within 4 days, NAMOA* returned no solutions) are not shown. 104
4.12 Change in time objective of the quickest solution found for different p_{fr} values. 104
4.13 Results of removing parallel arcs from arcs with below average length compared to keeping all parallel arcs and removing parallel arcs from all arcs in terms of number of domination tests and the best second objective values among the solutions found. 106

5.1 An example of a multigraph network and a solution path from node 1 to node 5 in it, indicated by the bold lines. 110
5.2 Illustration of WMX for the matrix chromosome. 119
5.3 Illustration of insertion mutation. 119
5.4 Illustration of the role of τ_{max} in heuristic initialisation. 122
5.5 Fitness and validity of the initial population depending on the initialisation method and value of τ_{max}, on two different scales. 123
5.6 Example of a successful approximation of the Pareto front. 133

6.1 The steps of the algorithm with the local search operator. 140
6.2 Illustration of direct crossover in multigraphs. 144
6.3 Effects of initialisation on number of runs of MARMT producing at least one valid solution for different network types, by time blocked interval length. . . . 150
6.4 Effects of initialisation on number of runs of MARMT producing at least one valid solution for different network types, by time blocked interval frequency. . 151
6.5 Effect of parallel arcs on number of instances with any valid solutions, note that a positive value of ρ_1 implies negative correlation between the objectives (see Section 4.1.1) 159
6.6 Effect of including arcs with dominated costs on number of instances with any valid solutions 161

7.1 Illustration of the layout of the Hong Kong airport with two alternative routes for an aircraft. 165
7.2 The reason behind inhomogeneous numbers of parallel arcs in the speed profile graph. There are two straight segments (angles below 30 degrees) between the same two nodes, A and B. 170
7.3 Illustration of non-additivity property. 170
7.4 Relationship between solution quality as measured by the relative weighted aggregate of objectives, local search rate, sum of computational time, and number of speed profiles for the 506 aircraft, with the direct representation. Experiments with different w_1 values are grouped together. 184
7.5 Difference between Airport-MARMT with and without local search and AMOA*. Three different weights are used for reserving trajectories for individual aircraft. Each marker represents the average of 10 data points. 186

List of Tables

3.1 Problem instance choice for Shortest Path Problems in the literature 60

3.2 Graph metrics collected in experiments. 75

3.3 R-squared values for XGB models on test data, where the testing set contains network types not contained in the training set. 80

3.4 Permutation importances for the prediction of the logarithmic transformation of the number of domination tests with the selected features. 82

3.5 Permutation importances for the prediction of the logarithmic transformation of the number of Pareto-optimal solutions with the selected features. 84

4.1 Distributions of number of domination tests depending on the position of the blocked intervals in the network regarding centrality of end nodes of the arcs. . 105

4.2 Distributions of number of solutions found depending on the position of the blocked intervals in the network regarding distance to destination node. 105

5.1 Representations and their variants for the Multigraph MSPP. Examples are shown for the encoding of the solution path in Fig. 5.1 using each representation. Variants investigated are listed for each representation. The details of the variants are described in Section 5.2.2 118

5.2 Values of the parameters for the different variants, tuned by the irace package. . 126

5.3 The performance of NAMOA* on the 32 problem instances. 129

5.4 The role of the heuristic initialisation methods in the solution quality achieved by variants of the proposed algorithm using different genetic representations and crossover operators, described by theε, R3 and RHV metric. 132

5.5 Comparing the best variants of the proposed algorithm for the four representations, separately for the 32 test instances with a time budget of 10s. 135

6.1 Number of runs returning at least one valid solution with each of the 20 considered variant. 148
6.2 Average performance measures for the 20 variants, only considering cases where at least one valid solution was found. 153
6.3 Average performance measures for the two best variants, the cases where no solutions are found are incorporated with a penalty. 154
6.4 Number of runs returning at least one valid solution with each of the 20 considered variants, with a time budget of 30 seconds. 154
6.5 Average performance measures for the 2 best variants, only including cases where at least one valid solution was found, with a time budget of 30 seconds. . 154
6.6 Average performance measures for the two best variants, cases where no solutions found are incorporated with a penalty, with a time budget of 30 seconds. . 155
6.7 Average and maximum running times of NAMOA*, in hours. 155
6.8 Number of cases when NAMOA* failed to find any solution, when some solutions were found by MARMT, by time window parameters. 156
6.9 Median values of the performance measures. 157
6.10 Mean values of the performance measures. 157
6.11 Instances with at least one valid solution found for different numbers of parallel arcs. 158

7.1 Notations for the current chapter. 167
7.2 Variants of Airport-MARMT with different representations. 178
7.3 Tuned values of the parameters. 178
7.4 Computational times of AMOA* (u=3) routing a single aircraft. 181
7.5 Number of solutions found by AMOA* (u=3) for a single aircraft. 181
7.6 Relative weighted aggregate of objectives for sequential routing of the 506 aircraft with different local search rates and different numbers of parallel arcs. 183

7.7 Mean ε indicator for sequential routing of the 506 aircraft with different local search rates and different numbers of parallel arcs. 185
7.8 Mean number of Pareto optimal solutions found for the 506 aircraft with different local search rates and different numbers of parallel arcs. Stopping criteria is 10 generations without improvement. 187
7.9 Mean relative weighted aggregate for the 506 aircraft with the 10 seconds time budget. Different local search rates and different numbers of parallel arcs are included for the direct representation. 188
7.10 Mean number of Pareto optimal solutions for the 506 aircraft with the 10 seconds time budget. Different local search rates and different numbers of parallel arcs are included for the direct representation. 189

1 Illustration of the role of the heuristic initialisation methods in the solution quality achieved by variants of NSGA-II using different genetic representations and crossover operators, described by the R3 metric. 197
2 Comparing variants of NSGA-II with their best initialisation methods for both the 10s and 20s time budgets. Average solution qualities on the 32 instances are shown according to the three quality indicators. 198
3 Comparing the best variants of NSGA-II with a time budget of 20s for the four representations, separately for the 32 test instances. The values of the average RHV quality indicator of 50 runs with each representation is shown. 199
4 Comparing the best variants of NSGA-II for the four representations, separately for the 32 test instances with a time budget of 10s. The values of the average Epsilon quality indicator of 50 runs with each representation is shown. 200
5 Comparing the best variants of NSGA-II for the four representations, separately for the 32 test instances with a time budget of 20s. The values of the average Epsilon quality indicator of 50 runs with each representation is shown. 201
6 Comparing the best variants of NSGA-II for the four representations, separately for the 32 test instances with a time budget of 10s. The values of the average Epsilon R3 quality indicator of 50 runs with each representation is shown. . . . 202

7 Comparing the best variants of NSGA-II for the four representations, separately for the 32 test instances with a time budget of 20s. The values of the average Epsilon R3 quality indicator of 50 runs with each representation is shown. . . . 203

List of Abbreviations

$\mathscr{F}_{i,j}$	Time windows for arc between nodes i and j
ε	Epsilon-indicator
A	Set of arcs in a given network
b	Time blocks per arc
c_a	Normalized average betweenness centrality
d_a	Normalized average hopcount
d_{max}	Diameter of I_G
g	Blocked interval attraction function
h	Hopcount between origin and destination
I_D	Destination node in instance I
I_G	Multigraph structure in instance I
I_O	Origin node in instance I
n_{tb}	Number of time blocks assigned to an arc
p_{freq}	Blocked interval frequency parameter
p_{len}	Blocked interval length factor parameter
t_{max}	Maximum considered period for time window assignment
V	Set of nodes in a given network

BA Barabasi-Albert random graph model

D Destination node

DF Direct Fixed Length Representation

DV-dir Direct Variable Length with Direct Encoding of Parallel Arcs

DV-indir Direct Variable Length with Indirect Encoding of Parallel Arcs

ER Erdos-Renyi random graph model

Exp Exponential random graph model

FPTAS Fully Polynomial-Time Approximation Scheme

GA Genetic Algorithm

GG Random geometric random graph model

Grav Gravity random graph model

Grid Grid-based random graph model

HeurI1 Heuristic initialisation based on graph structures

HeurI2 Heuristic initialisation based on single objective search

HK Holme-Kim random graph model

IP-PX Integer Priority with Position-based Crossover

IP-WMX Integer Priority with Weight Mapping Crossover

MA Memetic Algorithm

MARMT Memetic Algorithm for Routing in Multigraphs with Time constraints

MMSPP Multigraph Multi-objective Shortest Path Problem

MOA* Multi-objective A*

MSPP Multi-objective Shortest Path Problem

MSPPMTW Multi-objective Shortest Path Problem on Multigraphs with Time Windows

NAMOA* New Approach to Multi-objective A*

NM1 Undirected version of original Netmaker networks

NM2 Different version of undirected Netmaker

NSGA Non-dominated Sorting Genetic Algorithm

NSGA-II Non-dominated Sorting Genetic Algorithm II

NSGA-III Non-dominated Sorting Genetic Algorithm III

NWS Newman-Watts-Strogratz random graph model

O Origin node

Prox Proximity graph model

PX Position-based Crossover

R3 R3 indicator

RHV Relative Hypervolume indicator

RK-2ptX Random Keys with Two-point Crossover

RK-arithX Random Keys with Arithmetic Crossover

RK-uniX Random Keys with Uniform Crossover

RNG Relative Neighbourhood Graph random graph model

RR Random regular graph model

SMSPP Simple Multi-objective Shortest Path Problem

SPP Shortest Path Problem

Tree Random tree graph model

VRP Vehicle Routing Problem

WMX Weight Mapping Crossover

XGB Extreme Gradient Boosting

Chapter 1

Introduction

This chapter provides an introduction to the thesis by discussing the motivation and goals of the research presented in the thesis, followed by a summary of the main contributions, a list of the associated publications, and outlines of the thesis.

1.1 Background and motivation

The central problem considered in this thesis is a special case of the Shortest Path Problem (SPP), namely, the Multi-objective Shortest Path Problem on Multigraphs with Time Windows (MSPPMTW). SPPs are one of the most studied combinatorial optimisation problems in the field of Operational Research and Artificial Intelligence. Real-world applications of SPPs are abundant in routing problems related to transportation and communication. However, they also emerge in more surprising settings, for example, speech recognition, where the most likely meaning of a sentence can be estimated based on graphs of associations (Mohri 1997).

Despite what the name suggests, shortest path problems often are not looking for the optimal path in terms of distance, but other objectives such as time or cost. When there are more than one conflicting objectives to optimise for, the problem is known as the Multi-objective Shortest Path Problem (MSPP). SPPs can be differentiated based on the number of objectives, the type of constraints present, and the properties of the network (static/dynamic) including the costs associated with the arcs of the network (positive or not necessarily, additive/non-additive). This variety gives rise to a continuously developing area of research centred around the modelling and

solving of different SPPs. Small modifications, such as changes in the size of the problem, the inclusion of constraints or the adoption of a more detailed formulation might lead to challenges that call for novel solution approaches, as in the case of the MSPPMTW.

The MSPPMTW is a formulation motivated by modelling real-world transportation problems, in particular two main properties that are often present at the same time. The first such property of interest in transportation-related problems is the presence of multiple objectives. Minimising travel time is an obvious goal in transportation systems in pursuit of customer satisfaction and efficient utilisation of the available infrastructure. At the same time, the process of transportation is resource-intensive and the sector is the main contributor to greenhouse emissions, therefore it is important to minimise energy consumed from an environmental standpoint (Danloup et al. 2015, Bleviss 2021). Minimising emissions and the economic cost of the operation are not always perfectly aligned, however, they are usually correlated. Different transportation systems have different other objectives worthy of consideration, in relation to safety, aesthetics of the route, or other interests of shareholders involved. When there are multiple relevant objectives, they are often conflicting, thereby justifying a multi-objective approach. Considering multiple objectives means that generally there won't be a single solution that is the best according to all objectives. Therefore, multi-objective Shortest Path Problems (MSPPS) have multiple optimal solutions, the so-called Pareto-optimal solutions, also called Pareto front or efficient solutions. A solution is Pareto Optimal if it cannot be improved upon in terms of any objective without hindering at least one of the other objectives (see Definition 2.1.8).

The second property of interest in transportation-related problems is the presence of time window constraints. Real-world problems usually have a number of constraints associated with them. Constraints can be imposed by contracts, regulations, other vehicles or can simply be a consequence of physics. In this thesis, a particular type of constraint, the time window constraint is given special attention, which usually represents time-dependent restrictions on the movement of vehicles. Time windows specify time periods for certain parts of the network when that part is unavailable for routing purposes. The source of such constraints in transportation problems can be avoiding other vehicles, contracts, timetables or signalised intersections, however, the intermittent availability of a drawbridge could also be represented through time windows. The

presence of these restrictions makes it necessary to optimise movements of vehicles with regards to time, not just to space, introducing the scheduling aspect of the problem. The scheduling aspect is dependent on the routing aspect and vice versa, therefore they need to be integrated to find the most efficient solutions.

When considering real-world optimisation problems, researchers often ignore some details in the mathematical formulation, so the problem can be tackled by the current state-of-the-art solution approaches. A more detailed model might offer a more realistic depiction of the problem, even if it is harder to solve. In the case of shortest path problems with time windows, including the detail of multiple possible parallel arcs can be expected to lead to a higher chance of finding feasible solutions, by increasing flexibility. Even when time windows are not present, parallel arcs can be justified by providing a more thorough picture of the available trade-offs between objectives. However, the more detailed modelling approach comes at a price of an increased search space. In order for the multigraph modelling approach to be of practical use, there is a need for solution algorithms that can deal with the increased search space in an efficient way.

Shortest path problems are most often solved by Dijkstra's algorithm (Dijkstra et al. 1959), a classic algorithm used either to find the shortest path from a given node to a single other node, or all other nodes in the network (see pseudo-code in Section 2.2. Dijkstra's algorithm and its multi-objective variants are exact, meaning that they find all Pareto optimal solutions (given the problem satisfies some general conditions), which, in some cases, takes too long. The computational effort might vary significantly when solving different MSPP *instances*, particular realisations of the problem that include all information, such that a solution can be computed. Most instances are solved relatively quickly, while there are some outliers that take significantly longer. There is no way to tell before actually executing the algorithm if a given instance is an outlier or not. However, there are some indications that certain characteristics of the graph structure and the objective values relate to more difficult problems. The problem of computational effort becomes more pronounced with higher numbers of objectives considered and in the case of the Multigraph MSPP with higher numbers of parallel arcs, as the parallel arcs add a new layer of complexity. Therefore, when the computational time is limited for making the routing decision, exact algorithms are often not the best choice. Even if most of the problems

can be solved within the time budget, the risk of ending up with no solutions at all is generally not acceptable.

To be able to deliver solutions in a given time budget, it is necessary to sacrifice solution quality. This is often acceptable, as the purpose of finding all Pareto Optimal solutions is to map the trade-offs between the objectives and choose a single solution accordingly. A smaller set of solutions can be sufficient for this as long as they are representative of the real Pareto front. A class of approximate algorithms often used for finding a good compromise between running time and solution quality are metaheuristics (Blum & Roli 2003). Metaheuristics are stochastic optimisation algorithms that are inspired by natural processes such as evolution, the cooling of metal or the observed behaviour of animals. Metaheuristics have been employed for MSPP on simple graphs (Ahn & Ramakrishna 2002, Lin & Gen 2008), but are even more relevant for the MSPPMTW, because of the higher complexity caused by the parallel arcs. The number of possible routes on the multigraph that follow a given node sequence grows exponentially with the number of parallel arcs. One of the metaheuristic algorithms most often employed for SPPs is the Genetic Algorithm (GA) (Holland 1992).

The multigraph approach is not yet well established in the literature, with only a handful of studies embracing the benefits and challenges that are offered. An exception is the field of the multi-modal transportation problem, where the multigraph modelling approach is necessary to represent different modes of transport (Xiong & Wang 2014). The concept has also been proposed for multi-objective routing problems where legs of the route can be completed in multiple ways that give different trade-offs between the objectives (Garaix et al. 2010), such as driving on different road sections with more or less turns, or travelling at different speeds. One specific problem where the multigraph formulation has also been proposed is the airport ground movement problem (Chen, Weiszer, Locatelli, Ravizza, Atkin, Stewart & Burke 2016). A taxiing aircraft is the most fuel-efficient at certain speeds, and on routes with fewer turns. For this reason, there is a trade-off between taxi time and fuel consumption (Ravizza et al. 2013). The multigraph modelling approach was shown to provide better solutions than the simple graph approach in (Weiszer et al. 2020). A similar trade-off is often present when routing different vehicles (Gallet et al. 2018), suggesting wider applicability for algorithms devised for airport ground movement. The need for an integrated routing and scheduling approach also applies to

a broader set of problems, such as routing in maritime transportation (Wen et al. 2017), train operations (Yang et al. 2016) and transport of hazardous materials (Meng et al. 2005).

Widespread adoption of the multigraph modelling approach can only be expected after the maturation of the solution approaches for the MSPPMTW. Benchmark sets have a central role in designing new algorithms for any problem. The wide adaptation of a standard benchmark set is often followed by the rapid advancement of a field (Machuca Sánchez et al. 2012). Benchmark problem sets are essential for understanding the capabilities of the considered algorithm before deployment and they provide a basis for fair comparison of different algorithms. There is a lack of benchmark sets for the MSPPMTW in the literature that would facilitate progress in the solution of the relevant real-world transportation problems.

An instance of the MSPPMTW is a multigraph with cost vectors assigned to each arc (different for parallel arcs between same nodes) and time windows (same for parallel arcs between same nodes), and two nodes nominated as the origin and destination nodes. There are some benchmark sets proposed for SPPs (Skriver & Andersen 2000), that can be further adapted to the MSPPMTW, however, these include networks with similar properties to each other. The results of the field of Network Science can be used to diversify the network structures included in the benchmark set, thereby making the benchmark set more general, and a better reflection of real networks. This is also motivated by the observations that the network type affects which solution method is best suited for the MSPP (Raith & Ehrgott 2009).

1.2 Research Goals

The aims of this thesis can be divided into two parts, (1) devising a benchmark problem set for the MSPPMTW and (2) proposing a metaheuristic solution algorithm for the MSPPMTW.

The design of the benchmark problem set can be broken down to:

1. Understanding how different features contribute to problem difficulty for the Multigraph MSPP.
2. Generating fairly realistic instances of the MSPPMTW with controllable difficulties, based on the insights gained in the last step.
3. Demonstrating the use of the proposed benchmark set.

The task of proposing a metaheuristic solution algorithm for the MSPP and MSPPMTW can be broken down into:

1. Taking account of current solution approaches in the literature.
2. Exploring different representation methods and operators.
3. Testing the proposed algorithm on artificial and real world problem instances.

1.3 Contributions

Based on the research goals laid out in the previous section, the following contributions are included in the rest of the thesis:

1. The thesis examines what properties of MSPPMTW instances make such a problem computationally expensive to solve. A predictive model is proposed for assessing the expected computational effort required to solve Multigraph MSPP instances in Chapter 3, and the effect of the time window constraints is investigated separately in Chapter 4. The proposed predictive model based on an ensemble learning method can predict the magnitude of problem difficulty with reasonable accuracy. It is verified that the properties mentioned in the literature for the MSPP do affect the computational effort required also for the Multigraph MSPP. The position of the nodes in the network and a feature concerning costs were also identified as important, which are not yet discussed in the literature. The number of parallel arcs was found to be one of the most important features, which highlights the need for special consideration of the multigraph problem.
2. A benchmark problem instance generation method is proposed for the MSPPMTW in Chapter 4. The generator is capable of creating a wide range of problem instances, with regard to different network structures, costs and properties of the time windows.
3. A metaheuristic solution approach is proposed, based on comparing multiple representation techniques and genetic operators. The algorithm is systematically evaluated against a state of the art exact algorithm on three different versions of the problem:
   (a) the Multigraph MSPP without time constraints is considered in Chapter 5,

(b) the MSPPMTW, with instances generated by the generator proposed in Chapter 4 is considered in Chapter 6,

(c) the airport ground movement problem, a real-world application exemplifying the kind of problem modelled by the MSPPMTW is considered in Chapter 7

The observed performance of the proposed algorithm shows that it provides a reasonable compromise between computational time and solution quality in all mentioned cases, including on real-world data. There is a reasonable agreement between the findings regarding the different variants of the proposed algorithm on different versions of the problem.

1.4 Publication List

The publications resulting from the research described in this thesis are listed below.

1. Lilla Beke, Michal Weiszer, and Jun Chen. A comparison of genetic representations for multi-objective shortest path problems on multigraphs. In European Conference on Evolutionary Computation in Combinatorial Optimisation (Part of EvoStar), page 35-50. Springer, 2020. 47, 101, 102
2. Lilla Beke, Michal Weiszer, and Jun Chen. A comparison of genetic representations and initialisation methods for multi-objective shortest path problems on multigraphs. Springer Nature Computer Science, 2(3):1-22, 2021. 81, 84, 99
3. Tianci Zhang, Lilla Beke, Songwei Liu, Michal Weiszer, Jun Chen. An extended memetic algorithm for multi-objective routing and scheduling of airport ground movements with intermediate holding. 2022 IEEE Congress on Evolutionary Computation (CEC), 1-8
4. Lilla Beke, Lourdes Uribe, Adriana Lara, C.A. Coello Coello, Michal Weiszer, Edmund K. Burke, Jun Chen. Routing and Scheduling in Multigraphs with Time Constraints - A Memetic Approach for Airport Ground Movement. 2023 IEEE Transactions on Evolutionary Computation (Early Access)

The contents of Chapter 5 are based on the publication (Beke et al. 2021), and the contents of Chapter 7 are based on the publication (Beke et al. 2023). All numerical tests are performed

on Queen Mary's Apocrita HPC facility (Z n.d.*b*).

1.5 Thesis Outline

Chapter 2 includes some preliminaries for the thesis and details the research background and context. Chapter 3 includes the investigation of the computational efforts required by different problem instances of the MSPP on Multigraphs without time constraints. The proposed benchmark generation technique for the problem with time windows in Chapter 4. Chapter 5 introduces a metaheuristic algorithm for the MSPP on Multigraphs without time constraints. Chapter 6 extends the above mentioned algorithm to consider time window constraints and tests it on benchmark instances created by the generation method proposed in Chapter 4. Chapter 7 provides results about applying the proposed metaheuristic algorithm to a real-world application. Conclusion is drawn in Chapter 8.

Chapter 2

Background

This chapter introduces the background and main concepts providing the basis for this thesis. A brief summary of concepts from graph theory and optimisation is presented in section 2.1. The literature review follows. The history and variants of the shortest path problems are summarised in section 2.2. A review of the enumerative and metaheuristic algorithms for solving the Multi-objective Shortest Path Problem (MSPP) follow in sections 2.3 and 2.4. The airport ground movement problem and other real world applications are reviewed in section 2.5. Section 2.6 is concerned with methods of evaluating the solution quality obtained by different algorithms. The state of the art in testing algorithms and benchmark sets is presented in section 3.2.

2.1 Preliminaries

This section includes the definitions from graph theory and multi-objective optimisation that the thesis builds on.

2.1.1 Notions of Graph Theory

A graph is a mathematical structure that represents information about pairwise relationships between objects that are represented by nodes. Networks usually refer to graphs such that the nodes represent real-world objects, and arcs have attributes associated with them. One well known example is social networks, however, applications are countless, from electrical engineering to neuroscience. Networks can represent transportation systems by representing the

connections between important places in the transportation system. More details are discussed in Section 3.2.2. Each network has an underlying graph, where the relationship of nodes and arcs to the real-world is ignored. In this thesis the term "graph" and "network" are used somewhat interchangeably, however, the term "graph" is usually used in a more theoretical sense. An illustration of a simple graph is shown in Figure 2.1. Nodes are represented by numbered circles, the arcs are represented by lines connecting the nodes.

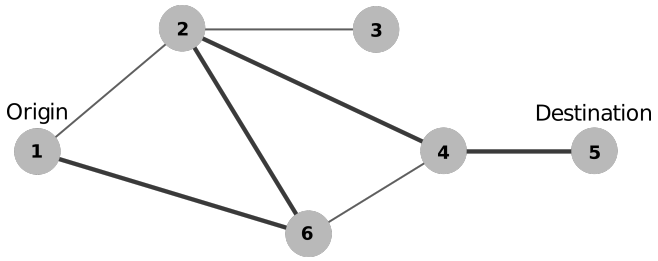


Figure 2.1: An example of a simple graph and a path in it from node 1 to node 5, indicated by the bold line.

This thesis is mainly concerned with weighted directed graphs. Undirected graphs are conceptualised as each undirected arc being replaced with directed arcs in both directions. Given the multi-objective problem considered, the definition of a weighted directed graph (Definition 2.1.1) is given with vector weights, where the weights have q components. The more common case of scalar weights is included when $q = 1$. The weights are assumed to be positive.

Definition 2.1.1. A weighted directed graph $G = (V, A, w)$ is defined by the set of nodes (or vertices) V and the set of arcs A, where $A \subset \{(u,v) \mid (u,v) \in V \times V \text{ and } u \neq v\}$ is a set of ordered pairs of nodes and w is a weight function $A \to \mathbb{R}_+{}^q$.

In directed graphs, the two end nodes of the arc are referred to as its *head* and *tail*, where the direction is from the tail to the head.

In some cases it makes sense to include multiple parallel arcs between the same nodes, such graphs are called multigraphs. An illustration of a multigraph is shown in Figure 2.2. A definition is given in Definition 2.1.2, which highlights that the set of arcs is A_M does not need to be composed of unique pairs of nodes, however, each element of A_M needs to be associated with a pair of nodes in G_M, as specified by function ϕ. Self loops (arcs such that both end nodes are the same node) are not allowed in this thesis, however, it is common to allow self loops in multigraphs.

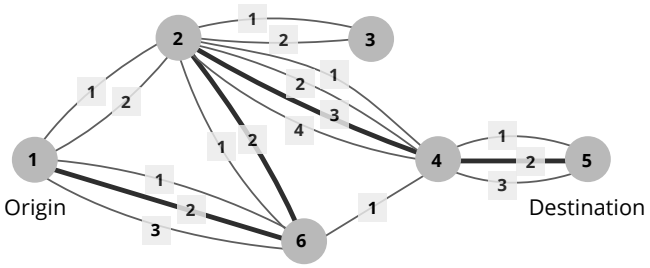


Figure 2.2: An example of a multigraph network and a path in it from node 1 to node 5, indicated by the bold lines.

Definition 2.1.2. A weighted directed multigraph $G_M = (V, A_M, \phi, w)$ is defined by the set of nodes V and the set of multi-arcs A_M, where $\phi : A_M \to \{(u,v) \mid (u,v) \in V \times V \text{ and } u \neq v\}$ maps each element of A_M to an ordered pair of nodes and w is a weight function $A_M \to {\mathbb{R}_+}^q$.

For the purposes of this thesis, the set of parallel arcs between nodes u and v are denoted $A(u,v)$. An arc $a \in A_M$ in a multigraph G_M can be specified as $a = (u,v,k)$, where u and v are its end nodes and k is an index, $1 \leq k \leq |A(u,v)|$, if $|A(u,v)| > 0$.

The weights in the context of this thesis correspond to cost vectors, in particular the first component of the weights $w(u,v,k)$ specifies the objective value of the kth arc between nodes u and v according to the first objective.

Different attributes can also be associated with each arc, such as time windows. Time windows ($\mathscr{F}_a$) signify when the arc a is available, and in this thesis they are defined for pairs of nodes, and not for individual parallel arcs, such that $\mathscr{F}_a = \mathscr{F}_{a'} \quad \forall a, a' \in A(u,v)^2$.

Definition 2.1.3. A set of time windows $\mathscr{F}_a$ assigned to an arc $a = (u,v,k) \in A_M$ is defined as $[[t_1,t_2],[t_3,t_4],...,[t_{s-1},t_s]]$, where

$$t_1 = 0,$$

$$t_s = \infty,$$

$$s = 2 * |\mathscr{F}_a|,$$

$$t_i < t_{i+1} \quad \forall i \in [1,s].$$

The set of time windows for the whole graph G_M is denoted $\mathscr{F}$.

$$\mathscr{F} = \{\mathscr{F}_a \mid a \in A_M\} \tag{2.1}$$

To be able to interpret the shortest path problem, the definition of a path is given in both simple graphs and multigraphs. In the simple graph case, it is enough to list nodes to specify a path. In the case of a multigraph, arcs with the parallel indices are needed, because there can be multiple arcs between the same nodes, and the differentiation is important. In this thesis, Definition 2.1.5 is used, because the simple graph case can be interpreted as a special case of a path in a multigraph.

Definition 2.1.4. A path Π in a simple graph $G = (V,A)$ is a sequence $(v_1, v_2, \ldots, v_{|\Pi|})$, where

$$\begin{aligned} v_i \in V, \quad & \forall i \in [1, |\Pi|], \\ (v_i, v_{i+1}) \in A, \quad & \text{and} \\ v_i \neq v_j \quad \forall i, j \in [1, |\Pi|]^2, \quad & \text{if } i \neq j. \end{aligned}$$

Definition 2.1.5. A path Π in a simple graph $G_M = (V,A_M)$ is a sequence $(a_1, a_2, \ldots, a_{|\Pi|})$, where

$$\begin{aligned} a_i \in A_M, \quad & \forall i \in [1, |\Pi|], \\ head(a_i) = tail(a_{i+1}) \quad & \forall i \in [1, |\Pi|-1], \\ v_i \neq v_j \quad \forall i, j \in [1, |\Pi|+1]^2, \quad & \text{if } i \neq j, \\ v_i = tail(a_i) \quad & \forall i \in [1, |\Pi|], \text{ and} \\ v_{|\Pi|+1} = head(a_{|\Pi|}). & \end{aligned}$$

$\Pi_{1,i}$ denotes the subpath of Π that includes the first $i-1$ arcs and the first i nodes of the path. (Accordingly, $\Pi_{1,1}$ denotes the subpath that includes only the first node.)

The first node of a path is referred to as the origin node (v_O), and the last node as the destination node (v_D). There are a number of properties that can be defined on paths, such as the sum of the weights associated with the arcs of Π, referred to as the weight of path Π.

$$w(\Pi) = \sum_{a \in \Pi} w(a) \tag{2.2}$$

When the length of a path is mentioned, it often refers to the sum of the length attribute of the

arcs in the path, when there is a length attribute defined. This is why there is a separate term, *hopcount*, for specifying the number of arcs included in a path. The term hopcount can also refer to the minimum number of arcs to be traversed in a graph to get to one specified node from another.

The set of all possible paths in a graph G between nodes v_O and v_D are denoted as $\mathscr{P}_{G,O,D}$.

Definition 2.1.6. A Shortest Path Problem (SSP) in the case of scalar weights the goal is finding a path Π such that $w(\Pi) \leq w(\Pi'), \quad \forall \Pi' \in \mathscr{P}_{G,O,D}$.

2.1.2 Multi-objective optimisation

When there are multiple objectives to be considered ($q > 1$), there is not necessarily a best solution, as there is only a partial order relation defined over cost vectors, which is called *dominance*.

Definition 2.1.7. A cost vector $\overline{c}$ *dominates* the cost vector $\overline{c}'$, indicated as $\overline{c} \prec \overline{c}'$, if and only if $\forall i c_i \leq c'_i$ and $\overline{c} \neq \overline{c}'$.

Definition 2.1.8. Given a set of cost vectors C, the cost vector $\overline{c} \in C$ is considered non-dominated if $\nexists \overline{c}' \in C$, such that $\overline{c}' \prec \overline{c}$.

Solutions for a problem with non-dominated cost vectors are called Pareto-optimal.

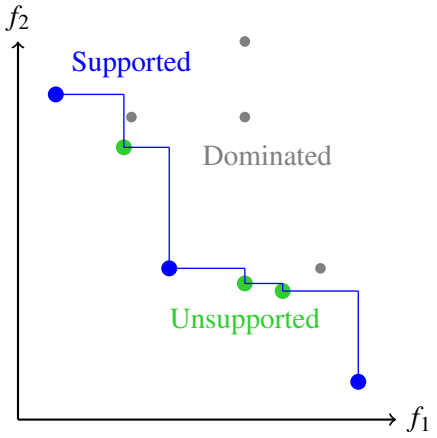


Figure 2.3: Illustration of supported and unsupported solutions on a Pareto front. Blue circles represent supported solutions, green circles represent unsupported solutions that are not on the convex hull, and gray points are dominated solutions.

In the MSPP, the goal is to find the set of paths among $\mathscr{P}_{G,O,D}$ with non-dominated cost vectors according to w.

Definition 2.1.9. A Multi-objective Shortest Path Problem (MSSP) in the case of vector weights the goal is finding a path Π such that $w(\Pi) \prec w(\Pi'), \quad \forall \Pi' \in \mathscr{P}_{G,O,D}$.

Shortest path problems are combinatorial optimisation problems (Ehrgott & Gandibleux 2000), where the name refers to the discrete nature of these problems. An important concept for combinatorial optimisation problems is the concept of *supported solutions*.

Definition 2.1.10. A single objective combinatorial optimisation problem with a sum objective is defined as a problem with a feasible set X such that $X \subseteq 2^Y$, where Y is a finite set $Y = y_1, \ldots, y_{|Y|}$, and 2^Y is a power set. The objective function has the general form $z(x) = \sum_{y \in x} w(y)$, where $x \in X$ and $w : Y \to \mathbb{Z}$ is a scalar valued weight function, and z should be minimised.

In multi-objective combinatorial optimisation problems, there are several objective functions, and the goal is to find the Pareto-optimal solutions, considering all objective functions. Among the Pareto-optimal solutions for a combinatorial optimisation problem, supported and non-supported solutions are differentiated.

Definition 2.1.11. Supported solutions can be obtained by aggregating objectives through weighted sums. These solutions are positioned on the convex hull in objective space.

Non-supported non-dominated solutions also exist as a consequence of the discrete nature of combinatorial optimisation problems. Given that non-supported non-dominated solutions cannot be found by the relatively simple technique of weighted aggregation of objectives, their existence has implications for the solution approaches. Figure 2.3 shows an example of supported and non-supported Pareto-optimal solutions.

2.2 Shortest path problems

The Shortest Path Problem (SPP) in general deals with finding minimum weight paths in a graph between two nodes. The SPP consistently attracts significant attention from researchers in the fields of Operations Research and Computer Science and still remains a foundational problem in

combinatorial optimisation, as it has been throughout the second half of the last century (Gallo & Pallottino 1986, Cherkassky et al. 1996). There are a number of applications in different fields from communications (Chitra & Subbaraj 2012) to different types of transportation, such as multi-modal transportation (Xiong & Wang 2014, Yu & Lu 2012), ride-sharing (Coltin & Veloso 2014, Enzi et al. 2020), hazardous materials transportation (Caramia et al. 2010, Meng et al. 2005) and dangerous goods transportation (Li et al. 2013), electric vehicle routing in urban settings (Tu et al. 2020) or taxiing operations of aircraft (Chen, Weiszer, Locatelli, Ravizza, Atkin, Stewart & Burke 2016). The differences in the applications listed above motivate the development of specialised algorithms and studies, while the fundamental problem of finding a shortest path between two nodes is at the core of all of these applications. The high relevance for many real-world problems keeps SPPs in the spotlight and facilitates the progress of the field.

In its simplest form, the shortest path problem can be considered on a graph without weights. This case has some real-world relevance, for example in social networks, most of us are familiar with social media websites giving information about the length of a friendship-chain connecting two people. This simple formulation is of limited utility in route planning. It might inform us about the number of transfers required in a multi-modal transportation setting, but in order to receive information about travel time or distance, the arcs of the network need to be associated with attributes generally called weights. Weighted graphs allow for including more details in the model, and therefore answering questions of minimal travel time becomes possible.

Such questions - finding shortest paths according to scalar-valued weights - are referred to as Single-objective Shortest Path Problems. The two most influential solution algorithms, Dijkstra's (Dijkstra et al. 1959) and Bellman's (Bellman 1958) algorithm have been published in the fifties for solving such problems. The ideas laid down then are still at the core of most modern solution methods. The main contributions in the next decades were largely aimed at refining data structures for efficient implementation of shortest path algorithms (Gallo & Pallottino 1986).

Dijkstra's algorithm visits each node in a strategic way starting form the origin node to establish the shortest distances. The pseudo-code is displayed in Algorithm 1. It finds the shortest path between nodes O and D in graph $G = (V, E)$ according to single-objective weights w. The map "dist" stores the shortest distance from the source node O to every vertex, which

is being updated by the algorithm. The map "prev" stores the previous node in the shortest path from O to every other vertex, and is used to decode the shortest path after the algorithm finishes. Q is a set of all nodes of G that have not been visited yet at any point. The algorithm initialises distances in "dist" to all nodes as infinity, with the origin node set to zero. It then iteratively selects the unvisited node with the smallest distance, updating the shortest paths for its neighbours, and halts when the destination node is selected.

Algorithm 1 Dijkstra's Algorithm (Single Destination)

Input: $G = (V,E), w, O, D$
Output: $dist[D], prev$

```
1:  Initialize all dist[v] to ∞ for all v ∈ V
2:  Initialize all prev[v] to undefined for all v ∈ V
3:  dist[O] ← 0
4:  Q ← V
5:  while Q ≠ ∅ do
6:      u ← node in Q with smallest dist[u]
7:      Remove u from Q
8:      if u = D then
9:          break
10:     end if
11:     for all v ∈ neighbors of(u) do
12:         alt ← dist[u] + w(u,v)
13:         if alt < dist[v] then
14:             dist[v] ← alt
15:             prev[v] ← u
16:         end if
17:     end for
18: end while
19: return dist[D], prev
```

The SSPP formulation is only capable of dealing with one objective, which as pointed out by Hansen (Hansen 1980) is insufficient in some situations. In the following, it was recognised that in order to provide decision makers with information of practical utility, multiple aspects of the problem need to be taken into account. Such aspects can be economic, ecological and social in nature, which are often not aligned. For example, in hazardous material shipment, the risk level of the route is clearly important as well as the length. The Multi-objective Shortest Path Problem (MSPP) describes these kind of problems in general, where weights of the graph are vectors and each component of the vectors describes an objective. An example of an MSPP would be finding a travel route in a city, while keeping both travel time and expense into account. The

MSPP defines the goal as finding the Pareto-optimal set of solutions (definition 2.1.8), which include all feasible solutions which cannot be improved on in terms of any objective without hindering another objective.

Initially, the MSPP was often reduced to the single-objective form by assuming a utility function on the objectives, often a weighted aggregation (Loui 1983). For discrete optimisation problems such approaches cannot find some of the Pareto-optimal solutions, known as *supported* solutions (see Definition 2.1.11). Another approach for using single-objective techniques for multi-objective problems is optimising for a single objective, while considering other objectives as constrained to some satisfactory level. This approach fits into the category of Resource Constrained Shortest Path problems (Irnich & Desaulniers 2005). These techniques do not guarantee to find all Pareto-optimal solutions for the MSPP, and therefore they can provide only partial information for decision making in practical problems.

Solution algorithms that can guarantee to find the whole Pareto front for a given problem when enough time is available are known as *exact algorithms*. The most prominent categories of exact algorithms proposed for the MSPP are labelling methods, ranking methods and two-phase methods (Clímaco & Pascoal 2012). The first labelling algorithm for the MSPP was pioneered in (Hansen 1980) extending the idea of Dijkstra's algorithm to two objectives. The labelling approaches and the adaptation to multiple objectives are reviewed in more detail in Section 2.3.

Ranking methods (Climaco & Martins 1982) for the bi-objective case generate shortest paths in non-decreasing order regarding one of the objectives, until the best solution according to the other objective is found. Dominated solutions are eliminated through the process. Ranking methods have been found to perform worse compared to other approaches in multiple studies (Ehrgott & Gandibleux 2000, Huarng et al. 1996).

Two-phase methods (Mote et al. 1991) find supported solutions in the first phase and non-supported solutions in the second phase. In the second phase usually, a labelling or ranking method is used. The advantages of the two-phase approach are that in the first phase, a single objective algorithm can be used, which are more effective than multi-objective approaches. In the second phase, a multi-objective algorithm is used on a limited search space, that has not yet been explored in the first phase. The efficiency of the above approaches has been compared in an extensive empirical study in (Raith & Ehrgott 2009), where labelling methods and two phase

methods were found to be the most efficient in most cases.

The above exact algorithms are not suitable for some MSPP instances when it is necessary to find solutions in a short time budget. Another broad family of algorithms often employed for solving the MSPP are metaheuristics, detailed in Section 2.4. These algorithms in general produce solutions faster, however, they cannot give any guarantee about the solution quality obtained. Nevertheless, metaheuristics are considered to be able to provide a good compromise between the speed of finding solutions and the solution quality (Blum & Roli 2003, Chitra & Subbaraj 2012).

Constraints are present in most real world problems. The carrying capacity of a vehicle, the need for refuelling, contracts and opening hours, traffic lights, and movements of other vehicles all constrain transport operations. In the case of time-tabled transportation, the time constraints are even more prominent. There are few studies of MSPP with time constraints. Most studies on constrained shortest path problems are considering resource constraints, and they are overwhelmingly about single objective problems (Shi et al. 2017). This thesis focuses on shortest path problems with time window constraints, a particular type of constraint that restricts the availability of parts of the transportation network in time. Pugliese and Guerriero (Pugliese & Guerriero 2013) (2013) reviews resource constrained shortest path problems, including the single-objective shortest path problem with time windows (SSPTW). In that review work, the four listed studies about the SSPTW (Desrosiers et al. 1983, Desrochers & Soumis 1988, Ioachim et al. 1998, Powell & Chen 1997) are all from the last century. Moreover, these studies considered a single time window for each node in the network. The general case of an arbitrary (finite) number of time windows is considered in (Chen, Weiszer, Locatelli, Ravizza, Atkin, Stewart & Burke 2016, Weiszer et al. 2021) in relation to the airport ground movement problem, however, the solution approaches proposed cannot guarantee optimality when time windows are present, because the optimality principle does not hold.

Most studies of the MSPP assume a simple graph structure. It was argued in (Garaix et al. 2010), that a multigraph formulation is often justified in multi-objective routing problems, because in many real-world examples vehicles have multiple ways of traversing an arc in the transportation network. Using a simple-graph model ignores all but one out of the potential non-dominated arcs between neighbouring nodes and thereby limits the search space, so that it does

not reflect the whole range of real possibilities. Moreover, as argued in (Chen, Weiszer, Locatelli, Ravizza, Atkin, Stewart & Burke 2016), when the parallel arcs of a multigraph correspond to ways of traversing the arc with different travel times, they provide more flexibility for finding solutions that comply with time windows. It is worth noting that limiting the search space by using a simple graph model has the obvious advantage of a smaller search space, therefore, the benefit of the multigraph model needs to be carefully studied. Most solution methods can be easily adapted to multigraphs by iterating over the parallel arcs. Metaheuristic solution approaches might be promising for their ability to efficiently explore large search spaces.

The complexity and tractability of shortest path problems depend on the formulation. The single-objective version can be solved in polynomial time. Hansen (Hansen 1980) has shown that the MSPP is intractable in the worst case, by showing a family of graphs for which the number of Pareto-optimal paths grows exponentially with the number of nodes. The MSPP with two objectives was proven NP-hard even on simple graphs without time constraints (Serafini 1987). It was argued in (Müller-Hannemann & Weihe 2001) that this high computational complexity is rarely reflected in practice, however, in the case of short time budgets, even polynomial running time might cause a problem. Also, the multigraph approach has further increased search space and computational complexity compared to the simple graph approach.

Looking through the evolution of shortest path problems, a trend toward more complex formulations is conspicuous. This can be explained by the variety of real-world problems encompassed and the pursuit of more detailed ways of modelling the real-world problems, such as including more objectives, constraints and formulating the problem on a multigraph. More detailed modelling often comes at a price of increasing the computational complexity of the problem, therefore the advancements in computational technology were indispensable in the advancement of the field. With the availability of more computational power, solution of more complex forms of the problem become attainable.

2.3 Enumerative algorithms to solve the shortest path problem

Dijkstra's algorithm (Dijkstra et al. 1959) (1959) is the most well-known algorithm for shortest-path problems. The algorithm maintains a set of provisional distances from the origin node to

each of the nodes, which are called *labels*. Labels in single objective problems at each point in the search process reflect the shortest distance to the given node found so far. Dijkstra's algorithm uses a best first strategy to expand labels with the current minimal cost, which means considering each of the possible next arcs for the continuation of the sub-path that belongs to the label. Labels are updated when a shorter path has been discovered to the given node. Once a node has been expanded, its label is final. That is why Dijkstra's algorithm belongs to the category of label-setting algorithms as opposed to label-correcting algorithms such as the Bellman-Ford algorithm (Bellman 1958), which are better suited for shortest path problems with negative weights.

In the case of multi-objective problems keeping track of all paths of interest becomes more labour-intensive. In the multi-objective case, labels store vectors, and only those labels can be discarded, or *pruned*, which are dominated by the cost of some other path arriving at the same node. Therefore, in general, nodes will have multiple labels associated with them at the same time that correspond to non-dominated paths that reach the given node.

Heuristic search such as the A* (Hart et al. 1968) algorithm enables space and time savings when solving shortest path problems, by exploiting domain-specific knowledge. The A* algorithm uses estimates of the distance to the destination node to improve the efficiency of the search. The estimates are usually precomputed, and therefore do not contribute to the computational burden. The heuristic function $h(v)$ denotes the estimated cost of a path from node v to the destination node. The cost of a path from the origin node to node v is denoted $g(v)$. Then the labelling algorithm calculates the estimated cost of a path from the origin node to the destination node via node v as $f(v) = g(v) + h(v)$, which is called the *evaluation function*. Partial solutions are then pruned according to the value assigned to them by the evaluation function. The heuristic function needs to underestimate the distance in order for the A* algorithm to remain exact for SPPs. Such heuristic functions are also called *admissible*. In order for the algorithm to only expand nodes once, the heuristic function has to be *monotone*. A heuristic function is considered monotone, when $h(v) + c(v,v') \leq h(v') \quad \forall v, v' \in A$, where $c(v,v')$ is the actual cost of the arc (v,v').

There are multiple strategies possible for extending label-setting solution methods to the multi-objective problem. One possibility is label-selection, which is expanding one label at a

certain node by all arcs of the node, or node-selection, when all labels of a node are expanded at the same time. The algorithm needs to keep track of the labels pending evaluation and the ones already evaluated. Similarly to the single objective problem, the set of labels at the destination node corresponds to the Pareto-optimal solutions when the algorithm finishes. The set of paths can be recovered through a backtracking process, from the information stored during the search process. It was found in (Guerriero & Musmanno 2001) that it depends on the problem instance if label-selection or node selection performs better, and the same can be said for label-setting and label-correcting approaches.

Extensions of multi-objective labelling algorithms based on the the A* algorithm such as the New Approach to Multi-objective A* (NAMOA*) (Mandow et al. 2005, Mandow & De La Cruz 2010) are able to make use of heuristic information and accelerate the search process, while still finding the whole Pareto front (when the principle of optimality holds). NAMOA* is an improvement over a previous version of multi-objective A* algorithm MOA*, as shown by the empirical studies (Machuca et al. 2010). The main difference lies in the label selection strategy used by NAMOA*, as opposed to the node selection strategy of MOA*. Node selection more closely resembles the original A* algorithm, although it is less efficient. An example for an admissible and monotone heuristic function for the multi-objective problem is the one proposed in (Tung & Chew 1992). It is defined as, $h_{TC}(n) = (c_1(n), c_2(n), ..., c_q(n))$, where $c_i(n)$ is the optimal scalar cost of a path from node n to the destination node, considering only the ith cost component.

Note that pruning dominated costs is based on the principle of optimality, and is an integral part of labelling algorithms. In the case of some constrained problems, or when the costs do not satisfy the additivity property, pruning dominated partial solutions might lead to disposing of useful solutions, as will be discussed in Section 4.2.1 and 7.2.4. Generally, the above approaches are not guaranteed to find all possible solutions for the MSPPMTW.

2.4 Metaheuristic algorithms for the shortest path problem

In practical settings, it is often important to find a good representation of the Pareto front in a given time budget, even if it means some compromise on solution quality. Metaheuristics are

popular in such situations compared to exact approaches, because they are able to provide good enough solutions in short times, and ideally when given enough time they can converge on the optimal solutions.

Metaheuristics are a relatively young research area, the term was introduced by Glover (Glover 1986), to describe how a higher level process is superimposed on a *heuristic*, where heuristic stands for a method without formally demonstrable convergence properties. Some well-known metaheuristic algorithms are Ant Colony Optimisation, Genetic Algorithms, Tabu Search and Simulated Annealing. Multiple definitions have been proposed for metaheuristics, however, a clear consensus has not formed. In a review of metaheuristics for combinatorial optimisation problems (Blum & Roli 2003), Blum identifies properties that metaheuristic algorithms share. The main points are summarised as follows. Metaheuristic algorithms are not problem specific, and usually non-deterministic, however, they might make use of mechanisms that exploit domain-specific knowledge. Their goal is to efficiently explore the search space to find optimal solutions or a good representation of them. For this reason they might incorporate mechanisms to escape from local optima. The different mechanisms are governed by a high-level strategy.

Metaheuristic algorithms strive for a balance in diversification and intensification. When this balance is not achieved, metaheuristics can be prone to premature convergence with too little or inefficient diversification on the one hand, or might default to random search with not sufficient intensification. The advantage of using metaheuristic algorithms apart from the expected low computational resource requirements is their adaptability. Using little information about the problem compared to exact approaches makes metaheuristic approaches easily adaptable to changes in the formulation of problems, such as adding constraints or removing assumptions.

There were a number of different metaheuristic algorithms proposed for SSPs. Genetic Algorithms (GA) are one of the main metaheuristics widely applied to MSPPs (Pangilinan & Janssens 2007, Chitra & Subbaraj 2012, Li et al. 2013) and to multimodal transportation problems (Dib et al. 2018, Xiong & Wang 2014). Because of the prevalence and variety of the GA solution methods for the MSPP, GAs were chosen for further exploration in this thesis. Also, (Liu et al. 2020) in a recent review of multi-objective metaheuristics for discrete optimisation problems, it was found that GAs are the most popular choice for such problems. A promising alternative could be Ant Colony Optimisation, which is well suited for graph-problems, and also

popular for solving SPPs (Glabowski et al. 2012, Ghoseiri & Nadjari 2010, Hassan et al. 2020). The comparison with GAs is mentioned in Chapter 8 as a future direction, as this thesis focuses on representation techniques and initialisation methods in terms of the metaheuristic solution methods. Particle Swarm Optimisation was also used to solve SPPs in (Mohemmed et al. 2008), and in (Marinakis et al. 2017) hybridised with Variable Neighbourhood Search.

2.4.1 Genetic Algorithms to solve shortest path problems

Introduction to GAs

GAs were proposed in (Holland 1992), and they belong in the wider category of Evolutionary Algorithms. GAs are population based methods, where a population of candidates is maintained and modified throughout multiple iterations by genetic operators. Each iteration includes well defined stages, including the execution of genetic operators and selecting the candidates to form a population in the next iteration.

The terminology used when talking about GAs, is highly influenced by the terminology used in biology to describe the process of Darwinian evolution and natural selection. Each candidate represents a potential solution for the problem, encoded as a *chromosome*. The chromosome consists of a sequence of *genes*, where depending on the representation method can have different values. The *locus* of a gene in a chromosome refers to its position. The representation scheme used for encoding solutions to chromosomes determines the search space to be explored and also what kind of evolutionary operators are available for the purposes of exploration. Therefore the choice of the representation method can influence the effectiveness of the search (Blum & Roli 2003), and it is an important design choice.

Another cornerstone of applying a GA is being able to quantify how good of a solution each candidate is for the given problem. The objective function associated with the problem can often be used for this reason. For example, in the case of SPPs, the cost of a path describes the desirability of a given solution path. In the context of GAs, the *fitness function* is used for assigning a fitness value for each candidate.

The process of a genetic algorithm starts with initialising the population, which is usually done randomly. Then in each iteration mutation, crossover and selection operators are used to explore the search space. The selection operator is mainly responsible for intensification,

while the crossover and mutation are responsible for diversification. The crossover operator aims to combine desirable characteristics of two candidates, while the mutation operator usually introduces larger changes, to explore new areas of the search space. The original candidates before executing genetic operators are often referred to as *parents*, and the candidates resulting from the operators are called *offspring*.

The selection operator is responsible for creating the population for the successive generation from some combination of the parent and offspring candidates of the previous generation. The selection process relies on the fitness of the candidates, as determined by the fitness function. Candidates with higher fitness are usually more likely to be selected, although some candidates with lower fitness are also selected in order to preserve a high diversity of the population and avoid premature convergence. In multi-objective problems, where the fitness functions have more than one dimensions, fitness values are judged through the concept of Pareto-optimality.

One of the most popular multi-objective GAs is the NSGA-II algorithm (Deb et al. 2002), which is an updated version of the NSGA algorithm (Srinivas & Deb 1994). NSGA-II includes an elitism mechanism not included in NSGA, that makes sure that the best candidate(s) of the population are always included in the next generation. The other two distinguishing features of NSGA-II is that it uses crowded comparison as a diversity preserving mechanism, and non-dominated-sorting to emphasise non-dominated solutions. In more detail, the next population is assembled by first combining the parent and offspring populations, then the combined population is separated into non-domination classes. The first non-dominated class (rank 1) consists of the set of non-dominated candidates, the second non-dominated class consists of the non-dominated candidates among the rest of the candidates, excluding rank 1 candidates, and so on. Only half of the candidates in the combined population get to the next generation. The candidates in non-domination classes of lower ranks are included preferentially in the next population. There will generally be a class that has some of its candidates included in the next population and some of them not. The choice between the candidates in this class is based on diversity, in particular crowded comparison. Crowded comparison quantifies the uniqueness of a candidate by measuring how far away the closest solutions are in the objective space. NSGA-III (Deb & Jain 2013) is an updated version that is suitable for problems with three or more objectives, however, NSGA-II was also found to perform well on three objectives (Weise & Mostaghim

2021). NSGA-III uses a mechanism other than crowded comparison to choose which solutions are included from the splitting front. While there are numerous algorithms and methods in the field of evolutionary multi-objective optimization, such as SPEA (Zitzler & Thiele 1998), PAES (Knowles & Corne 1999), ε-MOEA (Deb et al. 2003) and ARMOGA (Sasaki & Obayashi 2005) this thesis explores initialisation methods and representation techniques for the given problem, and a thorough exploration of other multi-objective evolutionary algorithms is beyond the scope of this work.

For GAs, the representation scheme defines the search space and restricts the operations that are available in the search process, therefore it is a central design question, along with the choice of operators. In the Single objective Shortest Path Problem (SSPP) and simple graph MSPP, candidate solutions need to encode a sequence of neighbouring nodes in a graph. Several genetic representation methods have been introduced for these problems. These methods can be classified as direct representations and priority-based representations, and are reviewed in the following.

Direct variable length encoding

The most straightforward way of encoding a path in a graph is the Direct variable length representation proposed in (Munetomo et al. 1997) for the SSPP. Chromosomes consist of lists of node IDs, that form a path starting with the origin node. An arbitrary list of nodes usually won't correspond to a feasible path in the graph, and this necessitates the use of problem specific genetic operators, that maintain the feasibility of the path. Ahn and Ramakrishna introduced a crossover based on common nodes in the solution paths and a mutation based on a random walk for the SSPP (Ahn & Ramakrishna 2002), which were adapted by multiple authors for the multi-objective problem (Chitra & Subbaraj 2012, Ji et al. 2011, Li et al. 2013).

The main advantage of this representation is that it gives a one-to-one mapping, which is usually preferable over one-to-n mapping, since it avoids introducing plateaus in the search space. In one-to-n mapping, several different chromosomes might encode the same solution path, and thus have the same associated fitnesses, forming a plateau. The algorithm then might rediscover the same solutions over and over again through different chromosomes and waste computational resources.

A disadvantage is that the genetic operators might lead to repeated nodes in the path that effectively result in a loop between the these nodes, which is inherently inefficient. Therefore offspring need to be checked and repaired after mutation and crossover. Also, according to (Lin & Gen 2009, Mohemmed et al. 2008) this representation is not suitable for large networks.

Direct fixed length encoding

Another representation that assembles a chromosome from node IDs was proposed by Inagaki et al. (Inagaki et al. 1999). The length of the chromosome equals n, the number of nodes in the network, and the node IDs are the numbers from 1 to n. The chromosomes incorporate a pointer to a neighbouring node for each node. The locus of a gene corresponds to a node in the network with the same ID, and the value of the gene is the ID of a neighbour that should be the next node in the path. The path is decoded by following the pointers until the destination is reached or a loop is formed.

The crossover operator applied in (Inagaki et al. 1999) is essentially uniform crossover. This approach is deemed ineffective and requiring large population sizes in (Ahn & Ramakrishna 2002, Lin & Gen 2009). The direct fixed length representation offers one-to-n mapping of solutions to possible chromosomes, because generally there are genes whose values do not affect the decoding process, sometimes referred to as ambiguity of the representation. This can cause plateaus in the search space, where the optimisation process might get stuck.

Integer-valued priority based encoding

A priority based encoding technique that uses integer priority values to encode solution paths indirectly was proposed in (Gen et al. 1997). In this representation, chromosomes consist of some permutation of the integers from 1 to n. The priority assigned to a node with ID i is specified by the value of a gene at locus i.

A path is decoded from the chromosome by starting at the origin node and step-by-step moving to the neighbouring node with the highest priority, given it is not yet in the path. If it is already in the path, the neighbour with the next highest priority is chosen instead, if there is one.

The main advantage of this representation is that a random permutation of the priorities will always be decoded to some valid path starting from the origin node, even if it does not

reach the destination node. This means that more traditional crossover operators that do not utilise information about the network structure can be used and they can be expected to produce feasible paths, unlike in the case of the direct representations.

Position based crossover (PX) with the integer representation technique was used in (Gen et al. 1997). Later the Weight Mapping Crossover (WMX) was proposed in (Lin & Gen 2008) specifically for the shortest path problem. WMX is an extension of the one-point crossover for integer-valued priority representation (an illustration can be found in Figure 5.2). This approach has been shown to perform well in comparison to the direct variable length representation for the bi-objective problem.

The integer-valued priority representation also offers one-to-n mapping, and therefore the potential drawback of plateaus holds.

Random key based encoding

Another similar encoding technique was proposed in (Gen & Lin 2006) by using floating-point numbers instead of integers as priorities for the SSPP. These floating point priorities can be thought of as the random keys and the approach therefore is equivalent to applying the random key representation to the problem (Bean 1994). Random keys were found to be a powerful method for permutation representation in other combinatorial optimisation problems. Given that the Integer priority representation encodes a path by a permutation, employing random keys is a natural next step.

The advantage of the random keys encoding is that distinct priority values do not have to be maintained, as most of the time the value of two priorities compared won't be equal to each other. This makes it possible to use different crossover operators that are not designed specifically for the MSPP, such as arithmetical crossover, uniform crossover or two-point crossover. In (Gen & Lin 2006), arithmetical crossover is employed, where the offspring are calculated as the weighted average of the two parent chromosomes. The authors report higher search capability, enhanced rate of reaching optimal solutions and improved computation time compared to the integer-valued priority based encoding and the Direct variable length encoding. The random keys representation includes the most ambiguity and the number of possible chromosomes that encode the same solution path grows steeply with the precision of priority values.

Genetic representations for paths in multigraphs

To encode solution paths in multigraphs, the indices of parallel arcs used in the path also need to be specified.

The direct variable length representation was extended to the multi-modal transportation problem in (Abbaspour & Samadzadegan 2010) by indicating the mode of travel for each edge in the path. The chromosome is twice as long as the number of nodes in the network. The genes at odd loci correspond to node IDs, as in (Ahn & Ramakrishna 2002). The genes at even loci are used to indicate the mode of travel between the nodes.

A slightly different approach was proposed in (Yu & Lu 2012), where they only indicated the changes in the mode of travel and do not specify it for each edge separately. Genes indicating the mode of travel for the consecutive node IDs have a negative value, to differentiate them from the node IDs. However, only a limited number of parallel arcs were used. When the number of parallel arcs grows, the possible advantage of a shorter chromosome is diminished by the burden of maintaining the chromosome structure with more complicated operators.

Initialisation

Heuristic initialisation techniques have been proven to be useful in different combinatorial optimisation problems (Burke et al. 1998, Deng et al. 2015, Hasan et al. 2007, Lee & Kim 2016), as they can lead to quicker convergence by starting the evolutionary process with an already high-quality initial population. This is done by locating promising areas of the search space by utilising a priori information about the specific problem. The initial population is then drawn partially or exclusively from those promising areas.

Most mentioned works applied random initialisation, which implies a random walk starting from the origin node, random pointers to neighbours or random priorities. There are two notable examples when heuristic initialisation methods were used, both with the Direct variable length representation. (1) Information about the spatial location of the nodes in the graph is utilised (Ji et al. 2011). The initial solutions are generated in a way that each edge moves away from the origin node and closer to the destination node measured by Euclidean distance. However, 2 dimensional spatial information is not always available and in some cases might be misleading. (2) Single objective search is utilised (Li et al. 2013). Shortest paths are found according to

weighted aggregations of smaller subsets of the objectives with the Dijkstra's algorithm. These are included in the initial population along with randomly generated chromosomes.

A heuristic initialisation approach was proposed for solving the SSPP with Particle Swarm Optimisation and priority based representation in (Mohemmed et al. 2008). Similar to (Ji et al. 2011), this method aims to decrease the risk of loop formation, but it uses information about the structure of the graph instead of spatial information. Priorities were randomly assigned and node IDs were assigned in a way that the higher the IDs are, the closer a node is to the destination. Then detours in solution paths can be detected by finding decreasing node IDs. Not all detours were prohibited, only the ones where the decrease in node IDs is above some pre-specified limit.

A common concern about such initialisation methods is that they might lead to premature convergence caused by a lack of sufficient diversity. This explains why there are relatively few studies utilising heuristic initialisation for the MSPP. However, the problem of premature convergence can be mitigated by introducing enough randomness into the population, while still preserving higher quality compared to a purely random population.

Constraint handling

Constraint handling for multi-objective evolutionary algorithms is an active area of research, with most studies focused on balancing the search appropriately between the feasible and the infeasible regions, and the special case of high numbers of objectives or constraints. Penalty functions are the simplest and perhaps the most widely applied methods. They can be dynamic or static depending on if they change during the evolutionary process (Michalewicz & Schoenauer 1996). They are thought to be less suited for handling a larger number of constraints, because tuning the penalty function is difficult. However, they can be sufficient for multi-objective problems with fewer constraints (Miranda et al. 2018). For combinatorial problems, preserving feasibility and repair mechanisms are also popular choices to limit the search space. A good overview of constraint handling in multi-objective optimisation can be found in the recent tutorial (Coello 2021), and in the slightly older reviews (Mezura-Montes & Coello 2011, Coello Coello 2009).

2.5 Real world problems

2.5.1 Airport ground movement problem

The airport ground movement problem is concerned with routing and scheduling of aircraft between gates and runways and vice versa. The aim is to coordinate aircraft moving on the airport runways in an efficient and safe way. Airports are often overloaded, multiple departing and arriving aircraft are present on the taxiways at the same time, resulting in a complex and interconnected transportation system.

Efficiency of airport ground movement can be evaluated according to multiple objectives. The two most important are taxi time and fuel consumption, although other objectives such as emissions can also be taken into account (Chen, Weiszer, Locatelli, Ravizza, Atkin, Stewart & Burke 2016).

Studies concerning the ground movement problem can be separated into two main categories, the sequential approach and the global approach. In the sequential approach, aircraft are routed in the order of their starting times, where the trajectory of the already routed aircraft needs to be respected by later aircraft. The global approach on the other hand considers the order of the aircraft as a decision variable, and usually assigns routes to aircraft from a predetermined set of routes in order to keep the complexity of the problem manageable.

Earlier studies (Ravizza et al. 2014, Lesire 2010) suffered from multiple limitations, such as considering only a single objective and assumption of a constant speed for calculating traversal times. Single objective approaches can only provide a single solution, unlike multi-objective approaches that provide the available trade-offs in a single run. The realism of calculating traversal times is of key importance to provide the decision maker with accurate information and to allow good conformance during the execution stage.

A multi-objective approach, k-QPPTW (Quickest Path Problem with Time Windows) was studied in (Ravizza et al. 2013). However, a decomposition method was applied to separate the routing and scheduling aspects of the problem. Realistic speed profiles are only considered for the scheduling component, while constant speeds are assumed for the routing component. Thus only a limited number of routes are being explored for the scheduling component, which compromises solution quality compared to an integrated approach.

A trajectory-based ground movement operations framework was proposed in (Chen, Weiszer, Stewart & Shabani 2016) and (Chen, Weiszer, Locatelli, Ravizza, Atkin, Stewart & Burke 2016). This approach achieved great improvements regarding taxi time and fuel consumption compared to previous approaches (Ravizza et al. 2014, Lesire 2010). The improvements can be attributed to the more realistic speed profiles. However, the k-shortest path algorithm was used for solving the MSPP, which cannot guarantee optimality.

The NAMOA* algorithm (Mandow et al. 2005) was adapted to solving the MSPP for the ground movement problem. The introduced algorithm, AMOA* (Airort multi-objective A*) provided 5-16% improvements for the objective values on the considered test data compared to other baseline algorithms. This improvement can be attributed to the integrated way of performing routing and scheduling. However, in some cases, especially for larger airports and for a higher number of parallel arcs, the running times of NAMOA* can be unacceptable. Multigraph reduction was hence proposed (Weiszer et al. 2020) to decrease the search-space, with some compromise on solution quality. AMOA* also suffers from the problem of non-additivity.

As pointed out in (Weiszer et al. 2020), a metaheuristic solution approach can be expected to scale better to a higher number of parallel arcs. Also, the costs do not satisfy the non-additivity property (Carraway et al. 1990) (see Section 7.2.4), which is also expected to be handled better by the metaheuristic approach.

2.5.2 Other real-world applications

The MSPP on multigraphs is a relevant problem for challenges faced by a number of other real-world transportation systems. Some of these have a heavier routing component and for others, the focus is more on scheduling. The common features are the presence of multiple objectives, being able to generate efficient speed profiles for an individual vehicle and interactions of different vehicles in the same system, such that the optimal solutions for individual vehicles do not result in system level optimality. The interactions are often modelled through time constraints.

One of the first problems where the multigraph model was shown to be valuable are time-constrained vehicle routing problems. Using a multigraph model for an on demand transportation

problem reduced associated costs compared to a simple graph model (Garaix et al. 2010). A similar approach was followed by multiple authors in vehicle routing problems (Lai et al. 2016, Ticha et al. 2017), providing further evidence.

Optimising energy efficiency of urban rail transit can also be conceptualised through a multigraph, where optimising speed profiles and time tables in an integrated way provides significant energy savings (Yang et al. 2016). In urban rail transit, vehicles interact not only through inflicting time constraints on each vehicle, but through their regenerative breaking, which entails synchronization of the accelerating/braking actions. It might be the case that a speed profile that is different from the optimal when only the single vehicle is considered would be preferable in terms of the system as a whole, thus the inclusion of parallel arcs with different speed profiles could improve system level efficiency. Optimizing only for energy efficiency might degrade travel time, as lower speeds and longer coasting periods result in energy savings.

Optimal speed control of individual electric vehicles taking into account queues at intersections is studied by (Wu et al. 2015). The system level efficiency, however, is not investigated. It is pointed out that optimising for individual vehicles might compromise system level efficiency. To study the system level problem, a multigraph approach can be used, where alternative speed profiles are included for each vehicle for each leg of its route.

In marine transportation, the speed of a ship is optimised with respect to fuel price and travel time (Psaraftis & Kontovas 2013). It has also been shown, that the optimal route depends on the optimised objective (Psaraftis & Kontovas 2014). (Psaraftis & Kontovas 2014) finds that treating the speed as a variable as opposed to it being fixed is clearly beneficial even for a single objective of minimizing economic costs. This suggests that maritime transportation problems can also be modelled through a multigraph. Routing and speed decision problems for fleets of ships are a recent area of research (Wen et al. 2017), where a similar integrated routing and scheduling framework as for the airport ground movement problem might be of great use.

Multimodal transportation problems (Xiong & Wang 2014, Li et al. 2020) and ride-sharing problems (Enzi et al. 2020, Hrnčíř et al. 2015, Coltin & Veloso 2014) concern the routing of passengers or goods in a network where multiple modes of transport are available. The multigraph representation is natural to such problems. The same leg of a route could be executed by travelling via different modes or by foot. Travel time and economic cost are usually relevant

objectives. Real world datasets in this field can be large enough that processing them with enumerative algorithms takes too long even in the single objective case (Pajor 2009). This application is different in the sense that the subjects of the routing are passengers, not vehicles. Time constraints stem from the timetable of the vehicles, which can also be optimised to achieve system level optimality (e.g. balancing congestion and customer demands). Regardless of the differences in detail, the resulting problem is similar to other problems mentioned, and thereby the MSPPMTW is highly relevant for multi-modal transportation problems.

2.6 Evaluation

The fair evaluation of different solution approaches is key for being able to propose improved algorithms. There are two parts of this process worth mentioning, the benchmark problem instances used, and the assessment of the quality of the returned solutions.

2.6.1 Benchmark sets

Empirical evaluation of algorithms is an important practice in Computer Science, as it is possible that the realised computational effort is not in line with theoretical considerations (Gallo & Pallottino 1988, Müller-Hannemann & Weihe 2001). Empirical evaluation relies on standardised benchmark sets, which are an important tool for making conclusions about how the different solution approaches compare to each other, thereby facilitating the advancement of the field. Benchmark sets for shortest path problems are reviewed in Section 3.2.1, because Chapter 3 is concerned with instance generation and the identification of important features of problem instances for the Multigraph MSPP.

2.6.2 Performance measures

For exact algorithms, that return the true Pareto front if given enough time, the quality of the solutions returned is not a question. The only concern is the running time and the memory used. However, for any algorithm that is not guaranteed to return all Pareto optimal solutions the degree to which they are a good representation of the true Pareto front needs to be assessed. This includes any approximate algorithms, such as metaheuristics as well as enumerative

algorithms that do not explore all relevant parts of the search space. These algorithms return an approximation of the Pareto optimal solutions, which, in an ideal case, can coincide with the real Pareto front. There are a number of studies on the performance measures that can be used for quantifying the quality of the set of non-dominated solutions (Knowles & Corne 2002, Zitzler et al. 2003). These performance measures quantify proximity to the real front, diversity, pertinence, or some combination of these.

In this thesis, the multiplicative Epsilon indicator (Zitzler et al. 2003), the R3 indicator (Hansen & Jaszkiewicz 1994) and the Relative Hypervolume (RHV) indicator are used (Liefooghe & Derbel 2016) to evaluate the approximations of the Pareto fronts, in line with the recommendations in (Fonseca et al. 2005). Also, the study in (Liefooghe & Derbel 2016) finds that for some kind of problem instances these three performance measures are not strongly correlated, which supports the theoretical point made in (Fonseca et al. 2005), that these three performance measures evaluate solution sets from different perspectives.

Informal interpretations are provided below for the indicators, as the exact mathematical formulation of these popular metrics is out of the scope of this thesis. The interested reader can find the mathematical definitions of these three indicators in (Liefooghe & Derbel 2016). The epsilon indicator specifies the minimum factor such that all cost vectors in the true Pareto-front can be multiplied by without the approximate front dominating the modified Pareto-front. Loosely speaking, this means that $\varepsilon = 1.2$ indicates that any cost vector in the approximation set is at most 20% worse than the true Pareto front. The Hypervolume indicator is generally understood as a measure of the volume of the objective space dominated by the approximation set. The RHV interprets the Hypervolume of the approximation set in the context of the Hypervolume of the true Pareto front, that is $\frac{HV_{appr}}{HV_{Pareto}} - 1$. The R3 indicator is in the family of R-metrics (Hansen & Jaszkiewicz 1994), which are based on utility functions, such as the Chebyshev scalarizing function. The R3 indicator assesses the uniformity and diversity of solutions in the objective space. All three indicators signal higher qualities of the approximation front by lower values for minimisation problems. When the approximate front is fully converged to the real Pareto front, the R3 metric and the RHV indicators have a value of 0, and the Epsilon indicator has a value of 1.

Chapter 3

Exploring the difficulty of Multigraph Multi-objective Shortest Path Problem instances

3.1 Introduction

This chapter explores what properties of Multigraph MSPP instances affect the computational effort required for solving them. An empirical approach is taken, building on a machine learning model that learns to predict the associated computational effort from features that describe the properties of problem instances and are easy to calculate. The problem instances are selected with care, as their diversity and realism are essential in creating a predictive model that is able to generalise well. The focus of this thesis is on transportation related applications, therefore the aim is to include networks that resemble real transportation networks among others.

To establish the computational effort associated with each instance, the state-of-the-art exact algorithm, NAMOA* (Mandow & De La Cruz 2010) is used to solve the MSPP, described in Section 2.3. Computational time is used as the most straightforward measure of computational effort. Additionally, the number of Pareto-optimal solutions as an algorithm independent measure of difficulty is also considered.

The features extracted from the instances for the purposes of predicting computational effort are metrics describing the network structure or describing the costs associated with

arcs. Multiple of these features have been previously understood to affect computational effort associated with simple graph MSPP instances (Raith & Ehrgott 2009, Brumbaugh-Smith & Shier 1989). Here, it is verified that they also influence computational effort associated with Multigraph MSPP instances, the main problem investigated in this thesis. Furthermore, it is shown that additional features such as ones describing the position of the origin and destination nodes within the network also influence computational effort in both the simple and multigraph cases, and adding these features to the predictive model improves its predicting power. This model is the first of its kind to the best of the author's knowledge.

There are multiple potential applications of predicting the computational complexity of a given instance of the Multigraph MSPP. Generating difficult MSPP instances is a long standing research question (Klingman et al. 1973, Skriver & Andersen 2000, Raith & Ehrgott 2009). In real-world applications the time available for solving the problem is often limited, therefore efficiency of algorithms in solving the most difficult instances is more important than in solving easier problem instances. Understanding what makes Multigraph MSPP instances difficult could help building a scalable benchmark set, which can in turn facilitate the design of better solution algorithms. Exact algorithms might meet the given time budget in most cases, but even rare failures would render them worthless. If running times could be predicted accurately enough, the choice between an exact or a quicker approximate solution method could be made based on the prediction. The predictive model can also inform the network modelling step. Given a problem and a time budget, the maximum number of parallel arcs and objectives to be considered could be estimated such that an exact solution to the problem would be found in the given time budget.

This chapter forms the basis of proposing a benchmark set for the MSPPMTW in Chapter 4, where time windows are also incorporated. Starting with the Multigraph MSPP is appropriate, because the problem difficulty arising in MSPP instances is generally not well understood even without including time constraints.

3.2 Network structure generation

In this section, the already proposed benchmark sets for the MSPP, and the main techniques for generating them are reviewed. The possibility of generating random networks with realistic

properties is also discussed, and in particular networks that reflect properties of transportation systems.

3.2.1 Benchmark sets for shortest path problems

Designing efficient algorithms in Computer Science and Artificial Intelligence relies heavily on benchmark sets, as theoretical evaluation is often infeasible or impractical. A number of benchmark sets have been proposed based on simple graphs for multi-objective routing problems, however none became widely adopted. It is common for authors to tweak benchmark sets for their purposes such as modification of the costs (Huarng et al. 1996) or to propose entirely new test instances, such as (Chitra & Subbaraj 2012), where only one small instance was used. A review of benchmark sets used for heuristic and blind search can be found in (Machuca Sánchez et al. 2012). Here, the most important benchmark sets are described and mention some recent developments.

Testing on real networks became more popular with the availability of large datasets. However, the importance of testing on artificial datasets is still generally recognised. Their main advantage is that they can potentially provide a more robust evaluation as many problem instances can be generated and their properties can be controlled (Machuca Sánchez et al. 2012). Real instances can be expected to be less diverse and observations made using only real data might not generalise well to problems from different settings. This is especially important in the case of shortest path problems, where solving seemingly similar instances might require highly varied computational times (Ehrgott & Gandibleux 2000).

The first notable attempt at creating a benchmark problem generator for network problems was Netgen (Klingman et al. 1973) (1973). It produced random directed networks by starting with connecting all the nodes in a cycle to ensure connectivity and then adding arcs randomly until the desired number of arcs is achieved. Some of the initial arcs in the cycle were assigned high costs in order to discourage their use. Netgen has been used in its original or extended forms by multiple researchers (Huarng et al. 1996, Guerriero et al. 2001, Guerriero & Musmanno 2001). In general, Netgen produced easy to solve problems and thus does not provide a fair and balanced comparison between solution algorithms (Skriver & Andersen 2000).

The Netmaker generator (Skriver & Andersen 2000) (2000) became arguably the most

popular benchmark generator, because it was able to create instances with more Pareto-optimal solutions. This was achieved by imitating the structure of solution paths observed in Netgen graphs. Netmaker starts similarly with a cycle connecting all nodes, but then the arcs outside the cycle are added in a semi-random way. A random ordering is established within the nodes, independently from the initial cycle. Each node then gets assigned some arcs that connect it to nodes within some distance with respect to the ordering. The resulting network has a local structure, which is claimed to facilitate the wider spread of Pareto-optimal solution paths. These instances have been used by multiple authors (Raith & Ehrgott 2009, Chen et al. 2013, Ghoseiri & Nadjari 2010, Soroush 2008, de las Casas et al. 2021, Hamacher et al. 2002).

Grid generators, that produce regular lattice-like structures (Ehrgott & Gandibleux 2000, Martins et al. 2007), are also a popular choice. Without randomisation, grids provide a straightforward way to control the solution depth (Machuca et al. 2010) (the hopcount between the origin and destination), but they are also often used with some level of randomisation (Bertsekas et al. 1996). Grids were observed to provide the highest numbers of Pareto optimal solutions for the given number of nodes.

Currently the most thorough approach in the literature for evaluating MSPP solution algorithms is employing grid networks, random networks and real networks of varying sizes together. Table 3.1 provides an overview of empirical studies evaluating shortest path algorithms. There are more recent studies that rely on real world data exclusively. However, random and grid networks are also used in a number of studies, with Netgen and Netmaker often representing random networks. Some authors also used complete networks, where all nodes are adjacent to all other nodes.

Cost generation is another important component for MSPP instances. Each arc in the network needs to be assigned a cost vector specifying an objective value for each considered objective. In Netgen (Klingman et al. 1973), each cost component was independently, randomly generated. It is however widely recognised that negative correlation between the objectives leads to more difficult problems (Brumbaugh-Smith & Shier 1989). In Netmaker (Skriver & Andersen 2000), the objectives were set up to be negatively correlated by specifying two disjunct intervals and choosing the first of the objectives from either of the intervals and the second objective from the other interval. A different technique that allows for specifying the target level

of correlation between the objectives has been proposed in (Mote et al. 1991), and extended to negative correlation in (Machuca et al. 2010).

More recently, some benchmark sets have been proposed for more specialised problems. Instances of bi-objective stochastic time-dependent networks are generated in (Nielsen et al. 2003), with a grid network structure. This work received little attention from the field. A benchmark MSPP problem set has been assembled for the 9th DIMACS Implementation Challenge that included twelve real-world road maps of different sizes based on multiple US states. The problems are bi-objective, corresponding to time and distance. A many objective path finding benchmark suite was proposed covering five realistic objectives for routing applications based on real-world data in (Weise & Mostaghim 2021). This benchmark targets testing path finding on maps with constraints limited to speed limits and obstacles.

There are many benchmark instances for other related problems such as the Vehicle Routing Problem (VRP). However, VRP is fundamentally different, in that it looks to find a route for a fleet of vehicles to serve multiple customers. Shortest path problems are a sub-problem of VRPs.

3.2.2 Network generation outside of shortest path problems

The field of MSPP benchmark problem generation developed relatively independently of the general field of realistic random network generation. Netgen was created chronologically after the introduction of the classical Erdos-Renyi (ER) random graph model (1959) (Erdös & Rényi 1959) and before the appearance of network models imitating real network formation. In those times, networks in complex systems were assumed to be wired together randomly. This assumption was disproved when data about real networks became available, and network science, the study of realistic networks was born. Network science has been growing in popularity ever since, with a wide range of applications, including network security (power grids, internet), cell biology, brain research and pandemic control (Barabási 2009).

One of the main areas of interest of this field is generating networks in a random way that can better reproduce the structure of real networks. This also requires developing methods for analysing network structure for comparing networks and for quantifying their properties.

Network types (i.e. methods for generating random networks), allow drawing random

Table 3.1: Problem instance choice for Shortest Path Problems in the literature

Paper	Random networks	Grid networks	Real networks	Complete networks
Single-objective SPP				
Cherkassky et. al. (Cherkassky et al. 1996) (1996)	X	X		
Gallo & Pallotino (Gallo & Pallottino 1988)	X	X		X
Zhan & Noon (Zhan & Noon 1998) (1998)			X	
Zeng & Church (Zeng & Church 2009) (2009)			X	
Multi-objective SPP				
Brumbaugh-Smith & Shier (Brumbaugh-Smith & Shier 1989) (1989)	X			
Mote et. al. (Mote et al. 1991) (1991)	X	X		
Skriver & Andersen (Skriver & Andersen 2000) (2000)	X			
Guerriero & Musmanno (Guerriero & Musmanno 2001) (2001)	X	X		
Paixao & Santos (Paixão & Santos 2013) (2007)	X	X		X
Martins et. al. (Martins et al. 1999) (1999)	X	X		X
Raith & Ehrgott (Raith & Ehrgott 2009) (2009)	X	X	X	
Caramia et. al. (Caramia et al. 2010) (2010)	X	X		
Galand et. al. (Galand et al. 2010) (2010)	X			
Pulido et. al. (Pulido et al. 2015) (2015)		X	X	
Breugem et. al. (Breugem et al. 2017) (2017)	X	X	X	
Calvete et. al. (Calvete et al. 2017) (2017)	X	X	X	
Sedeno-Noda & Colebrook (Sedeno-Noda & Colebrook 2019) (2019)	X	X	X	
Chen et. at. (Chen et al. 2020) (2020)			X	
Pugliese et. al. (Pugliese et al. 2020) (2020)	X	X		
Weise & Mostaghim (Weise & Mostaghim 2021) (2021)		X	X	

networks with predictable properties. Flipping a coin for each potential arc as done by the ER network type is theoretically able to produce any graph with the given number of nodes. However when multiple network are generated by the same method, some structural properties are much more often obtained than others, while some properties are extremely rarely produced. This is why more sophisticated ways of generating networks are needed that can reliably produce networks with desired properties.

Multiple researchers proposed many different network models, often inspired by real world observations in network formation. The two main network models mimicking realistic network properties, scale-free (Barabási & Albert 1999) and small-world (Watts & Strogatz 1998) network models, were introduced in the late '90s, just before Netmaker was proposed, however, these and similar network models were not incorporated in benchmark sets for the MSPP.

Network models and relevance to transportation

Multiple empirical studies have shown different kinds of transportation networks to exhibit well-known structural properties from a complex network perspective. The structural properties of transportation networks depend on how the system is represented as a network. There are two methods for this representation depending on the relationship required between two nodes to be connected by an arc. The first, perhaps more intuitive method is called the space of stations, where two stations are connected if there is a direct route between them without going through any other stations. The other method is the space of transfers, where two stations are connected by an arc if there is a direct route between them, even if there is multiple stops in between. This way, each node visited in the network corresponds to a transfer. Both widely used in the topological analysis of these systems, and both suited for route planning.

Various heavy-tailed distributions have been observed in transportation networks in (Lin & Ban 2013), which correspond to the existence of hubs in the network. The scale-free property (Barabási & Albert 1999) has been observed in the case of railway networks, urban transport systems, aviation networks and maritime networks of varying sizes. A number of real networks have exponential degree distributions, which is also a heavy-tailed distribution. The most notable examples in the category are the worldwide maritime transportation network and railway networks when represented in the space of transfers. Gravity models have long

been used to capture the traffic flows between points in transportation networks, and they also produce heavy-tailed degree distributions. A recent model of network formation explains the structure of multilayer transportation networks as result from multi-objective optimisation from the part of the service providers (Santoro et al. 2018). In the model service providers are represented as separate layers, and the network is built layer by layer. Service providers are trying to create connections between pairs of nodes that are connected in less layers, while also connecting to nodes with high degrees that serve as hubs. The idea is similar to gravity models (De Benedictis & Taglioni 2011, Jung et al. 2008), where the interaction between two nodes nodes is proportional to the size of the individual nodes (where size might have an economic meaning such as exports of a country, or the degree of a node), and inversely proportional to some properties of the node pair (such as their distance or the number of connections in the different layers). The model proposed by (Santoro et al. 2018) was found to reproduce well the structure observed in real life bus, train and air plane networks.

The small-world property (Watts & Strogatz 1998) which corresponds to high clusterisation but low mean path length is generally detected in the case of aviation networks, and some other transportation networks in the space of transfers (Lin & Ban 2013). Various large Indian bus networks formulated in the space-of-stations were studied in (Chatterjee et al. 2016) and their structures ranged from exponential to scale-free, while at the same time the small-world property was also detectable. Position in space is often an important factor in the development of transportation networks. Grid-like regular structures can be observed in planned road networks. Other unplanned road networks are sometimes modelled by proximity graphs (Osaragi & Hiraga 2014), where two nodes are connected if they are close according to some geometric rules in a space. Examples of proximity graphs are Relative Neighbourhood Graphs (RNGs), Gabriel graphs and Delaunay triangulations. Random geometric graphs (Gilbert 1961) are similar, in that they connect two nodes if the distance of their coordinates in space is below some threshold. However, they are used more often as representations of wireless networks than transportation networks.

3.3 The proposed predictive model

In the following, the main components of the predictive models are described in more detail, including the generated instances, the collected metrics and difficulty measures, the feature selection process and the resulting predictive model.

3.3.1 Overview of the predictive model

A problem set of Multigraph MSPP instances is built using many different network models. Each of the instances are solved, and data is collected regarding the computational effort required, the structure of the underlying networks and the cost matrices associated with the arcs. The goal is to predict the difficulty measures given the properties of the instances. To this aim, an extreme gradient boosting (XGB) algorithm (Chen & Guestrin 2016) is used, because of its ability to handle unbalanced data and capture non-linearities and interaction between the features.

Some of the considered features were previously observed in the literature to influence the computational effort required for solving the MSPP. Others are commonly used to characterise the structure of networks. Some novel features are also proposed to describe the costs. For measuring the difficulty of the instances, the running time of a state-of-the-art algorithm (Mandow et al. 2005) and the number of Pareto-optimal solutions are considered.

3.3.2 Employed artificial network models

Artificial random networks are used for generating the network structures for the experiments, to obtain a large number of problem instances with diverse properties. These networks are generated by different probabilistic methods, and the networks created by the same method generally have similar properties, and it is said that they belong to the same *network type*.

Multiple network types are employed in the experiments, including those similar to the previous benchmark sets for MSPP, imitating properties observed in real transportation systems, and some additional network types to diversify the problem set in order to achieve higher generalisability.

This section details the network generation methods. In all cases, the inputs for generating a network are: (1) the network type, (2) the number of nodes, (3) the arcs to nodes ratio and

(4) a random seed. Note that the arcs to nodes ratio is equivalent to specifying the number of arcs. Controlling the number of nodes and arcs in the generation process should make it easier to identify additional important features. Cost generation and the inputs used for that purpose are discussed in the Section 3.3.3.

A number of established network generation methods were employed, in some cases additional modifications were applied to control the number of arcs, form a connected network or adapt the network model for generating undirected networks. All employed network models provide direct control over the number of nodes but not over the number of arcs, although, often, the number of arcs can be set approximately. For Grids and Proximity graphs, the number of arcs is normally not controllable. For these network types, ways of achieving the desired ratio of arcs to nodes were devised in a network type specific way. Most of the time, the number of arcs needs further adjusting after the initial network is generated, this is done by randomly removing or adding arcs, as necessary. In some cases, the number of arcs is too low to create a connected network of the given network type. In these cases, connectedness of the initial network is ensured, by allowing a higher ratio of arcs to nodes when it is needed, then removing extra arcs randomly, while avoiding disconnecting the network. In the following the considered network types are discussed one by one, including any modifications from the original network type and the parameter settings chosen to get an initial network with the number of arcs close to m.

The Erdos-Renyi (ER) network type (Erdös & Rényi 1959) has one parameter, the probability p_{ER} of each pair of nodes being connected by an arc. This parameter gives indirect control over the number of arcs. Specifying $p_{ER} = \frac{2m}{n(n-1)}$ results in approximately m arcs. With low values of p_{ER}, it is possible that the resulting network is not connected. In this case, the value of p_{ER} is increased by 0.02, until a connected network is generated, and then the arcs are adjusted randomly if needed.

The Barabasi-Albert (BA) network type (Barabási & Albert 1999) grows a scale-free network by preferential attachment to high degree nodes. Nodes are added step by step and each node is connected to p_{BA} already existing nodes, chosen based on their degree. The parameter value is specified as $p_{BA} = \lfloor \frac{m}{n} \rfloor$.

The Holme-Kim (HK) network type (Holme & Kim 2002) creates networks with powerlaw

degree distribution with a high clustering coefficient, as opposed to the BA, which creates no clustering in the network structure. Clustering, corresponding to highly connected small groups of nodes in the graph is considered to be more realistic. The network type has two parameters, p_{HK1} which has the same role as p_{BA} and p_{HK2} which is the probability that instead of a single arc, two arcs are added in a way to form a triangle with an existing arc in the network. The parameter p_{HK2} is set up randomly below 0.75, and $p_{HK1} = \lfloor \frac{m}{n} \rfloor$.

The Newman-Watts-Strogratz (NWS) network type (Newman & Watts 1999) first creates a ring where each node is connected to its p_{NWS1} neighbours. Then for each existing arc (u,v) a new random arc (u,w) is added with probability p_{NWS2}, where w is an existing node. The parameter value are specified as $p_{NWS1} = 2 * \lfloor \frac{m}{n} \rfloor$ and p_{NWS2} is chosen randomly below 0.75.

The random regular graph (RR) network type generates networks such that all nodes have the same degree, controlled by the parameter p_{RR}. In this regard they are similar to some grid networks, but have a less regular structure. The parameter value is specified as $p_{RR} = \lfloor 2 * \frac{m}{n} \rfloor$.

A model is employed that is based on a random tree graph (Tree) (a graph without any cycles). These graphs are extremely sparse, and therefore the random adjustment of arcs usually has a large effect on the network structure. It was found that graphs generated this way are sufficiently different from other network types. Also, this network type can be thought of as an adaptation of Netgen (Klingman et al. 1973), where the connectedness is ensured by starting the network generation with a tree instead of a cycle. The Tree network type does not require any parameters

For generating exponential networks (Exp), the network model proposed in (Deng et al. 2011) is adapted. This network type is grown by the process of adjacent random attachment. The algorithm for generating instances of this network model is presented in Algorithm 2. A small ER network is grown by one node (v_2 in line 5) and K arcs incident to it in each step, until it has the desired size. The K neighbours of the newly added node v_2 are chosen in line 7, such that they are the set of K nodes at the smallest hopcount from node v_1, which is chosen randomly in line 4. This way, all the neighbours of the newly added nodes are clustered in the graph.

A recent model explaining the formation of multilayer transportation networks (Santoro et al. 2018), that is based on a modified gravity model (De Benedictis & Taglioni 2011, Jung

Algorithm 2 Exponential network model (modified version of an existing method in (Deng et al. 2011))

Input: The number of nodes (n), the number of arcs (m)
Output: An exponential network (G)

1: $G \leftarrow$ a random ER graph with 10 nodes and 6 arcs
2: $K \leftarrow \frac{m-6}{n-10}$
3: **while** number of nodes in G is less than n **do**
4: $v_1 \leftarrow$ randomly chosen node in G
5: $v_2 \leftarrow$ new node not yet in G
6: Add arc (v_1, v_2) to G
7: $W \leftarrow$ the K nodes at the closest hopcount from node v_1
8: Add all arcs to G of the form (v_1, w), where $w \in W$
9: **end while**
10: Return G

et al. 2008) (Grav) was also employed as a network type. The network is grown layer by layer, and each of the layers corresponds to a service provider. Nodes with higher overall degrees attract arcs, while higher number of arcs between the same two nodes in other layers discourages an arc. The resulting networks are collapsed to single layer ones in order to include them in the experiments. This is equivalent to ignoring the different service providers and only keeping the underlying network structure. Converting to simple graph is necessary, because the experiments include instances with homogeneous number of parallel arcs. Still, the network type reflects the properties of networks underlying multi-layer transportation networks, and therefore it is worthwhile to include. The algorithm for generating instances of the Grav network model is presented in Algorithm 3. Individual layers are built according to the probabilities specified by the gravity model in lines 4-19. If there are any isolated nodes, each new layer starts with incorporating an isolated node (lines 5-8). Then the probabilities are calculated for each potential arc to be added (lines 11-16). When a given size is reached by the layers, specified as $n/10$, they are finished, added to the multi-layer network (line 20) and a new layer is started.

A grid based network type is also employed. Initially, a square grid is generated with the lowest square number of nodes s, such that s is above n. Then the extra $s-n$ nodes are removed from the last row and column, starting from the corner, moving on to its two neighbours, and so on. In square grid networks (Grid) most nodes have a degree of four, and thus the number of arcs is usually not controllable. To achieve higher arcs to nodes ratio, while limiting the introduced randomness to the regular structure of a grid, each node is joined to the 8 closest nodes, instead of only the 4 closest. This more dense grid is the starting point when the desired

Algorithm 3 Gravity model for multilayer network growth (modified version of an existing method in(Santoro et al. 2018))

Input: The number of nodes (n), the number of arcs (m)
Output: G_s: a single layer network

$G_{base} \leftarrow$ a random tree graph on n nodes
$G \leftarrow$ a multi-layer network, with G_{base} as the first layer
while The number of arcs with distinct node pairs in $G < m$ or G is not connected **do**
  $H \leftarrow$ empty graph for the new layer
  if there are isolated nodes in G **then**
    Add a random arc with an isolated node as its end node to H
  else
    Add a random arc to H
  end if
  while The number of arcs in $H < n/10$ or the number of arcs with distinct node pairs in $G < m$ **do**
    for all $(v_i, v_j) \in$ distinct pairs of nodes in G_{base} **do**
      if v_i or v_j is in H and not both are in H **then**
        $prob(v_i, v_j) \propto \frac{degree(v_i) * degree(v_j) + 1}{arcs(v_i, v_j) + 1}$ ($arcs(v_i, v_j)$ is the number of layers where v_i and v_j are adjacent
      else
        $prob(v_i, v_j) \propto 0$
      end if
    end for
    Randomly choose one node pair according to *prob* and add the corresponding arc to H
  end while
  Add H to G as a new layer
end while
$G_s \leftarrow$ convert G to a simple graph
Return G_s

number of arcs is at least 1.7 times the number of nodes. For even denser networks, above the arcs to nodes ratio of 3, in the initial network each node is being connected to the nodes closer than $\sqrt{2}$ units, where one unit is the minimum distance between nodes in the grid.

Random geometric graphs (Penrose et al. 2003) (GG) are generated by placing n nodes randomly in the unit square in the Euclidean plane. Then pairs of nodes are connected by an arc, if the distance of their coordinates is at most p_{GG}. The value of the parameter is set up as $p_{GG} = 0.01$, then the value is incrementally increased by 0.01 until the desired number of nodes is reached, and the network is connected.

Two models are employed in the process of generating proximity graphs (Prox). The Relative Neighbourhood Graph (RNG) (Toussaint 1980) is an undirected graph defined on a set of nodes with coordinates in the Euclidean plane. Each pair of nodes v_i and v_j are adjacent if a third node v_k does not exist that is closer to both v_i and v_j than they are to each other. The RNG provides networks with a low arcs to nodes ratio . If the desired number of arcs is not reached

with the RNG, a different proximity graph model, the Delaunay triangulation is used to create a network on the same nodes with the same coordinates that produces different arcs. Then the superposition of the arcs from the two networks creates the considered network. A triangulation draws connections in between a set of nodes in the Euclidean space, such that the convex hull of the set of nodes is divided into triangles. The Delaunay triangulation maximizes the minimum angle of the triangles in a triangulation. Then, if even higher density needs to be achieved, arcs of Delaunay triangulations of random subsets of the nodes are further superimposed on the network repeatedly until the desired number of arcs is reached.

Netmaker (Skriver & Andersen 2000) originally produces directed networks. The first network type included based on Netmaker is the undirected version of original Netmaker networks (NM1). Netmaker starts out with connecting all nodes in a cycle to ensure connectivity. Since undirected networks do not need a cycle for this, a different version is also included (NM2), where a path is used instead of a cycle to ensure connectivity.

After the network structure is generated, the two special nodes need to be selected. The origin (v_O) and destination (v_D) nodes were chosen in a way to cover potential interesting cases. It might be easier to find optimal routes to a central node (C) of the network from a peripheral node (P), than in the opposite direction (from C to P), because generally there are more directions to search in starting form a central node. Also, higher hopcounts between the v_O, v_D nodes are expected to lead to an instance with higher computational demand. Thus in the first quarter of the instances v_O and v_D are chosen to be two end points of a diameter of the network. In the second quarter of the instances the origin is a central node and the destination is a node from the periphery. In the third quarter, the origin is a node from periphery and the destination node is central. In the last quarter v_D and v_O is chosen randomly.

3.3.3 Cost generation

A problem instance is complete when a cost vector is assigned to each of the parallel arcs of the network. Two established cost assignment methods are adapted to the multigraph case.

The first approach from Mote et. al. (Mote et al. 1991) produces bi-objective costs with a positive correlation. The first cost component (C_1) is randomly generated from a uniform distribution within a certain range, here $C_{min} = 10$, $C_{max} = 1000$. The second cost component

(C_2) is a convex combination of the first cost component and a randomly generated value from the same distribution($(C_2)^*$). Equation 3.1 shows the formula, where ρ is a correlation multiplier.

$$C_2 = \rho * C_1 + (1-\rho) * (C_2)^* \tag{3.1}$$

This approach has been extended to negative correlation (Machuca et al. 2010). Equation 3.2 shows the formula, where in case $\rho < 0$.

$$C_2 = C_{max} + C_{min} - (|\rho| * C_1 + (1-|\rho|) * (C_2)^*) \tag{3.2}$$

The generation of a cost-matrix starts with the first objective values for all parallel arcs, which is the first column. The first cost component in the first column is chosen randomly in the allowed interval and the second and third component of the first column is calculated with the above formula with a random correlation multiplier $\rho_{column} \in [-1,1]$. This way, it can be controlled how similar the first objective values are for the multi-arc. The second objective value in each row is calculated from the first objective value of that row with another random correlation multiplier $\rho_{row1} \in [-1,1]$. If it exists, the third objective value is calculated from the first objective value similarly, but with a different random correlation multiplier $\rho_{row2} \in [-1,1]$. If a row is dominated by another row the dominated one is regenerated with the same method until all the rows are non-dominated.

The cost matrices generated this way undergo a further transformation, to control the range of the objective values. This is achieved by replacing the objective values (C_1,C_2) of any arc, by $(\frac{C_1}{f_s} + \frac{C_{max}+C_{min}}{2} * (1 - \frac{1}{f_s}))$, where f_s is the *shrinking factor*. This way, the range of objective values is decreased while keeping correlation of objectives the same. Then it becomes possible to differentiate between the effect of the two features, the correlation of objectives and the high range of objective values. The effect of the shrinking factor is illustrated in Figure 3.1.

The values ρ_{column}, ρ_{row1}, ρ_{row2} and f_s are the cost specific parameters for the instance generation model of the current chapter.

3.3.4 Overview of the generated set of instances

In summary, the parameters of our generator with their scopes are summarised in the following:

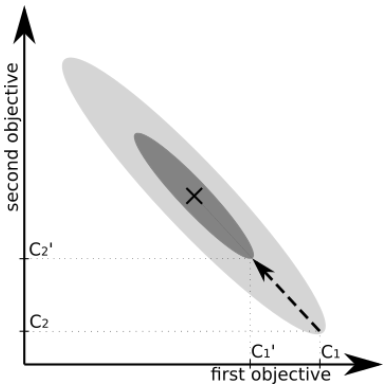


Figure 3.1: Illustration of the role of f_s the shrinking factor.

- The network types used for structure generation: Netmaker (NM1, NM2) (Skriver & Andersen 2000), Erdos-Renyi (ER) (Erdös & Rényi 1959), Barabasi-Albert (Barabási & Albert 1999) (BA), Holme-Kim (Holme & Kim 2002) (HK), Newman-Watts-Strogatz (NWS) (Newman & Watts 1999), Geometric graphs (GG), Random regular graphs (Kim & Vu 2003) (RR), Randomised trees (Tr), Proximity graphs (Prox), Exponential networks (Exp) (Deng et al. 2011), Gravity models (Santoro et al. 2018) and Grids. With the modifications described earlier in the section.
- The number of nodes: $\in (50, 500)$.
- The ratio of the number of arcs to the number of nodes: $\in (1.1, 5)$.
- The position of the origin and destination nodes within the network: random, from periphery to center, from center to periphery, from periphery to periphery where the two nodes define a diameter of the network.
- The approximate correlation of the first and second and the first and third objective values: $\in (-1, 1)$.
- The number of objectives: $\in (2, 3)$.
- The number of parallel arcs : $\in (1, 5)$, which is homogeneous in the network.
- The shrinking factor for cost generation: 1, 2, 5 or 10. This results in the cost component ranges: $(10, 1000), (250, 750), (400, 600)$ or $(450, 550)$.

These parameters define the set of instances that will be used to learn what characteristics make instances more difficult to solve. The values have been set to have as much diversity as possible while keeping the instances at a size where most of them can be solved in a reasonable

time. Diversity in characteristics suspected to influence the difficulty is particularly important. Specifically, the computational time consideration limits the size of the network structure, the number of objectives and parallel arcs. The settings for the shrinking factor are aimed at collecting data at different levels of the parameter, other settings could have been used, as long as they give a good variation in terms of cost component ranges.

The generated problem set comprises 130000 instances. However, some instances could not be evaluated within 5 days allowed for the experiments. After removing those instances, our final dataset contains 119075 data points.

3.3.5 Features considered

The high variance in the measured running times of solution algorithms is not well understood even for simple graphs, after decades of research. Here, a set of metrics are collected that characterise the Multigraph MSPP problem instances, so that it can be found out how the properties of the instances affect the computational effort associated with the instances. Some of the metrics are adopted directly from the literature of MSPP instances, as the multigraph case has not yet been studied. Other metrics are inspired by the fields of Graph Theory and Network Science.

Properties affecting instance difficulty

Multiple studies suggest that some properties of the MSPP instances have an effect on the computational effort required to solve the given instance. In the following the related observations are summarised.

The size of the network as measured by the number of nodes or the number of arcs have been observed to affect problem difficulty in a number of studies. Most notably Brumbaugh and Shier (Brumbaugh-Smith & Shier 1989), who found that computational time grows linearly with the number of nodes. They made this observation by analysing the average runtime of multiple instances with the same structure and correlation of objectives. This finding have been confirmed in (Cherkassky et al. 1996) and (Maristany de las Casas et al. 2021). However, Raith and Ehrgott (Raith & Ehrgott 2009) found that contrary to the general assumption, increasing the network size did not influence the running times in an obvious way.

Guerriero and Musmanno (Guerriero & Musmanno 2001) investigated solution approaches for the single origin all destination variant of MSPP . They observed that a higher number of Pareto optimal paths are usually paired with high running times, and identified the ratio of arcs to nodes as a structural property influencing running times.

It is clear from the results of Guerriero and Musmanno (Guerriero & Musmanno 2001) that increasing the number of objectives can drastically increase running times. This has been also observed in the experiments of Maristany et. al. (Maristany de las Casas et al. 2021).

Brumbaugh and Shier (Brumbaugh-Smith & Shier 1989) also determined that larger interval for costs and negatively correlated objectives result in more solutions. Negatively correlated objectives are widely understood to lead to the most difficult problems (Skriver & Andersen 2000, Machuca et al. 2010). High positive correlations lead to a problem akin to a single objective problem.

Regarding the size of the interval for the cost components, there is less support for the above observation. A similar, but fundamentally different characteristic was the center of the study of Muller-Hanneman and Weihe (Müller-Hannemann & Weihe 2001). They argued that most realistic MSPPs have a small number of Pareto-optimal solutions, and thus can be solved in reasonable time by conventional algorithms. In fact, with the two assumptions (1) the number of distinct values the ratio of the two components of the cost vectors might take is limited in the network and (2) such ratios along solution paths are *monotonous* or can be separated into two monotonous parts, it was proven that the number of efficient solution paths grows at most polynomially with the size of the network. These assumptions were observed in a real train network investigated by the authors. The ratios are called monotonous along a solution path, if their values monotonously increases or decreases when following the arcs of the whole path.

Raith and Ehrgott (Raith & Ehrgott 2009) identified the hopcount between origin and destination nodes as a potentially important factor. Also, how “branched” the route is between these nodes was pointed out as another factor that might be important. The hopcount between origin and destination can also be called the solution depth and was shown to be important in other studies (Machuca Sánchez et al. 2012). Here “branchedness” of the route between the origin and destination node is interpreted as the average degree of the nodes included in the path with the smallest hopcount (*hop-path*) between the two nodes, as there aren’t any established

metrics with this aim.

In light of the above, the list of the features that were already proposed in the literature includes the number of nodes, the number of arcs, the correlation of objective values, the hopcount between the origin and the destination, the average degree of hop-path and the interval size for cost component generation.

Other metrics

A single origin single destination Multigraph MSPP instance on a static network is defined by the network structure and the costs. The structure corresponds to a network with two special nodes, the origin (O) and destination (D) nodes. Only undirected networks are considered (or equivalently directed networks such that all the arcs are bidirectional), and the costs are equal in both directions. The costs assigned to each arc in the network are represented as vectors in the simple graph case and as matrices in the multigraph case. The simple graph case is included in the multigraph case, when the first dimension of the matrix is one. The dimensions of the matrices are u by q, where q denotes the number of objectives and u denotes the number of parallel arcs. The cost matrix assigned to arc $(v_i, v_j$ can be interpreted as follows. The rows correspond to the u different ways the route between node v_i and node v_j can be traversed, and the columns correspond to the q objective values characterising the given way of traversal.

A multigraph MSPP instance has various properties which might potentially affect the imposed difficulty. A set of metrics not conventionally considered in MSPP problem sets are calculated. These can be categorised as global metrics that describe the structure of networks as a whole, such as the diameter, and local metrics that describe the position of nodes in the network, such as centralities. The latter are often used to characterise the position of a given node in a given network from different aspects. There have been numerous graph metrics introduced for describing and analysing the structure of graphs. There are groups among these metrics that are describing similar properties of the structure. The presence of highly correlated features, and the high number of initially considered features makes feature selection an important part of our study.

All the collected metrics are presented in Table 3.2, most of which are well known in the Network Science and Graph Theory literature (Hernández & Van Mieghem 2011), and

therefore only two recent, lesser known sets of metrics are mentioned here. Note that most of these metrics will not be among the selected features. Gragnostics (Gove 2019) is a set of ten easily computable and understandable metric for comparing graphs. The DDQC method (Aliakbary et al. 2015) defines a set of values that characterise degree distributions of networks for network analysis. Both of the above sets of metrics have been shown to be highly effective at distinguishing between different network types.

Some novel metrics are proposed for describing the costs, based on the magnitude and direction of the cost vectors. Correlation of objective values is generally recognised as an important feature of problem difficulty. However, it does not cover all important aspect of the costs. The range of objective values in the network might also be important, as pointed out in (Brumbaugh-Smith & Shier 1989). However, when all cost components are multiplied by the same number, the range changes and the difficulty of the instance does not change. This is an undesirable characteristic for a feature. Alternatively one might consider the number of different values the ratio of two different objectives for the same arc takes, as proposed in (Müller-Hannemann & Weihe 2001). This metric does not change with the scaling of costs, however, it ignores the magnitude of the cost vector components. The direction of the cost vector expressed as an angle can be used equivalently with the ratio of the cost components, while the length of the cost vector characterises the magnitude of the costs. Therefore, the direction and length of the cost vectors are considered.

The range of the directions can be calculated as ($\alpha = tan^{-1}(C_1/C_2)$) of the cost vectors (C_1, C_2) in the network. Similarly, the range of the lengths ($l = \sqrt{C_1^2 + C_2^2}$) of the cost vectors are also calculated. In the case of three objectives, these values are also calculated for the vector (C_1, C_3).

To characterise how different the cost vectors are on the parallel arcs between the same nodes in the instance, the direction ranges are also calculated for arcs and their mean is included as a metric. In the case of three objectives, the range is considered over each of the three pairs of the objectives. The resulting feature is named *mean cost direction range on arcs*.

Table 3.2: Graph metrics collected in experiments.

Model parameters	• number of objectives • number of parallel arcs
Global structural metrics	• arc number • node number • arcs to nodes ratio • degree diversity (Li et al. 2011) • clustering coefficient (Hernández & Van Mieghem 2011) • diameter (Hernández & Van Mieghem 2011) • radius (Hernández & Van Mieghem 2011) • characteristic path length (Lin & Ban 2013) • the ten gragnostics (Gove 2019) • DDQC metrics (Aliakbary et al. 2015) • assortativity (Hernández & Van Mieghem 2011) • small-worldness (omega) (Telesford et al. 2011) • sigma (Humphries & Gurney 2008) • global efficiency (Latora & Marchiori 2001) • local efficiency (Latora & Marchiori 2001) • size of cycle basis (Kavitha et al. 2008)
Local structural metrics	• degree • average neighbour degree (Kogotkova et al. 2018) • closeness centrality (Lin & Ban 2013) • betweenness centrality (Lin & Ban 2013) • eigenvector centrality (Bonacich 1987) • harmonic centrality (Boldi & Vigna 2014) • eccentricity (Hernández & Van Mieghem 2011) • closeness vitality (Kogotkova et al. 2018) • coreness (Hernández & Van Mieghem 2011) • load centrality (Kogotkova et al. 2018) • square clustering (Zhang et al. 2008) • efficiency (Latora & Marchiori 2001) • hopcount (Hernández & Van Mieghem 2011) • average degree of hop-path • node connectivity (Hernández & Van Mieghem 2011) • arc connectivity (Hernández & Van Mieghem 2011)
Cost metrics	• correlation of values of first and second objectives, correlation of values of first and third objectives, • cost direction range, • cost magnitude range, • mean cost direction range on arcs

3.3.6 Measuring difficulty of instances

To evaluate the difficulty of our problem set, the state-of-the-art exact MSPP solution algorithm, NAMOA* (Mandow et al. 2005) is employed, with the extension to the multigraph case. The extension requires iterating through parallel arcs in expansion. Heuristic functions are not employed here, because their use and the choice of a particular heuristic function might affect the computational effort. Problem instances for which the running time exceeded 5 days for

solving the Multigraph MSPP are not included in the results.

The running times and the number of domination tests only characterize the execution of the NAMOA* algorithm, which outputs the set of non-dominated cost vectors, and not the solution paths themselves. The solution paths can be found using a backtracking algorithm. Backtracking can also be time consuming if all solution paths are found. However, this is usually not needed, as the decision maker can make the choice between the available solutions based on the cost vectors, and then only one solution path needs to be found by the backtracking algorithm. Therefore, this additional computational burden is negligible.

The most obvious measure of difficulty is the running time a given instance requires when it is solved by an exact algorithm. Its main shortcoming is that it is platform dependent. The number of domination tests performed by the algorithm can be used as a more consistent proxy for the running time. Domination tests take up the most time when executing the NAMOA* algorithm, and the Spearman correlation coefficient of the running times and the number of domination tests is above 0.99. The number of domination tests is platform independent, and hence it will be exactly the same for different runs of the algorithm on the same instance.

The number of Pareto-optimal solutions is an important aspect of problem difficulty often considered in the literature (Machuca Sánchez et al. 2012). It is expected that more Pareto-optimal solutions generally mean higher running times. However, fewer Pareto-optimal solutions might not always mean low difficulty. It is possible that up to some point a high number of Pareto-optimal solutions were held in memory, but later many of them were eliminated. Nevertheless, the number of Pareto-optimal solutions provides a difficulty measure that is independent of the algorithm used (as long as the algorithm is exact), and therefore only depends on the problem instance. However, the correlation of the number of Pareto-optimal solutions and the running times is not as high (0.841) as for the number of domination tests. Both the number of domination tests and the number of Pareto-optimal solutions are considered as difficulty measures.

The logarithmic transformation of the difficulty measures is used for the prediction of difficulty. The data points become increasingly sparse at higher running times, for example a fraction of the experiments have running times higher than 5 days, while around half of the experiments require less then 5 seconds. At the same time the accuracy is less important at

higher values. By predicting the logarithm, the focus is put on the magnitude of the running times. The logarithmic transformation of the number of domination tests is denoted as log_dom, and the logarithmic transformation of the number of solutions found is denoted log_sol.

3.4 Model building and results

The methods are implemented in Python 3. All numerical tests are performed on Queen Mary's Apocrita HPC facility (Z n.d.*b*). Parallelisation has not been utilised.

3.4.1 Domination checks performed as the difficulty measure

In the first part, the logarithmic transformation of the number of domination tests is considered as the measure of difficulty. Starting with all the metrics listed in Table 3.2, the best features are chosen in the feature selection process to use in our model. Three main questions: (*i*) is our model able to generalise and predict difficulty well for unseen network types, (*ii*) is the predictive power of our model better than a model using only the features that were noted in the literature to affect difficulty, (*iii*) which features are the most important.

Feature selection

Feature selection is needed, to eliminate features that do not carry important information, and also because including pairs of features that are highly correlated skews feature importance scores. Discarding unnecessary features is an essential step towards an interpretable model and to improve accuracy.

The feature selection process was composed of three steps. First, the most highly correlated features are removed, such that there are not any two features in the feature set with a correlation value higher than 0.8. Secondly, an initial XGB model is fitted to the training data. The feature importances are extracted from the model, and the 20 most important features are kept according to the default importance measure provided by the XGB model, which is based on the gains in the impurity, when the given feature is used for a split in the decision trees. In the third step, recursive feature elimination was performed, where the least important feature is dropped if this results in a better performance on the validation set. Afterwards the model is refitted. The third

step is repeated until each remaining feature contributes positively to the performance on the validation set as measured by the R-squared score.

At the end of the feature selection process, twelve selected features remain: the number of parallel arcs, the hopcount between the origin and destination nodes, the number of arcs, the number of nodes in the network, correlation between the first and second objectives, correlation of first and third objectives (includes information about number of objectives), mean cost directions range, range of cost directions for the first and second objectives, square clustering of the origin node, harmonic centrality of the origin node, closeness of the origin node, and degree diversity.

There are four features describing the costs among the selected features. Correlation of objectives is showing to be a useful feature even if the direction of cost vectors is also considered. The direction is also important, both the average over the full network and the range covered. The length of the cost vectors was not selected, which seems to support the claims that the ratio of the objective values is the most important (Müller-Hannemann & Weihe 2001). There are not any features selected that describes the position of the destination node in the network, but there are three that describe the position of the origin node. It can be expected that the origin node is more important, given the search usually explores the local environment of the origin node, where the search starts, while not necessarily exploring the environment of the destination node.

It is important to note that the number of objectives is not being included as a feature directly, because it is it is included indirectly. For example, when the correlation of the first and third objectives is not interpretable, that is, when there are only two objectives, it is a missing value. Then the model has access to the information of which instance has two or three objectives based on such missing values.

Generalisation to unseen network types

In this section, the generalisation potential of the predictive model is examined. This is done by predicting difficulty of networks that were generated by network models not included in the training data. Ideally, the model should be trained on instances similar to the target instance. However is not always practical. Similar networks are not always available in high numbers and even if they are, retraining the model takes time. Artificial networks however are readily

available and thus large and diverse training sets can be assembled. Our hypothesis is that a mix of different structures in the training set provides some level of generalisability to unseen network types.

To test this hypothesis, the following experiment is conducted. One by one, each of the network types was treated as an unknown network type, meaning that it is assumed that there are not any instances available for training purposes from the unknown type. Therefore it is reserved only for testing. Multiple XGB models are fitted using the data corresponding to each of the network types one by one for training, using the selected features. Additionally, the set Mix1 includes the same number of data points as the single type training sets (6000) drawn randomly from all data points that belong to a different network type, than the testing type. Mix2 includes a sample of twice that many data points and Mix3 includes all appropriate data points. The mixed sets are only used for training.

The results are shown in Table 3.3. The scores are an average of 3 runs in each case, and all models have 200 estimators and a learning rate of 0.05. The training set is either comprised of instances with a single network type, or mixed network types that does not contain the testing type. In each column, corresponding to a testing network type, two values are highlighted in bold. The best scores reached with a single training type and the overall best score in the column. For a given test set, some training sets show significantly better results than others. This suggests that using only one network type for training is risky if the similarity to the test network is not verified.

When comparing best R-squared value for each column among the ones trained on a single network type to the ones achieved by mixed training sets in Table 3.3, it can be seen that even Mix1 is very close to the best score achieved by a single type training set in most columns. The only exception is the first column, when the testing network type is NM2. In this case Mix1 performs worse than the Grid training set. However Mix1 outperforms most of the single type training sets even in the first column. When the number of data points in the mixed training set is increased, the predicting power is increased. If the size of the mixed training set is not limited (Mix3), the predicting power is better in all cases than the any of the single type training sets. In conclusion, these results show that using a mixed training set to make predictions about unknown instances is beneficial for two reasons. Firstly, because of the decreased risk of training

Table 3.3: R-squared values for XGB models on test data, where the testing set contains network types not contained in the training set.

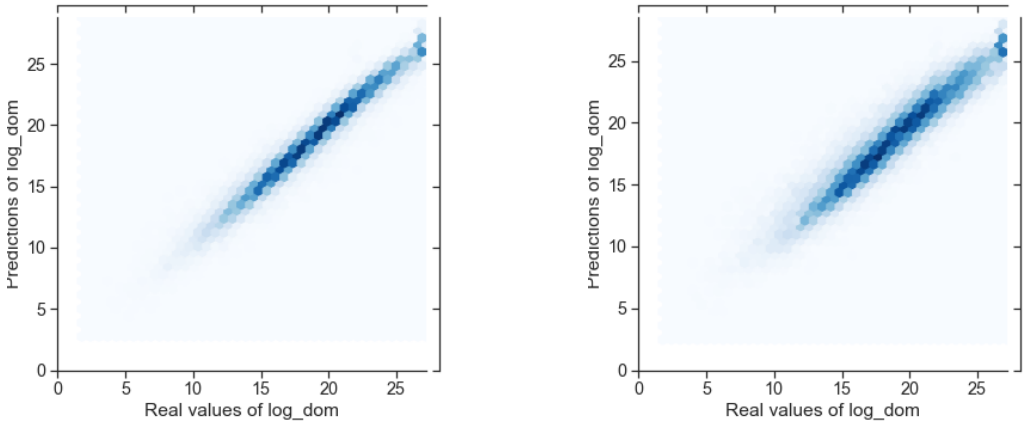
	Testing network type												
Train n. t.	NM2	ER	Exp	GG	HK	RR	Tree	Grid	Grav	NM1	NWS	Prox	BA
NM2	-	0.55	0.54	0.84	0.27	0.58	0.56	0.82	0.49	0.69	0.69	0.84	0.27
ER	0.40	-	0.86	0.76	0.86	0.89	0.90	0.79	0.83	0.89	0.89	0.84	0.87
Exp	0.51	0.87	-	0.77	0.88	0.85	0.88	0.82	**0.86**	0.87	0.86	0.84	0.87
GG	0.76	0.69	0.69	-	0.51	0.72	0.68	0.88	0.61	0.75	0.79	**0.91**	0.48
HK	0.37	0.86	0.86	0.71	-	0.85	0.87	0.77	0.84	0.85	0.85	0.80	**0.89**
RR	0.34	0.89	0.84	0.76	0.84	-	0.90	0.79	0.81	0.91	0.89	0.86	0.85
Tree	0.40	0.89	0.86	0.77	0.85	0.90	-	0.80	0.83	0.90	0.88	0.84	0.86
Grid	**0.78**	0.83	0.82	0.88	0.74	0.84	0.84	-	0.77	0.87	0.87	0.90	0.75
Grav	0.38	0.83	0.85	0.67	0.84	0.78	0.84	0.75	-	0.79	0.81	0.75	0.85
NM1	0.40	0.88	0.84	0.77	0.80	0.90	0.90	0.80	0.80	-	**0.89**	0.86	0.80
NWS	0.55	**0.89**	**0.87**	0.83	0.83	**0.91**	**0.90**	0.85	0.83	**0.91**	-	0.88	0.83
Prox	0.65	0.78	0.79	**0.88**	0.66	0.81	0.81	**0.89**	0.72	0.85	0.85	-	0.66
BA	0.36	0.86	0.86	0.70	**0.88**	0.85	0.87	0.77	0.84	0.85	0.84	0.79	-
Mix1	0.68	0.89	0.88	0.85	0.87	0.89	0.90	0.87	0.84	0.90	0.89	0.88	0.88
Mix2	0.72	0.90	0.89	0.87	0.90	0.90	0.91	0.89	0.86	0.91	0.90	0.90	0.89
Mix3	**0.80**	**0.93**	**0.92**	**0.91**	**0.92**	**0.94**	**0.94**	**0.93**	**0.89**	**0.94**	**0.94**	**0.94**	**0.92**

on an incompatible network type, even if only a fraction of the training set is from a network type that is similar to the unknown network, the R-squared value is not compromised compared to using only the similar network type as training data. Secondly, it is easy to collect large quantities of diverse training data using multiple random network generators.

In the following a training set including all the network types is used to build a predictive model. Based on the above results, it is reasonable to expect it to generalise well to unseen network structures.

The predictive model

The data set of 119075 data points (see Section 3.3.4) is separated into three parts, the test set is 20% off all data points, 20% of the remaining data points is in the validation set used for parameter tuning, and the remaining data points are in the training set for the final model. The XGB model with the selected features is tuned using the validation data by grid search on the two parameters including the number of estimators and learning rate. A different XGB model with different features, those already mentioned in the literature is also built and also tuned in the same way. The fit of the two models on the test set is compared in Figure 3.2. The R^2 score of the model with literature features is 0.891, while this value is 0.955 for the model with the



(a) Selected features, $R^2 = 0.955$ **(b)** Literature features, $R^2 = 0.891$

Figure 3.2: Goodness of fit shown in hexagon plots on unseen test data for the prediction of the logarithmic transformation of the number of domination tests by the model with the selected features (3.2a) and the model with the literature features (3.2b) .

selected features, which clearly points to superior predictive power.

Permutation importance was used to obtain a measure of how important each of the selected features are. Permutation importance is measured as a drop in the model score when values of one of the features are shuffled. The main advantage of using the permutation importance is that it does not require retraining of the model and it can deal with a mix of continuous and discrete features. It has been shown to be slightly biased in the case of correlated variables (Strobl et al. 2009), which is not a problem here, as the selected features do not contain highly correlated pairs of features.

In Table 3.4 the permutation importances of the selected features are shown on the test data in the tuned model. The features are detailed in Section 3.3.5. The most important ones are the hopcount and the number of parallel arcs, which is not surprising given that using these two values a lower bound can be given on the number of possible paths between the origin and destination nodes. Correlation of the third and first objective unavoidably is only interpreted for three objectives. Therefore it includes information about the number of objectives. That is why the correlation of first and second objectives appears less important. The closeness centrality of the origin node appears more important than the number of nodes and the number of arcs, which supports the usefulness of including local metrics of the special nodes in the model. However, it

Table 3.4: Permutation importances for the prediction of the logarithmic transformation of the number of domination tests with the selected features.

Feauture	Permutation importance	Std
hopcount between the origin and destination nodes	1.158	0.006
the number of parallel arcs	0.909	0.005
correlation of the first and third objectives	0.396	0.004
closeness centrality of the origin node	0.215	0.002
correlation of the first and second objectives	0.189	0.002
the number of arcs	0.088	0.001
mean cost direction range on arcs	0.070	0.001
harmonic centrality of the origin node	0.050	0.001
range of cost directions (obj. 1 and 2)	0.045	0.001
the number of nodes	0.017	0.000
degree diversity	0.011	0.000
square clustering of the origin node	0.006	0.000

is possible that using a larger range of network sizes would lead to a different result and these importances should be interpreted in the context of our dataset.

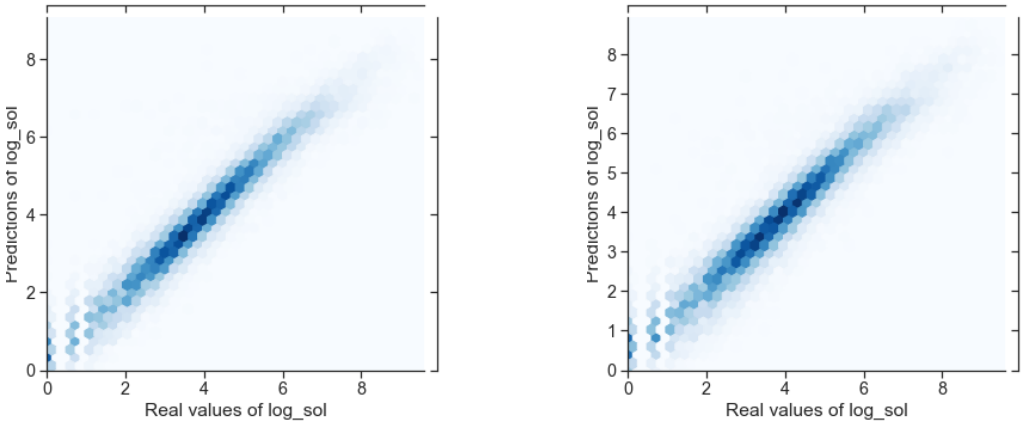
3.4.2 Number of Pareto optimal solutions as the difficulty measure

In the second part, the logarithmic transformation of the number of Pareto-optimal solutions is considered as the measure of difficulty. The experiments are repeated starting from the feature selection.

The feature selection produced different results. The major differences are that the number of nodes and arcs are not included, only their ratio. Also, there are more features describing the structure of the network. One such feature is the bridge gragnostic, which counts the number of arcs in the network such that the removal of the arc would increase the number of components in the network.

The fit achieved with the literature features and the selected features can be observed in Figure 3.3. Here the predicting power of the two models is closer than for the number of domination tests, with the selected features ($R^2 = 0.911$) still outperforming the literature features ($R^2 = 0.894$). The literature features performed similarly as in the case of the domination tests, while the selected features performed better in that case.

The permutation importances of the selected features for the tuned model are shown in



(a) Selected features, $R^2 = 0.911$ **(b)** Literature features, $R^2 = 0.894$

Figure 3.3: Goodness of fit shown on hexagon plots on unseen test data for the prediction of the logarithmic transformation of the number of Pareto-optimal solutions tests by the model with the selected features (3.3a) and the model with the literature features (3.3b) .

Table 3.5. The features are detailed in Section 3.3.5. The hopcount and the number of parallel arcs are the most important features, just as in the case of the domination tests in Table 3.4. Features describing the costs, in particular cost vector directions, are shown to be more important when predicting the number of Pareto-optimal solutions than the computational effort. The two features that characterise the origin and destination nodes are the least important among the selected features, the position of the origin node is less important than in the case of predicting the domination tests.

3.5 Discussion

In this Chapter, the magnitude of the computational effort is predicted that is required by Multigraph MSPP instances using an extreme gradient boosting regression model based on simple metrics of the network structure and the objective values. The networks used in the Multigraph MSPP instances are random network models including ones imitating properties of real world networks.

It was shown that the predictive model can potentially generalise for unseen network types, by testing on network types not seen in training. It was also shown that our selected features

Table 3.5: Permutation importances for the prediction of the logarithmic transformation of the number of Pareto-optimal solutions with the selected features.

Feature	Permutation importance	Std
hopcount between the origin and destination nodes	0.992	0.005
the number of parallel arcs	0.855	0.005
range of cost directions (obj. 1 and 3)	0.300	0.003
correlation of the first and second objectives	0.130	0.002
range of cost directions (obj. 1 and 2)	0.065	0.001
degree diversity	0.039	0.001
mean cost direction range on arcs	0.032	0.001
arcs to nodes ratio	0.016	0.001
bridge	0.014	0.000
correlation of the first and third objectives	0.013	0.000
transitivity	0.007	0.000
average degree of neighbours of the origin node	0.006	0.000
square clustering of the destination node	0.004	0.000

achieve higher accuracy in predictions as compared to using only the features that were already suggested in the literature for both of the measures of difficulty considered. The selected features include centrality metrics describing the position of the origin node within the network and novel metrics describing the costs that were not previously shown to affect the computational demand of MSPP or Multigraph MSPP instances.

Most of the features of the predictive model are relatively simple. More intricate metrics might be designed in future to provide more information not yet captured by our metrics. One promising direction is characterising local interactions between structure and costs. One way to approach this question would be designing new cost generation methods specifically for this reason, such that differences might be observed when the same cost vectors are distributed according to different rules in the network.

As mentioned earlier, our framework is not without limitations. These are mostly inflicted by the set of problem instances used for training, such as the size and the used cost generation method. These type of limitations might be overcome by a larger training set, and more deliberate manipulation of network structure and costs. Additionally, in practice, the network might be directed or the number of parallel arcs might not be uniform within the network. Resolving these limitations remains to be addressed in future work. Even within the limitations of our set of instances, improvements are almost surely possible. Future research might provide better

predictive models by either finding better features to utilise, or by building a more accurate predictive model from the same features.

This work provides a starting point for benchmark set design for the MSPP and Multigraph MSPP. A benchmark set should allow the controlled evaluation of performance with respect to the main properties that might influence the performance. Our results have shown that this set of properties includes features not previously considered, namely the centralities of the origin node and the metrics based on cost vector directions. In the next chapter, a benchmark set is proposed based on these observations for the MSPPMTW.

Chapter 4

Constrained benchmark problem generation

Benchmark problem sets are an important resource for the fields of Operational Research and Artificial Intelligence, to provide a systematic way of evaluation of algorithms. This way, they facilitate fair comparison between solution approaches and serve as a compass for researchers, showing the way towards potential improvements in designing algorithms. Benchmark sets should provide a reasonable representation of problems faced in practical applications, including all important aspects of the problem. There aren't any benchmark sets for shortest paths in multigraphs in particular, nor specifically for shortest paths with time windows, as it was discussed in 3.2.1. To fill these gaps, a novel benchmark generation framework is proposed for the MSPPMTW in this chapter.

While solution methods for targeting specific real world problems are naturally of continuous interest, such specialised studies cannot guarantee good generalisability to a different real world application. It was observed by multiple authors that depending on the problem instances, different solution algorithms might perform the best (Ehrgott & Gandibleux 2000, Beke et al. 2021). Furthermore, a widely adopted and diverse benchmark set would be important for the field to establish strengths and shortcomings of different solution algorithms, so that given an instance of the problem, some foresight might be derived into which algorithm might be the most suitable. Our proposed generator uses a high number of different random network generators to achieve high diversity in the characteristics of the network structure, and offers

methods to create instances with different characteristics regarding costs and time windows.

With the multigraph model and time window constraints, the trade-off between the quality of solutions and computational complexity becomes interesting to explore. More parallel arcs mean better chances for being able to satisfy time windows. On the other hand the computational burden increases steeply with more parallel arcs. As seen in Tables 3.5 and 3.4, this is the second most important feature for predicting the difficulty of instances. Given that the number of parallel arcs included is often a design choice, finding a good value high enough to get feasible solutions but low enough to fit into the time budget can be challenging. This problem is explored with regards to the airport ground problem in (Weiszer et al. 2020). The proposed benchmark set can help to gain insight into similar real-world problems, such as maritime transportation (Psaraftis & Kontovas 2014), multi-modal transportation (Li et al. 2020), electric vehicle routing (Wu et al. 2015), and urban rail transit (Yang et al. 2016) as reviewed in Seciton 2.5.2.

The main design considerations for the proposed benchmark set for the MSPPMTW include:

1. **The graph structure, cost matrices and time windows for each edge should be generated in a stochastic way.** Stochastic generation of the instances assures that as many different problem instances can be supplied as required with the specified properties. Generating multiple instances with similar properties might help in being able to identify further important features, when seemingly similar instances show different difficulty.

2. **Instances with characteristics resembling real world problems should be included.** This is supported by the network models used (see Section 3.2.2), and modifying the costs to reflect a simplistic model of a transportation problem.

3. **The first objective corresponds to travel time**, so compliance with time windows can be checked.

4. **Frequency and length of blocked intervals in between time windows should be controllable** as they are expected to affect difficulty and feasibility of the problem. The positioning of time blocked intervals in the network can also be controlled, if desired. The two options included are controlling (1) their proximity to destination node and (2) their propensity to be associated with central arcs. It is hypothesised that these characteristics might also affect difficulty and feasibility.

5. **The starting time of the blocked intervals is generated randomly** according to a uniform distribution.

6. **Option for including dominated costs** in the cost matrices is included to evaluate if dominated costs can help with satisfying time windows.

7. **Option for only including parallel arcs for longer** (defined in Section 4.1.1) **arcs** is included, as it is hypothesised to be more helpful.

4.1 Instance generation for the proposed benchmark set

Instances of the MSPPMTW differ from instances of Multigraph MSPP in that they include time window constraints. Time windows are assigned to the arcs and correspond to time intervals when the given arc can be used. Time intervals between two time windows on an arc are referred to as *blocked intervals*. Blocked intervals refer to the time periods when the arc cannot be traversed. Unconstrained Multigraph MSPP instances can be conceptualised as MSPPMTW instances where each arc is assigned one time window that lasts from zero to the end of the considered period, or infinity, leaving the arc available at all times.

The time window assignment method refers to how the time windows or equivalently the blocked intervals are generated for the arcs of the instance. The network structure and cost generation of the proposed benchmark is based on the previous chapter with some modifications aimed at improved realism. The modifications and the time window assignment method are explained in this section.

4.1.1 Methods for instance generation

Network structure

The generation of network structures is in line with Chapter 3.3.2. Some recommended network types to choose are Grav, for its relation to transportation problems, Prox for providing closely spatial networks resembling patterns of city maps, and NM2 for providing the most difficult instances (the definitions of the network types can be found in 3.3.2). However, network structure can also be easily customised by specifying graphs in a NetworkX (Hagberg et al.

2008) format.

Cost assignment

The cost assignment method needs to be modified compared to the last chapter in order to integrate time windows into the instances. The first objective is explicitly considered to be travel time, in line with most transportation related applications. Furthermore, to be able to decide if the time windows are violated or not, the travel times associated with each arc need to be taken into consideration. To build a rudimentary model of a real world transportation system, each arc is assigned a length, and each parallel arc is assigned a speed limit, then travel times are calculated from the lengths and speeds of each parallel arc.

Proximity graphs and grids have coordinates assigned to their nodes inherently, and therefore using them to define the length of arcs is sensible. In the case of other network types, the nodes are positioned according to the Fruchterman-Reingold force-directed graph drawing algorithm (Fruchterman & Reingold 1991), to obtain coordinates for the nodes and lengths for arcs. Speed limits are assigned to each node in the network, chosen from three possible values $50, 100$ and 150 (units do not need to be associated with the speed, because objective values for the same objective can be scaled by constants and the problem remains equivalent). Then speed limits for arcs ($v_{a,max}$) can be calculated as the average of the speed limits of the two end nodes of the arc.

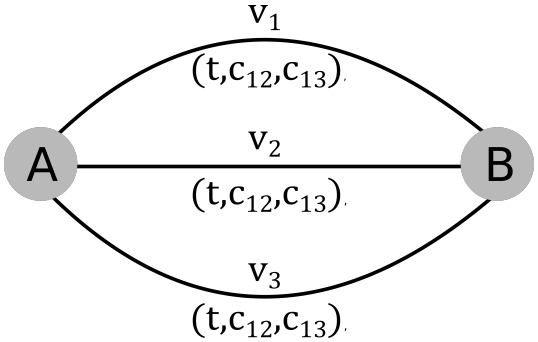


Figure 4.1: Cost assignment for the benchmark instances.

Figure 4.1 helps illustrate the cost assignment method. To model the trade-off between objectives belonging to the same parallel arc, each parallel arc is assigned a different speed between $0.5 * v_{a,max}$ and $v_{a,max}$, which are v_1, v_2 and v_3 in Figure 4.1. There is a distance d_{AB} associated with arc a_{AB}, that is shared by all parallel arcs associated with a_{AB}. Then the time

objective can be calculated as $t_i = \frac{d_{AB}}{v_i}$. Each of the further objective values are generated similarly to the method introduced in Section 3.3.3. For example, the value of the jth objective for the ith parallel arc is calculated according to Equation (4.1).

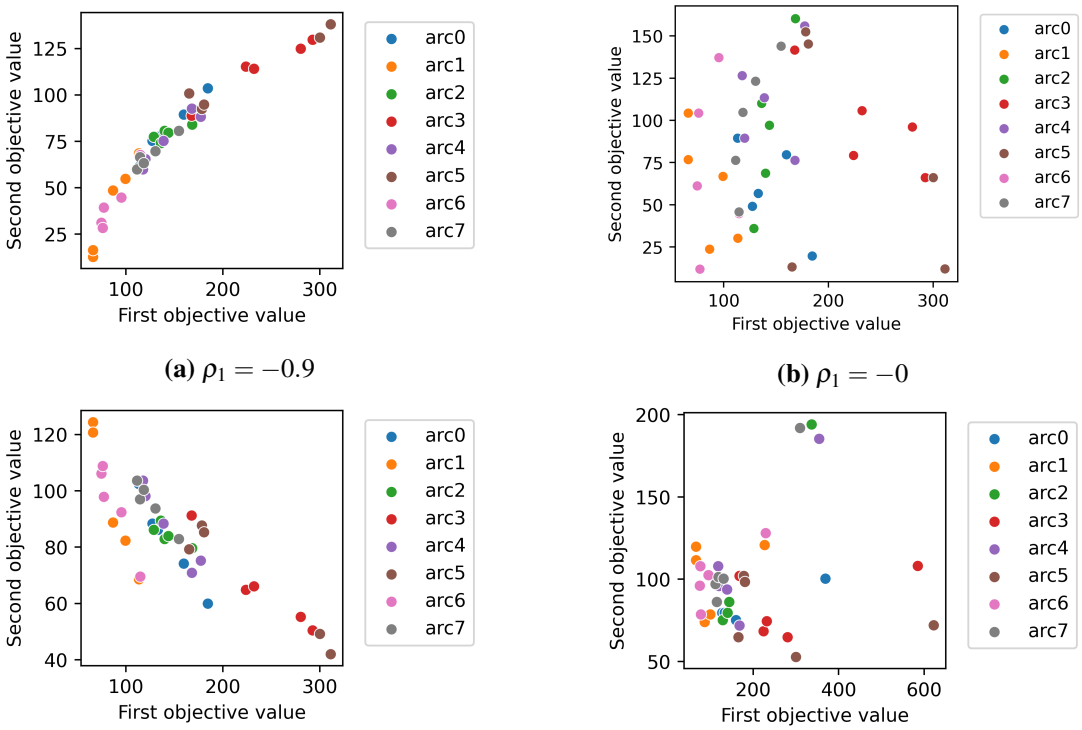
$$c_{i,j} = \begin{cases} \rho_j * v_i + (1 - \rho_j) * v^*, & \text{if } \rho_j \geq 0 \\ v_{max} + v_{min} - (|\rho_j| * v_i + (1 - |\rho_j|) * v^*), & \text{otherwise} \end{cases} \tag{4.1}$$

where v^* is randomly generated between $v_{min} = 1$ and $v_{max} = 150$. Target correlation multipliers (ρ_j) can be specified in the interval $(-1,1)$. There is no guarantee of parallel arcs having non-dominated costs, however at high values of negative correlation between the objectives, costs that belong to the same arc have a higher chance of being non-dominated.

The objective values resulting from the above introduced cost generation method for the whole network are illustrated in Figure 4.2. Each point reflects a pair of objective values on a parallel arc in a very small (4 nodes) proximity graph. Different values of ρ_1 are shown. Note that ρ_1 specifies the correlation with the speed associated with the arc, and not the correlation between the objectives directly. The first objective (time) is inversely correlated with the speed values. A strong negative correlation with the speed values gives a positive correlation between the two objectives (see Figure 4.2a), which results in the problem being similar to a single objective one. Positive correlation with the speed values results in more difficult problems with stronger trade-off between objectives, as it can be seen in Figure 4.2c). There is an option to double the costs on one of the parallel arcs on each arc in the network, illustrated on Figure 4.2d. This can be useful in time constrained routing problems when intermediate holding is not an option, or can serve as a model for stopping mid route, as that usually brings extra costs in terms of time and fuel.

The number of objectives can be specified as 2 or 3, however, the cost generation method generalises easily to a higher number of objectives. The third objective is calculated similarly as the second objective. The number of parallel arcs can be specified arbitrarily, However, computational effort required increases steeply with the increased number of parallel arcs.

There is also an option to only include parallel arcs for arcs with above-average arc-length in the network, because these can provide the most flexibility for the routing algorithm. This



(a) $\rho_1 = -0.9$

(b) $\rho_1 = -0$

(c) $\rho_1 = 0.9$

(d) $\rho_1 = -0.7$, double costs included

Figure 4.2: Various possibilities for cost generation with two objectives.

approach is expected to provide a middle-ground between using parallel arcs for all arcs for better quality of solutions and using a simple graph model for low computational complexity. Such instances are created by only keeping the first parallel arc for some arcs starting from a base instance where all arcs have the same number of parallel arcs. The base instance is also available for the purpose of comparison. Furthermore the instance created by keeping only the first parallel arc for all arcs, thereby reducing it to a simple graph can also be generated.

Common objectives such as fuel-efficiency and emission reduction are often dependent on travel speed. This cost assignment method gives a simplified way to model real world objective values in general. The model might be extended to a more nuanced formula based on vehicle type and a more detailed consideration of speed profiles for specific applications. The current cost assignment method is easy to characterise and sufficient to gain insight into the effect of time windows in general.

Time windows

A stochastic time window assignment method is designed in the following way. The length and frequency of the blocked intervals are included as the parameters for the instance generation process, because increasing either length or frequency can make it difficult to find feasible solutions for an instance. An opportunity to assign blocked intervals with higher probabilities to certain parts of the network is also included.

The two parameters are defined as follows. Blocked interval length factor (p_{len}) describes how the lengths of the blocked intervals relate to the quickest possible traversal time of the arc they are associated with. If $p_{len} = 2$, it means that the blocked intervals have approximately double the length of the quickest possible traversal time for the given arc. Blocked interval frequency (p_{freq}) is the expected number of blocked intervals on an arc per 100 time units in the generated instance.

The number of blocked intervals on arcs can be further adjusted by the blocked interval *attraction function* (g), which assigns a real value to each node in the network such that the average of the values is normalised to equal to one. The normalisation achieves that the overall length of the blocked intervals do not change. Different attraction functions can be of interest to adjust the number of blocked intervals for arcs within the instance, for example, depending on the centrality of their end nodes, to simulate higher traffic in certain parts of the network. The default is $g(a) = 1, \forall a \in A$, which is the same as not using any attraction function.

In the following two alternatives are proposed for the attraction function, which are corresponding to cases that are suspected to be more difficult to solve than the default case. Establishing if the distributions of the blocked intervals is important, not just the amount and length would be an important conclusion by itself to inform future research into benchmark problems for this problem.

The first alternative results in more blocked intervals being placed near the center of the network, which models higher traffic in the center of the network, eg. road traffic in cities. In this case the attraction function g is relating the number of blocked intervals on an arc to the centrality of the arc. This is achieved by calculating betweenness centrality of each node, normalising it such that the average of the values equals one. Then the function $g(a) = c_a, \forall a \in A$ can be used, where c_a is the normalised average betweenness centrality of the two end nodes of

a.

The second alternative for the attraction function for g is assigning a higher number of blocked intervals to arcs closer to the destination node. This is not necessarily to model a typical practical situation, but a situation that can arise through randomness and is a risk factor for the problem being difficult to solve, for example in the airport ground movement problem. In this case, the hopcounts of all the nodes to the destination node are calculated and normalises such that the average of the values equals one. Then the function $g(a) = d_{max} - d_a + 1, \forall a \in A$ can be used, where d_a is the normalised average hopcount of the two end nodes of a, and d_{max} is the diameter of I_G.

Both options serve as a test for the hypothesis that placing the blocked intervals in this way would result in differences related to the quality of solutions found and the difficulty of the problem. To make sure that any observed effect is caused by the positioning of the blocked intervals and not just their uneven distribution, it makes sense to provide an option to shuffle time windows over the arcs randomly. The shuffling is done after using the attraction function to create an instance. This way the distribution of the numbers counting the blocked intervals in the instance will be the same as before the shuffling, but the association with the positioning in the graph will be broken. Experiments including this method are presented in Section 4.2.3.

Algorithm 4 Time window assignment

Input: Unconstrained instance (I), Blocked interval frequency (p_{freq}), Blocked interval length factor (p_{len}), Blocked interval attraction function (g)
Output: Constrained instance I

1: $h \leftarrow$ hopcount between (I_O) and (I_D) in I_G
2: $t_{max} \leftarrow$ average traversal time of arcs in I with the average max speed, scaled with $h * 1.2$
3: $b \leftarrow p_{freq} * t_{max}/100$ {blocks per arc}
4: **for all** $a = v_i v_j$ in A **do**
5: $n_{tb} \leftarrow \lfloor \frac{g(v_i)+g(v_j)}{2} * x \rfloor$, where $x \propto N(b, b/2)$
6: Time windows ($\mathscr{F}_{i,j}$) are created with n_{tb} time blocks whose starting times are drawn randomly from the interval $(0, t_{max})$ and the length of each blocked interval is drawn from $N(p_{len}, p_{len} * 0.1)$ independently
7: Assign $\mathscr{F}_{i,j}$ to arc $v_i v_j$
8: **end for**

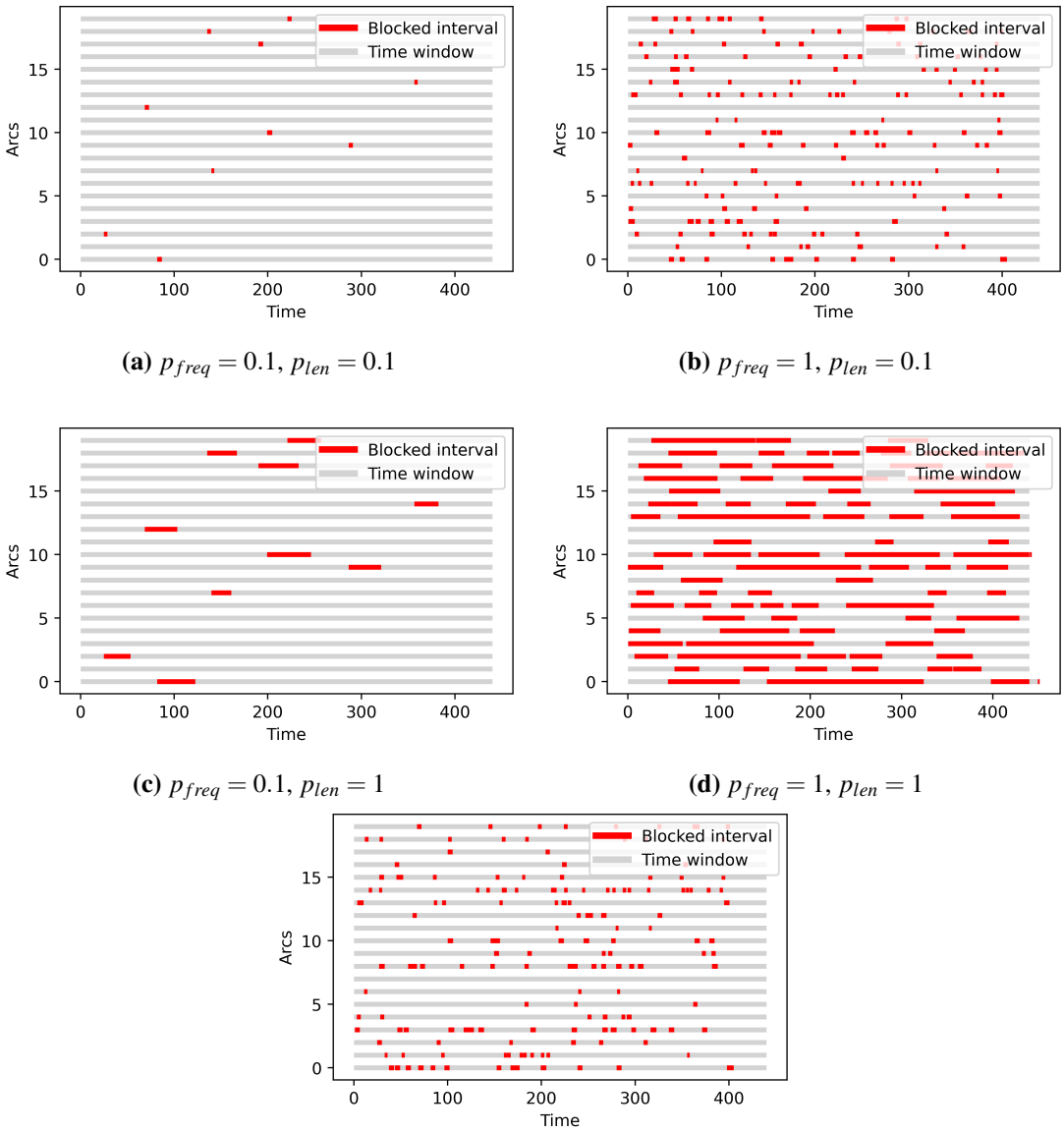
The time window assignment method is summarised in Algorithm 4. The unconstrained instance (I) consists of the multigraph $I_G = (A, V)$, the costs associated with each parallel arc, and the specification of origin (I_O) and destination (I_D) nodes. I defines all aspects of the instance except the time windows. In line 2 the end of the considered period for time window assignment

is defined. If it is too high, the solution will not depend on later time windows and the instance takes up unnecessary memory. if it is too low, it might not include the whole traversal time associated with solutions. The value of the considered period is set up as a rough estimation of how much time it might take to traverse the minimum number of arcs present between the origin and destination nodes (line 1) with a 20% wiggle room. Normal distributions are defined to draw values for the number of time blocks n_{tb} and the length of time blocks respectively. The number of time blocks to be assigned to the current arc is calculated in line 5. In line 6 time windows are created such that the starting points of blocked intervals are drawn from a uniform distribution. It is possible that blocked intervals will overlap, in this case they are considered to form a longer blocked interval together. In this case, the number of blocked intervals will be lower than the target, n_{tb}.

The parameters for time window generation are illustrated in Figure 4.3, using a proximity network with 10 nodes and 20 arcs. The difference between $p_{freq} = 0.1$ and $p_{freq} = 1$ can be seen in the number of blocked intervals created by comparing 4.3a and 4.3b. The difference between $p_{len} = 0.1$ and $p_{len} = 1$ can be seen in the length of blocked intervals by comparing 4.3a and 4.3c. Note that in the latter case, the time blocks start at the same time on the same arcs, it is just their length that changes. When both parameters are high enough, the instance becomes infeasible. This might be the case in Figure 4.3d when both parameters equal 1. Figure 4.3e shows a case with centrality based time window assignment, with more arcs having a lower number of blocked intervals compared to Figure 4.3b.

4.2 Experiments

Experimental results are presented in this section about the behaviour of the benchmark generation framework. First, the effects of the modification of the cost assignment method is explored without including time windows. Secondly, the effect of the time window generation parameters is explored on the difficulty and feasibility of the instances. Thirdly, the three hypotheses about benchmark generation are examined. The blind NAMOA* algorithm adapted to multigraph problems is used for solving the instances, as in Chapter 3. To adapt NAMOA* for shortest path problems with time constraints, at each expansion only those parallel arcs are considered that



(a) $p_{freq} = 0.1$, $p_{len} = 0.1$

(b) $p_{freq} = 1$, $p_{len} = 0.1$

(c) $p_{freq} = 0.1$, $p_{len} = 1$

(d) $p_{freq} = 1$, $p_{len} = 1$

(e) $p_{freq} = 1$, $p_{len} = 0.1$, centrality based distribution

Figure 4.3: The effect of different values of the parameters p_{freq} and p_{len} on the created time windows, and an example of adjusting the number of blocked intervals based on the centrality of the arcs.

comply with the time windows, in line with the AMOA* algorithm developed for the airport ground movement (Weiszer et al. 2020).

The methods are implemented in Python 3. All numerical tests are performed on Queen Mary's Apocrita HPC facility (Z n.d.*b*). Parallelisation has not been utilised.

4.2.1 Non-additivity of costs

NAMOA* in general will not find all possible solutions in shortest path problems with time window constraints, because of a phenomenon akin to non-additivity of costs. Dominated partial routes in general might be part of ultimately optimal origin-destination routes because of the time constraints rendering the dominating routes infeasible in later sections of the route, as illustrated in Figure 4.4. Routes containing the dominated parallel arc (dark blue) will be eliminated at node 2, even though using the quicker parallel arc (light blue) will lead to infeasibility due to the time constraint on arc 2-3 represented in red.

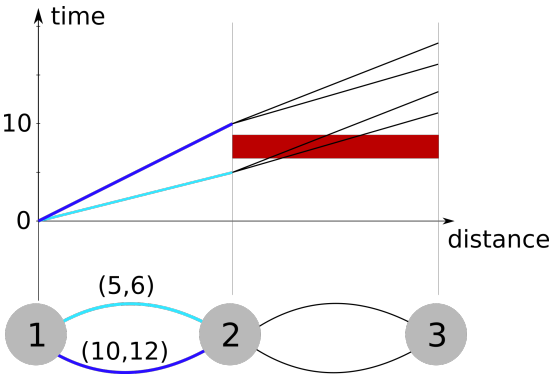


Figure 4.4: Illustration of non-additivity caused by time window constraints.

Even though dominated partial routes and dominated parallel arcs might be part of optimal solutions in the MSPPMTW, NAMOA* lacks the foresight to explore such solutions. The foresight needed to recognise such opportunities is not native to NAMOA*. In Chapter 6 a metaheuristic algorithm will be employed to avoid this problem, as a metaheuristic algorithm can be expected to find solutions that NAMOA* is not able to find.

In all experiments the origin and destination nodes are chosen to be the endpoints of a diameter of the unweighed network. All experiments are including a small number of biobjective instances. The aim of these experiments is to explore the use of the benchmark generation framework and not to characterise a representative set of benchmark instances.

4.2.2 The effect of modified costs on difficulty of unconstrained instances

The modification of the cost assignment method is aimed at making instances more realistic and expected to make them more difficult. Relating costs of an arc to its length according to an embedding in Euclidean space gives higher costs to arcs that otherwise might create a short cut in the network. This effect is expected to be limited for highly non-spatial networks, where the graph drawing algorithms (Fruchterman & Reingold 1991) cannot guarantee good separation of the nodes. Objectives also become correlated overall for the whole network, which is realistic, given that transporting for large distances usually incurs higher costs according to all objectives. However, based on the findings of the last chapter, this might make instances less difficult. In this section, it is tested if the instances become more or less difficult. The same properties are collected the predictive model trained in the last chapter is used for predicting logarithm of the number of dominations (log_dom).

For this experiment, 24 problem instances were used. Four network types (Grav, NM2, Grid and Prox, the definitions of the network types can be found in 3.3.2) were chosen, with each type being represented by six instances, with the combinations of two or three objectives and 3 or 5 parallel arcs or a single arc (parallel arcs refer to multiple distinct arcs that connect the same pair of nodes). The rest of the parameters were chosen randomly and independently from each other for each of the 24 instances from the intervals listed in the following, all within the scope of the predictive model. Number of nodes from the interval (100, 400), the ratio of edges to nodes from the interval (1.1, 4), correlation of second objective and speed values from the interval (0, 1), correlation of third objective and speed values from the interval (0, 1). For the new instances, metrics such as correlation of objectives can be calculated the same way as for the training data.

Figure 4.5 shows the predictions of the model on the new instances with the modified cost assignment method (without any time windows) compared to the test set of the last Chapter shown in gray. Based on this small experiment, the model from the last chapter still has predictive power ($R^2 = 0.53$), even with the modified costs, when time windows are not included. Therefore the modification of the cost assignment method did not render the model obsolete. Out of the 24 instances 10 remained unsolved by the end of the 58 hours allocated for the experiment. The majority, 12 of the new instances were observed to have higher actual

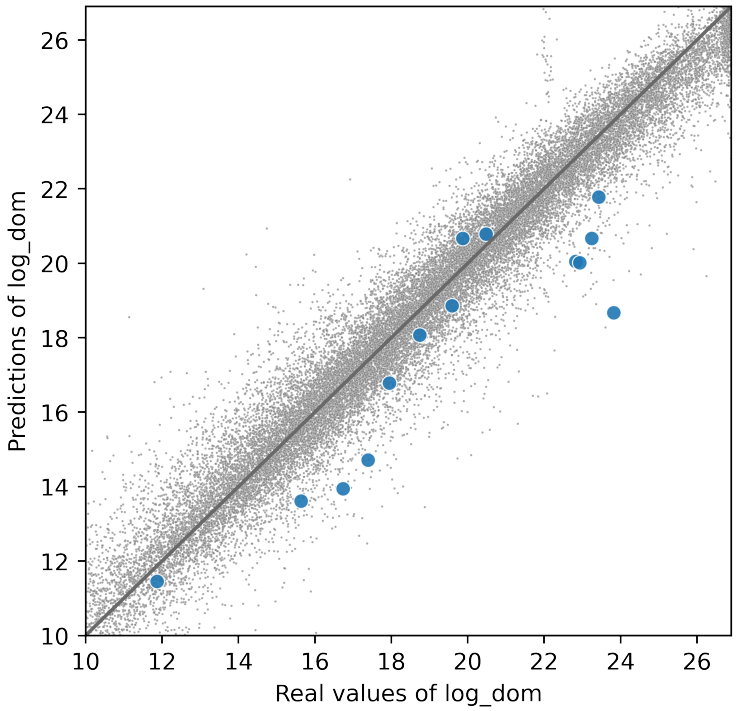


Figure 4.5: Predicted and actual difficulty of the proposed benchmark instances without time windows (blue points) against the instance set of Chapter 3. All predictions are generated with the prediction model from Chapter 3.

difficulty than the predicted and 2 had slightly lower difficulties. Therefore this experiment suggests that the modified cost generation results in more difficult instances.

4.2.3 The effect of time windows on the difficulty and feasibility

It is investigated how the time windows affect difficulty of the instance as measured by *log_dom*. A high enough number of blocked intervals makes any MSPPMTW instance unsolvable. There is a question of what values of the time window parameters make the instances infeasible. Infeasibility is understood in terms of NAMOA* stopping without finding any solutions, although, note that this might not prove none exists, as illustrated in Figure 4.4. Given that there is not any state-of-the-art exact algorithm for this problem, judging the quality of the solutions found is not straightforward in terms of what solutions are missed by NAMOA*. This question is revisited in Chapter 6. In the current chapter, time objectives of the quickest solutions for given instances are investigated. Constrained instances built on the same unconstrained instance cannot be better in terms of any objective than the best solution in terms of the given objective among

the unimpeded solutions (established without considering the time windows). An interesting question is how much worse the objectives get with increasing number or frequency of blocked intervals.

Infeasible problems are expected to have lower runtime and number of domination tests compared to the unconstrained case, because in general time constraints might make parts of the network unreachable. It is not expected that adding time windows will often increase the number of domination tests, although this is also a possibility. If an unconstrained optimal solution is blocked, longer routes might be explored within parts of the graph that would not need to be explored in the unconstrained case. This does not mean that the computational complexity of these problems is lower. It is only lower with the assumption of additive costs, which does not hold, as described in Figure 4.4.

In the following increasing the frequency or the length of blocked intervals is examined, while keeping the graph structure and costs constant. All instances in the rest of the chapter have 200 nodes, 400 arcs and 2 objectives with the second objective created from the speed values of the parallel arcs with a correlation multiplier of $\rho_1 = 0.7$.

Length of blocked intervals

The effect of the length of the blocked intervals is examined first. Five values of p_{len} are considered $(0, 0.1, 0.5, 1, 2)$ while p_{fr} is kept constant at 0.1 for 18 unconstrained base instances (network structure and costs), resulting in 90 instances. The 18 base instances were created with 1, 5 or 10 parallel arcs, three of each for two network types BA and Prox.

Figure 4.6 compares the numbers of domination tests performed for the 18 base instances with different p_{len} values on a parallel plot, where there is one line corresponding to each instance. As expected, with longer blocked intervals, the difficulty in terms of domination tests decreases most of the time, although the effect is small and some rare upward movement can also be seen. Higher values of the p_{len} parameter could be explored, where at some point a more drastic drop is possible. However, given that the frequency of the blocked intervals is low in this case, there will be a number of arcs without any blocked intervals, therefore regardless of their length, feasible solutions can be found. In contrast, when the frequency is increased all arcs will be blocked at some point, even if the length of the blocked intervals is short.

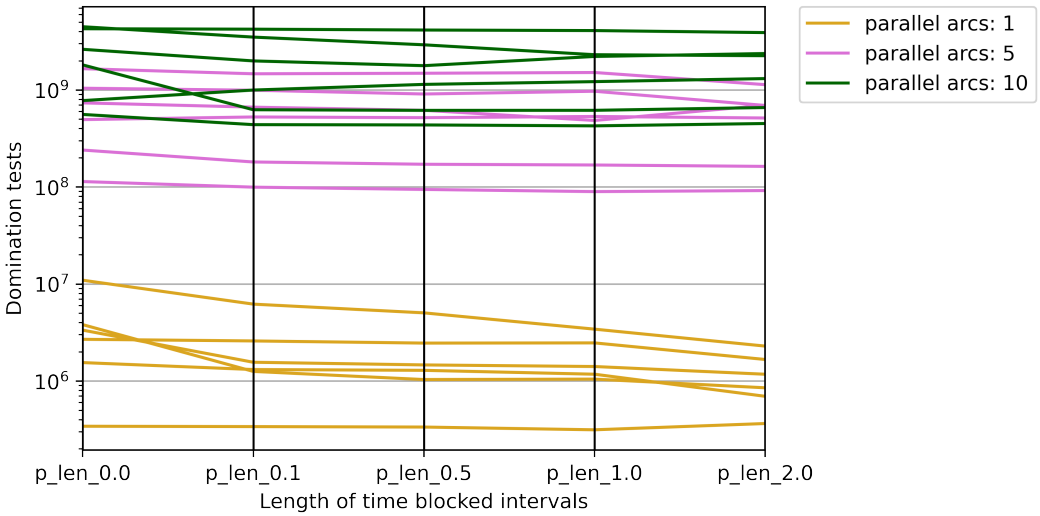


Figure 4.6: The number of domination tests for benchmark instances with an increasing time window length.

Figure 4.7 compares the number of solutions found for the 18 base instances with different p_{len} values on a parallel plot. Similar trends can be observed as in the case of the number of domination tests. A higher number of parallel arcs in general lead to more solutions than the simple graph formulation, although as all instances are feasible, it cannot be concluded if more parallel arcs help with feasibility.

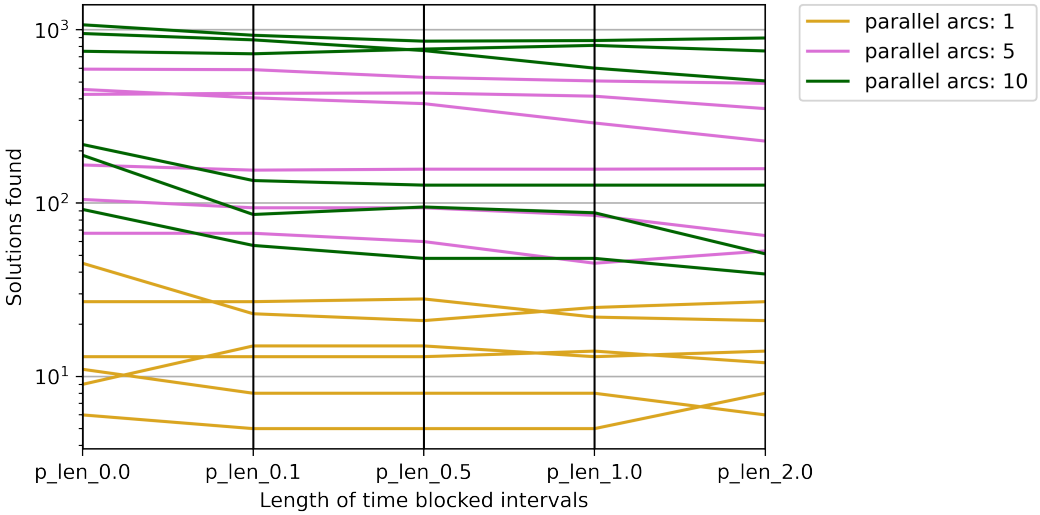


Figure 4.7: The number of solutions found for benchmark instances with an increasing time window length.

Figure 4.8 compares time objectives of the quickest solutions found for the 18 base instances with different p_{len} values on a parallel plot. Higher numbers of parallel arcs perform better in the

unimpeded case ($p_{len} = 0$). For some instances the quickest path has unchanged time objective regardless of the value of p_{len}, in the rest of the cases the quickest path becomes slower with longer blocked intervals.

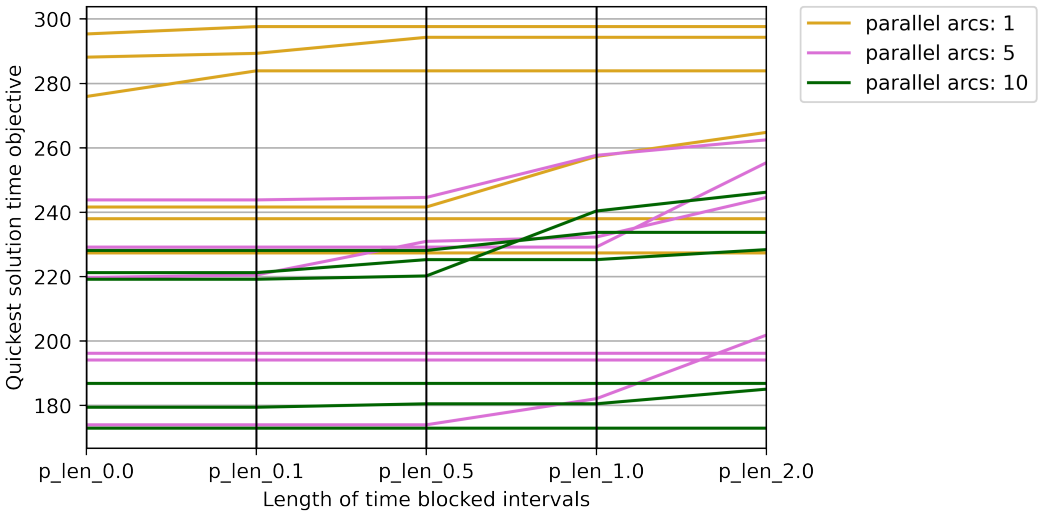


Figure 4.8: Time objective of the quickest path in the Pareto front for benchmark instances with increasing time window length.

The observed effects of the length of the blocked interval are small, which is not surprising given that even a momentary block of an arc would result in it not being traversable for the length of the minimum travel time associated with that arc. If the length would be increased to even higher values, the instances would become infeasible at some point.

Frequency of blocked intervals

Next, the effect of the frequency of the blocked intervals is investigated, in a similar way to the length. The same 18 base instances are used as above, only this time p_{fr} is taking the values $0, 0.1, 0.5, 1$ and 2, and $p_{len} = 0.1$. All instances were solved in the allocated time, although in some cases no solutions were found.

The predicted and real difficulty of the instances in terms of the log_dom values are shown in Figure 4.9, which confirms that most infeasible instances are easier to solve than their base unconstrained instances. Predictions are the same for the instances that only differ in their time windows, therefore they are positioned on a horizontal line on the plot. It can be observed that increasing the frequency of time blocks decreases the number of domination tests of NAMOA*

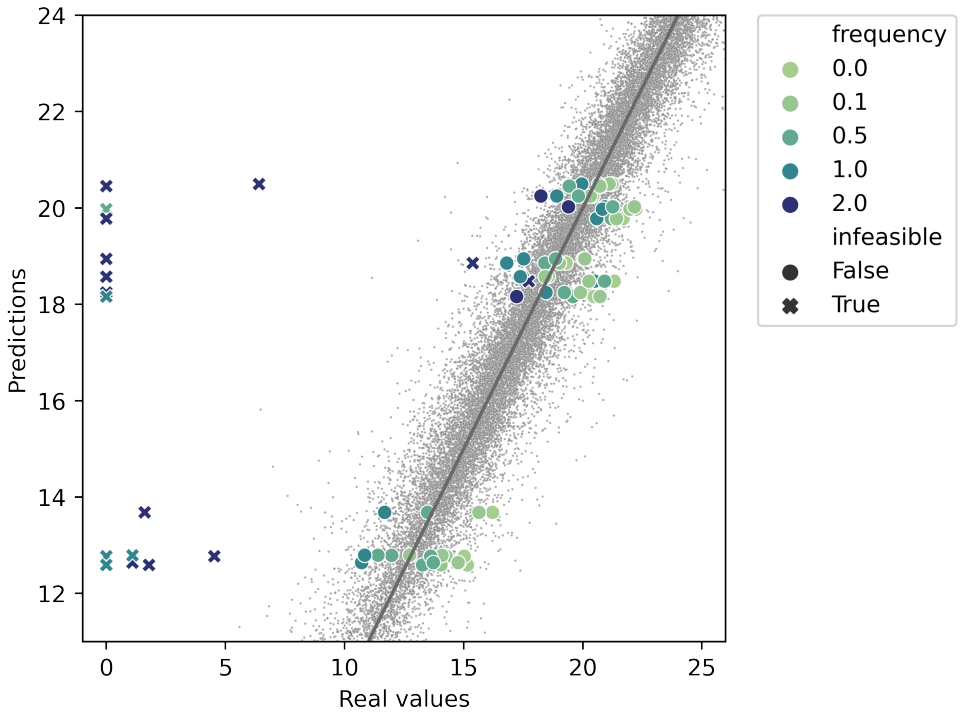


Figure 4.9: Difficulty of benchmark instances with increasing time window frequency against the prediction model from the previous chapter. Infeasible instances (instance solved within 4 days, NAMOA* returned no solutions) are differentiated.

for the most part, and this can be seen more clearly on Figure 4.10. Comparing to the test set from Chapter 3 (shown in gray) reveals that unless the instance is infeasible, its difficulty is close to the predicted value. There are also a couple of cases, where infeasible instances are comparable in difficulty to their feasible counterparts created from the same unconstrained instance. This can be explained by solution paths becoming infeasible close to the destination node.

Next, instances are differentiated based on the number of parallel arcs included, to see if more parallel arcs have any effects on the feasibility of the instances. The resulting number of solutions found are shown on a parallel plot on Figure 4.11. It can be observed that higher blocked interval frequencies lead to decreasing number of solutions, in accordance with the decreasing difficulties on Figure 4.9. It can also be seen that higher numbers of parallel arcs correspond to higher numbers of solutions. In particular the simple graph instances become infeasible sooner, most of them by $p_{fr} = 1$, while with higher numbers of parallel arcs some instances are still feasible at $p_{fr} = 2$. Note that some of the instance with 10 parallel arcs are

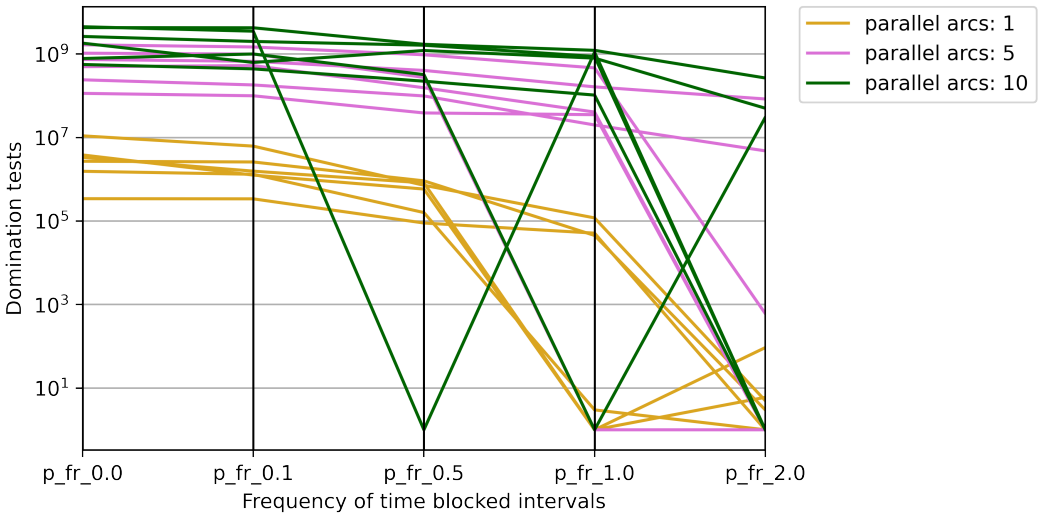


Figure 4.10: The number of domination tests for benchmark instances with an increasing time window frequency.

infeasible at some p_{fr} value, but become feasible again at a higher value of p_{fr}. This is possible, because the blocked intervals are generated from scratch each time, as opposed to adding more blocked intervals to the previous instance as p_{fr} is increased. These examples are not visible in Figure 4.11.

The increase in the time objectives of the quickest solutions found with increasing p_{fr} values is shown in Figure 4.11. Some rare downward movement can be seen, as in Figure 4.8, apart from this, the quickest solutions are getting slower as expected, as there are more blocked intervals placed on the arcs.

The effect of the distribution of blocked intervals regards to graph structure

The next experiment examines if different distributions of blocked intervals of the same length and frequency have an effect on the difficulty and feasibility of the instance. Two hypotheses are tested (1) positioning blocked intervals in the central part of the network causes instances to have lower difficulties in terms of *log_dom* then otherwise (2) positioning blocked intervals close to the destination node causes instances to have lower number of solutions.

First the effect of positioning blocked intervals in the central part of the network with higher probabilities is investigated, as described in Section 4.1.1. The 30 base instances were created with 1, 5 or 10 parallel arcs, five of each for two network types 'BA' and 'prox'. The parameters

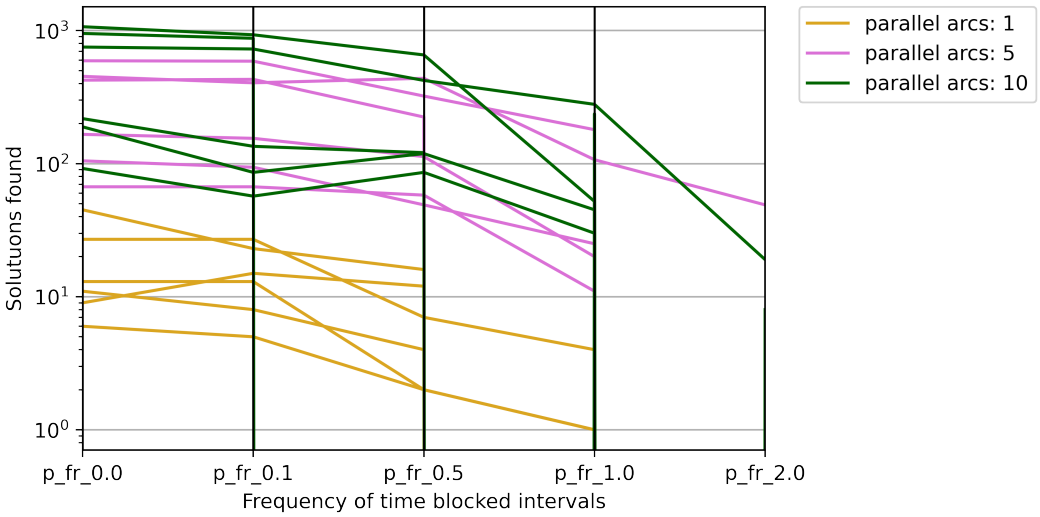


Figure 4.11: The umber of solutions returned by NAMOA* for benchmark instances with increasing time window frequency. Infeasible instances (instance solved within 4 days, NAMOA* returned no solutions) are not shown.

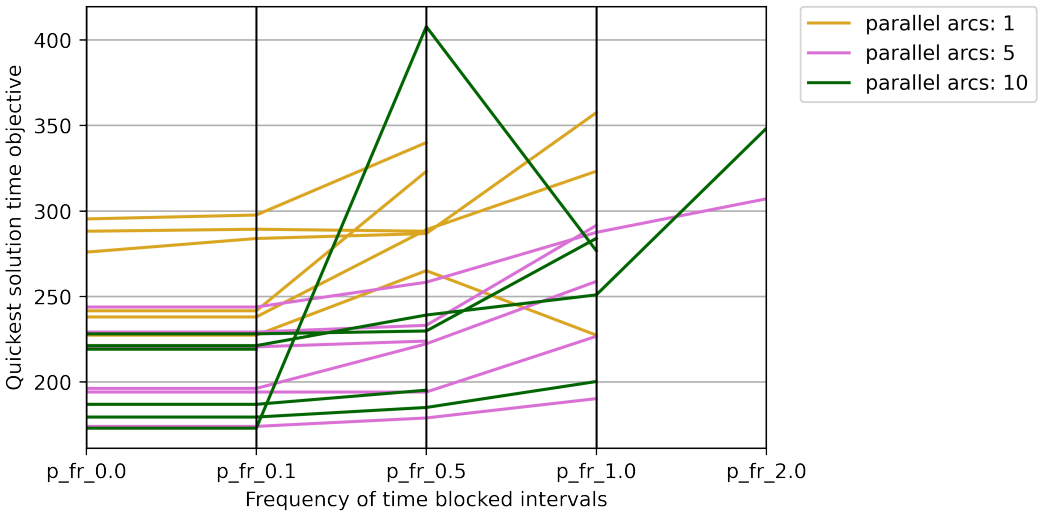


Figure 4.12: Change in time objective of the quickest solution found for different p_{fr} values.

Table 4.1: Distributions of number of domination tests depending on the position of the blocked intervals in the network regarding centrality of end nodes of the arcs.

	Mean	Std
Blocked intervals close to center	67,992,1284.73	1,078,338,110.62
Control	1,019,398,657.0	1,553,774,712.08

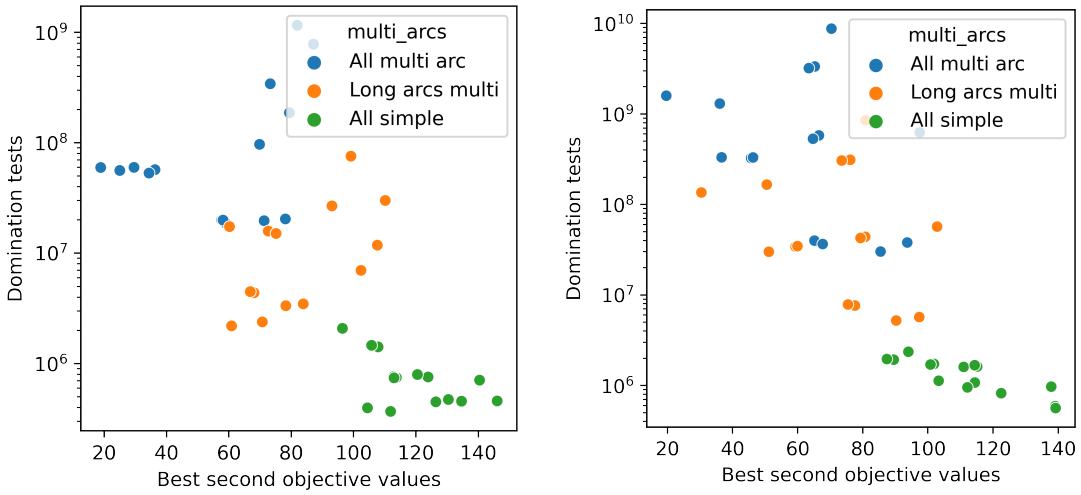
Table 4.2: Distributions of number of solutions found depending on the position of the blocked intervals in the network regarding distance to destination node.

	Mean	Std
Blocked intervals close to v_D	121.03	162.85
Control	149.23	186.75

$p_{len} = 1$ and $p_{fr} = 0.1$ were used for the experiments, and each of the 30 base instances were assigned time windows according to the centrality based assignment and the centrality control assignment. This way related pairs of instances are created with the same network structure and costs, same number of blocked intervals with similar lengths, the only difference being the positioning of the blocked intervals. All instances were solved within the allocated time.

Table 4.1 summarises the results regarding number of domination tests. Based on the one sided Wilcoxon signed rank test, the p-value is 0.0027, therefore at $p = 0.05$, the null hypotheses can be rejected, concluding that the blocked intervals assigned to arcs in proportion to their centrality decreases the domination tests performed by NAMOA*. Regarding the number of solutions, the same effect is not observable, with a p-value of 0.12, the hypothesis that the number of solutions found in the two cases is from the same distribution cannot be rejected.

Next, the effect of placing blocked intervals on arcs close to the destination node with higher probabilities is tested. Table 4.2 summarises the results. Using the one-sided Wilcoxon signed rank test, with a p-value of 0.047 it can concluded that when blocked intervals are closer to the destination node, there are fewer solutions found. No conclusion can be drawn about comparing the distributions of the number of dominations with a 0.5 confidence level, however this value is lower on average among the in the control group.



(a) Graph type: 'BA'

(b) Graph type: 'prox'

Figure 4.13: Results of removing parallel arcs from arcs with below average length compared to keeping all parallel arcs and removing parallel arcs from all arcs in terms of number of domination tests and the best second objective values among the solutions found.

Parallel arcs only for longer arcs

The last experiment aims to assess if using parallel arcs only in the case of the longer arcs can provide a compromise between solution quality and computational complexity. The three groups of instances compared either have the same number of parallel arcs for each arc, have only simple arcs, or have parallel arcs for only the arcs which have above average length. In each group there are 6 instances. These were created with 2, 5 or 10 parallel arcs, one of each for two network types 'BA' and 'prox'. The parameters $p_{len} = 1$ and $p_{fr} = 0.1$ were used for the experiments, and all instances were assigned time windows according to the random method.

Figure 4.13 shows the trade-off between difficulty of instance as measured by domination tests and solution quality as measured by best second objective values found. Because of the cost assignment method used, the quickest solutions did not show any variability, that is why best second objectives are shown. It can be observed that in both cases removing parallel arcs from shorter arcs provides a compromise between the two extremes, which includes multiple Pareto-optimal solutions.

These results suggest that considering parallel arc only for the above average length arcs can be beneficial. This is expected to be dependent on the solution algorithm used and the costs and network structure of the instance. For example, in the case of a metaheuristic algorithm

it would not be expected to see any benefit in using this method to decrease the search space. Also, in networks with larger variability in arc lengths there might be an even stronger case for using this method.

4.3 Discussion

In this chapter, a novel benchmark generation framework was proposed for the MSPPMTW, building on the network generation and cost assignment methods presented in Chapter 3. The benchmark generator offers a wide range of instances with different characteristics. Different sizes and structures of networks, cost correlation values and time window distributions might be specified. There is also an option for doubling costs on one arc for each multi-arc, which might be utilised by some algorithms to meet time windows, as later shown in Chapter 6. This can also serve as a rudimentary model for the possibility of stopping to meet time windows, as stopping usually incurs costs in terms of time and fuel.

It was shown that increasing either the length or the frequency of the blocked intervals in between time windows decreases the required runtime and the number of solutions found in most cases and worsens solution quality in terms of objective values. In the experiments of this chapter, instances without any solutions were only observed when increasing the frequency of blocked intervals. These instances typically had a simple graph structure. Note that this threshold or the existence of a threshold can be useful information in real-world routing problems, where routing decisions need to be made in a short time.

It was also shown that number of domination tests performed and number of solutions found can depend on the position of arcs with high numbers of blocked intervals. The investigated cases included higher number of blocked intervals in central parts of the network and higher number of blocked intervals close to the destination node. Lastly, it was shown that it may be beneficial to include parallel arcs only for some arcs, in particular the ones with above average length, to find a compromise between solution quality and runtime.

The chapter also serves as an example of how the benchmark set can be used. The main decisions that need to be made before generating a concrete set of instances is (1) what type of networks to include on how many nodes and arcs, (2) how many parallel arcs to include, and

if reverting to a single arc for shorter arcs is desired (3) how correlated the objectives should be, (4) length and frequency of the time blocked intervals, (5) number of objectives (two and three objectives are currently covered). Most of these decisions might be influenced by the applications in question motivating the study, if there is one. If there are real-world networks available for a given problem, their properties might be characterised by the features of the predictive model introduced in Chapter 3, and the most similar network types can be chosen by for example employing a clustering algorithm.

The benchmark generation literature has been focused on generating more and more difficult instances as mentioned in Section 3.2.1. Therefore, the instances that are the most challenging to solve deserve discussion. The most difficult instances in terms of runtime resulted from using the network type NM2, strong negative correlation between objectives and low values of the time window parameters. However, these are not the same instances that are the most interesting in terms of the question of being able to solve them or not. It is important to distinguish these two aspects, both of which have a practical importance: (1) how much computational effort does it take to solve an instance, (2) how likely it is that there won't be any solutions found. Instances with higher time window parameters might be less computationally demanding, however, solutions might be missed by standard enumerative algorithms as it will be shown in in Chapter 6. In the experiments, $p_{freq} = 1$ was observed as the value, where the majority of instances become infeasible using the enumerative algorithm (see 4.11).

Chapter 5

Genetic Algorithm for the Unconstrained MSPP on Multigraphs

5.1 Introduction

This chapter presents a multi-objective genetic algorithm to solve the MSPP on a Multigraph. This is an essential step towards designing a metaheuristic algorithm for the MSPPMTW, the goal of Chapter 6, where time windows are also incorporated. This chapter explores the possibilities of extending existing genetic representation methods for shortest path problems to a multigraph problem, and characterises the performance of the proposed algorithms on instances without time windows.

Metaheuristic algorithms received a lot of attention regarding their potential for solving network design problems (Gen et al. 2008, Fernandez et al. 2018). In particular, multiple genetic algorithm (GA) based techniques have been proposed for the simple graph MSPP (Chitra & Subbaraj 2012, Li et al. 2013, Lin & Gen 2008) to obtain a good representation of the Pareto-optimal solutions in a given time budget, where exact approaches would not be efficient enough. One important aspect of these techniques is the solution representation schemes chosen, as it defines the search for the given problem and restricts what operators can be used. Multiple such representation schemes have been proposed in the literature for the MSPP, these are the direct variable length (Ahn & Ramakrishna 2002), direct fixed length (Inagaki et al. 1999), random keys (Gen et al. 2006) and integer-valued priority (Lin & Gen 2008) representation schemes.

Each of them come with their own advantages and disadvantages as described in Section 2.4.1.

The Multigraph MSPP received little attention, even though it is relevant for a wide range of real-world problems (see Section 2.5) and a natural extension to the body of research aimed at investigating the simple graph MSPP. Section 3.4.1 in Chapter 3 has shown that shortest path problems on multigraphs are highly computationally challenging compared to the simple graph case, further supporting the case for using metaheuristic algorithms. To address this gap, the representations introduced for the simple graph problem need to be extended for use with the Multigraph MSPP. The goal of the extension is to incorporate the choice between parallel arcs in between any two nodes that follow each other in a path into the chromosome. Fig. 5.1 illustrates an example of a solution path in a multigraph. To the author's knowledge, the only previous attempts at applying metaheuristics to the Multigraph MSPP were in the context of multi-modal transportation (Abbaspour & Samadzadegan 2010, Yu & Lu 2012). Both of these studies used genetic algorithms and extend the direct variable length (Ahn & Ramakrishna 2002) representation scheme.

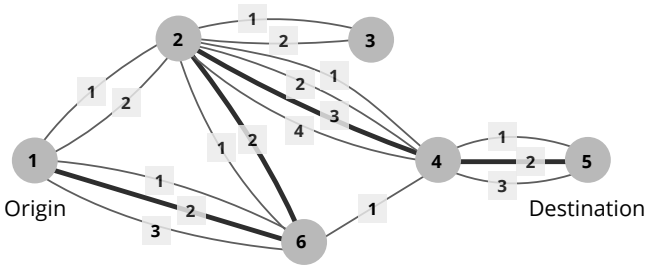


Figure 5.1: An example of a multigraph network and a solution path from node 1 to node 5 in it, indicated by the bold lines.

In this chapter, the four mentioned representations are extended to the Multigraph MSPP, three of which has not been previously used for multigraph problems. A novel way of representing the choice between parallel arcs is introduced, that allows the extension of the priority based representations to the Multigraph MSPP. A novel heuristic initialisation method is also introduced to accelerate convergence, based on information about the graph structure. The proposed techniques are tested on a diverse set of artificial problem instances using dominance-compliant performance measures. The number of objectives considered is 2 or 3 for the instances.

5.2 Proposed representations and initialisation methods

In the following, the question of incorporating parallel arc indices into the chromosomes is discussed for each of the representations. The choice of the operators is also described, along with multiple ways of initialising the population. The discussed methods build on the foundational understanding of the differences among the four representations, detailed in Section 2.4.1. In the following, the main points are summarised.

Representing paths in a graph for evolutionary computation is not straightforward, given that the length of paths can be variable. There are two options, (1) making the chromosome a variable length one (listing node IDs in a path), or (2) using a fixed length chromosome (including a gene for each node that stores some information that is redundant in decoding the path).

The direct variable length representation lists only node IDs providing a one-to-one mapping, while the other three representations have fixed length chromosomes, one gene for each of the nodes. The direct fixed length representation stores the next node to visit for each node in the graph. The priority based representations store priority values for each node, so that at any point the next node is chosen as the neighbour with the highest priority. Depending on the format of the priorities, integer-valued priority representation and random keys representation are differentiated, where the latter uses floating-point numbers as priority values.

Encoding a path as a fixed length chromosome can be described as having higher ambiguity, meaning the values of some genes are redundant for decoding the path. This property leads to larger search spaces with plateaus. On the other hand, fixed length chromosomes allow more flexibility, such that an arbitrary chromosome might be decoded to a path, therefore in general more conventional genetic operators can be utilised, without repair mechanisms.

5.2.1 Representing parallel arcs

Representations for the multigraph MSPP need to include the parallel arc indices for each pair of consecutive nodes in a solution path, as the choice of parallel arcs is also to be optimised. The previously proposed methods (Abbaspour & Samadzadegan 2010, Yu & Lu 2012) for encoding mode of transport in the multimodal transportation literature employ the direct variable length

representation, which allows for direct representation of the parallel arc indices too, not just the node sequence. Extending the priority based representations to the multigraph case is less straightforward.

First, the direct representation of parallel arc indices is described, which is only used for the direct variable length representation. The directly encoded parallel arc indices are represented as floating point numbers assigned to node IDs. The value of a parallel arc index is divided by 100 to get a floating point number, that can be conveniently stored in the same gene with the node ID it belongs to. The integer part of the gene encodes the node ID, and the fractional part the parallel arc index. This way, up to 100 parallel arcs can be represented between two nodes.

The multigraph might contain varied numbers of parallel arcs between pairs of nodes, as in the multimodal transportation problem. This does not cause a problem in the direct representations, because once the parallel arc indices are initialised to be feasible, the genetic operators do not ruin feasibility. This is because all of the arcs present in the offspring are present in the parents in crossover, at least for the one point crossover making use of common nodes, that is the most popular crossover operator for this representation. This is not the case with the priority based representations. Following a similar strategy for the priority based representations would be to assign parallel arc indices to each node in the graph. This can lead to infeasible solutions after applying genetic operators. Take an example of a crossover that results in a solution path in which a node u is followed by another node v. In both of the parent solution paths, the node u might be followed by nodes other than v. Then the parallel indices assigned to node u in the parents might be higher than what is available between u and v. A similar problem arises even if the parallel arc indices are associated with the next node to be visited.

One possible way to overcome this problem would be to apply a repairing mechanism after the solution path is decoded. This solution would increase computational burden, especially for problems with highly inhomogeneous numbers of parallel arcs in the multigraph. Furthermore, apparently, the choice between parallel arcs would be mostly governed by this repairing mechanism, not taking full advantage of the exploration and exploitation capabilities rendered by the mechanism of the evolutionary process. Therefore, an alternative indirect method for representing the parallel indices is proposed that does not require repairing for managing parallel

arcs for priority-based representation.

A floating-point number r between 0 and 1 is used, denoted the *parallel arc indicator* to indirectly encode indices of parallel arcs. Parallel arc indicators are assigned to node IDs, and they encode which parallel arc to use when leaving the given node they are associated with. Given two neighbouring nodes u, v and the number of parallel arcs between them $l(u,v)$, the index of the chosen parallel arc can be calculated as $\lfloor r(u) * l(u,v) \rfloor + 1$ when moving from u to v (indexing starts at 1). Any random value of parallel arc indicators can be decoded to an index of a parallel arc among the ones available between two given nodes. The parallel arc indicators are employed with all four representations as a way of extending them to the multigraph problem.

One possible drawback of using the above described indirect method, the parallel arc indicators, is the increased number of alternative chromosomes that can encode the same solution path, leading to the increased ambiguity of the representations. To investigate the effect of this ambiguity, the directly encoded parallel arc indices are also implemented with the direct variable length representation. This provides a way of encoding solution paths in the multigraph without any ambiguity, providing an interesting case for comparison with all other representations. The directly encoded parallel arc indices are also applicable to the direct fixed length representation, if all parallel indices are initialised to be feasible. This variant of the representation is not included in the current investigation because it already includes some ambiguity. The indirect method can be used also with the direct variable length representation.

Direct variable length representation for the Multigraph MSPP

Two versions of this representation are implemented, abbreviated *DV-indir* and *DV-dir* as mentioned above. The difference is in the encoding of the choice between parallel arcs, which is done indirectly with parallel arc indicators for *DV-indir*, and for *DV-dir* it is done directly with parallel arc indices. Note that *DV-dir* is similar to the solution representation used in (Abbaspour & Samadzadegan 2010), but different genetic operators are used. In (Abbaspour & Samadzadegan 2010), an expensive mutation operator was employed that includes using the Dijkstra's algorithm and a repairing mechanism. Instead, the mutation suggested in (Ahn & Ramakrishna 2002) is employed, as it is the most often used in the direct variable length representation.

The base of the chromosome is a sequence of node IDs, that are integers, such as in (Ahn & Ramakrishna 2002). As the parallel arc indicators are floating-point numbers below 1, it is convenient to add them to the node IDs. This way, the chromosome is the same length as in the simple graph problem. The integer parts indicate the node sequence of the path and the fractional parts indicate which of the parallel arcs to use when leaving the nodes.

The solution path can be decoded according to Algorithm 5. The algorithm loops through the chromosome (lines 3-14) and collects the nodes and indices determining the solution path. If the choice of parallel arcs is encoded directly, lines 8-9 are executed and if they are encoded indirectly, line 11-12 is used to decode the parallel indices.

Algorithm 5 Decode direct variable length chromosome for the Multigraph MSPP

Input: chromosome, G, v_O, v_D, $directIndices$
Output: nodes: node sequence, indices: sequence of parallel indices

1: $nodes \leftarrow$ List with a single element: v_O
2: $indices \leftarrow$ Empty list
3: **for** $i \leftarrow 2$ to length of chromosome **do**
4: $v_{prev} \leftarrow$ Integer part of the $(i-1)$th gene of chromosome
5: $v_{next} \leftarrow$ Integer part of the ith gene of chromosome
6: Add v_{next} to $nodes$
7: **if** $directIndices$ is True **then**
8: $frac \leftarrow$ Fractional part of the $(i-1)$th gene of chromosome
9: $idx \leftarrow 100 * frac$
10: **else**
11: $r \leftarrow$ Fractional part of the $(i-1)$th gene of chromosome
12: $idx \leftarrow \lfloor r * l(v_{prev}, v_{next}) \rfloor + 1$
13: **end if**
14: Add idx to $indices$
15: **end for**
16: **return** *nodes, indices*

An example of a chromosome with direct variable length representation that encodes the solution path in Fig. 5.1 without specifying the choice between parallel arcs is:

$$[1, 6, 2, 4, 5].$$

An example of a chromosome with *DV-indir* that encodes the solution path in Fig. 5.1 is:

$$[1.38, 6.82, 2.67, 4.51, 5.29],$$

where the value of the first gene means that the first node in the solution path is 1 and the parallel arc to be used to reach node 6 is calculated as $\lfloor r(1) * l(1,6) \rfloor + 1 = \lfloor 0.38 * 3 \rfloor + 1 = 2$. An example of a chromosome with *DV-dir* that encodes the same solution path in Fig. 5.1 is:

$$[1.02, 6.02, 2.03, 4.02, 5.29].$$

Direct fixed length representation for the Multigraph MSPP

Given that this is a fixed length chromosome, it cannot be known which nodes are in the encoded solution path without decoding the chromosome first. So, the parallel arc indicators are needed for each node, in case they are in the path. The parallel arc indicators are added to each of the genes to extend the representation introduced in (Inagaki et al. 1999). The solution path can be decoded according to Algorithm 6. The loop in lines 3-12 is executed until the destination node is reached by the path, or a node would appear a second time in the path. The next node to add to the path is identified in line 5, and if it is not yet in the node sequence, the parallel index to use is decoded in lines 10-12 and added to the parallel index sequence.

Algorithm 6 Decode direct fixed length chromosome for the Multigraph MSPP

Input: chromosome, G, v_O, v_D
Output: nodes: node sequence, indices: sequence of parallel indices

1: *nodes* ← List with a single element: v_O
2: *indices* ← Empty list
3: **while** Last node in *nodes* $\neq v_D$ **do**
4: v_{prev} ← Last element of *nodes*
5: v_{next} ← Integer part of the v_{prev}-th element of chromosome
6: **if** v_{next} is in *nodes* **then**
7: Add v_D to *nodes* {Path got stuck, will be penalised}
8: **else**
9: Add v_{next} to *nodes*
10: r ← fractional part of v_{prev}-th element of chromosome
11: *idx* ← $\lfloor$ r * $l(v_{prev}, v_{next})$ $\rfloor + 1$
12: Add *idx* to *indices*
13: **end if**
14: **end while**
15: **return** *nodes, indices*

The direct fixed length representation for the Multigraph MSPP using parallel arc indicators is abbreviated as *DF*. An example of a chromosome with direct fixed length representation that

encodes the solution path in Fig. 5.1 without specifying the choice between parallel arcs is:

$$[6,4,2,5,4,2].$$

An example of a chromosome with *DF* that encodes the solution path in Fig. 5.1 is:

$$[6.38,4.67,2.24,5.51,4.09,2.82].$$

Integer-valued priority based representation for the Multigraph MSPP

The representation presented in (Lin & Gen 2008) is extended to the multigraph case by incorporating parallel arc indicators for each node. The resulting chromosome is in the form of a 2 by n matrix M, where n is the number of nodes in the graph. Then the priority value of node v can be found at position $M_{1,v}$, and the parallel arc indicator for node v can be found at $M_{2,v}$.

The node sequence and sequence of parallel indices can be decoded according to Algorithm 7. The loop in lines 3-13 is executed until the destination node is reached by the path, or any node that only has neighbours that are already in the path. The next node to add to the path is found in line 10, and the parallel index to use is decoded in lines 12-13 and added to the parallel index sequence.

Algorithm 7 Decode priority based representations for the Multigraph MSPP

Input: M, G, v_O, v_D
Output: nodes: node sequence, indices: sequence of parallel indices

1: *nodes* ← List with a single element: v_O
2: *indices* ← Empty list
3: **while** Last element of *nodes* $\neq v_D$ **do**
4: *neighbours* ← Set of neighbours in G of last element in *nodes*
5: *allowed* ← Set of elements in *neighbours* not in *nodes*
6: **if** *allowed* is empty **then**
7: Add v_D to *nodes* {Path got stuck, will be penalised}
8: **else**
9: v_{prev} ← Last element of *nodes*
10: v_{next} ← Node with maximum priority in *allowed* according to M
11: Add v_{next} to *nodes*
12: $idx \leftarrow \lfloor M_{2,v_{next}} * l(v_{prev}, v_{next}) \rfloor + 1$
13: Add *idx* to *indices*
14: **end if**
15: **end while**
16: **return** *nodes, indices*

An example of a chromosome with Integer-valued priority representation that encodes the solution path in Fig. 5.1 without specifying the choice between parallel arcs is:

$$[6,3,1,2,5,4].$$

An example of a chromosome from this representation that encodes the solution path in Fig. 5.1 is:

$$\begin{bmatrix} 6 & 3 & 1 & 2 & 5 & 4 \\ 0.38 & 0.67 & 0.24 & 0.51 & 0.09 & 0.82 \end{bmatrix}$$

Random keys representation for the Multigraph MSPP

Random keys (Gen & Lin 2006) are floating point priority values. The extension to the multigraph case is analogous to the integer valued priority representation described above. The sequence of nodes and parallel arc indices can be decoded from a chromosome according to Algorithm 7.

An example of a chromosome with random keys representation that encodes the solution path in Fig. 5.1 without specifying the choice between parallel arcs is:

$$[0.93,0.36,0.12,0.25,0.51,0.45].$$

An example of a chromosome from this representation that encodes the solution path in Fig. 5.1 is:

$$\begin{bmatrix} 0.93 & 0.36 & 0.12 & 0.25 & 0.51 & 0.45 \\ 0.38 & 0.67 & 0.24 & 0.51 & 0.09 & 0.82 \end{bmatrix}$$

5.2.2 Variation operators for the representations

Different representations require different genetic operators. This section presents the genetic operators employed in this paper with each of the representations. For some representations, multiple crossover operators are considered, resulting in different variants of the same representations. The variants are summarised in Table 5.1 with example chromosomes that encode the same solution path for each representation.

Table 5.1: Representations and their variants for the Multigraph MSPP. Examples are shown for the encoding of the solution path in Fig. 5.1 using each representation. Variants investigated are listed for each representation. The details of the variants are described in Section 5.2.2

Representation	Variants	Example candidate	Description
Direct variable length	*DV-dir*	[1.02, 6.02, 2.03, 4.02]	Node IDs listed with parallel arc indices
	DV-indir	[1.38, 6.82, 2.67, 4.51]	Node IDs listed with parallel arc indicators
Direct fixed length	*DF*	[6.38, 4.67, 2.24, 5.51, 4.09, 2.82]	Pointers to neighbouring nodes with parallel arc indicators
Integer priority	*IP-PX*, *IP-WMX*	[6.38, 3.67, 1.24, 2.51, 5.09, 4.82]	Integer priority values with parallel arc indicators
Random keys	*RK-arithX*, *RK-2ptX*, *RK-uniX*	[(0.93,0.38), (0.36,0.67), (0.12,0.24), (0.25,0.51), (0.51,0.09), (0.45,0.82)]	Floating point priority values with parallel arc indicators

There are two variants for the direct variable length representation, the difference being the method used for encoding the parallel arcs. Both variants use the same genetic operators. Unlike the simple graph case, crossovers can be conducted even if the resulting node sequence is the same as a parent node sequence, the parallel arcs might change. The mutation operator generates new partial solutions by a random walk from a randomly chosen node in the path.

There is only one variant for the direct fixed length representation, where uniform crossover is employed. In mutation, the integer part of each gene is changed with probability 0.5. The integer part at locus i is changed to a random neighbour of the ith node. There is no need to change the fractional part, i.e., the parallel arc indicators, because they indicate a decision between a different set of parallel arcs.

In the integer priority representation, chromosomes are always a permutation of the first n integers. Therefore, the original one point crossover cannot be used. Two variants are included for the integer priority representation, with two different crossover operators, position-based crossover (PX) (Gen et al. 1997) and weight mapping crossover (WMX) (Lin & Gen 2008) that were specifically introduced for the MSPP. In the integer priority representation, chromosomes are always a permutation of the first n integers. Therefore, the original one point crossover cannot be used. WMX reorders part of the priorities in a chromosome according to the order of the corresponding priority values in another chromosome. Position-based crossover copies

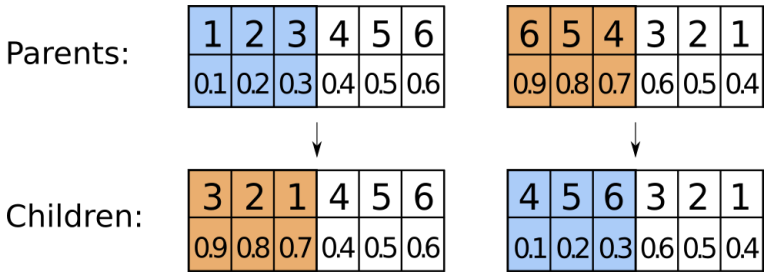


Figure 5.2: Illustration of WMX for the matrix chromosome.

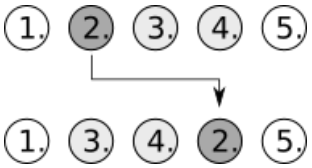


Figure 5.3: Illustration of insertion mutation.

some genes from the first parent and then fills in the missing priority values according to their order in the second parent. Weight mapping crossover similarly takes some genes directly from the first parent and then fills in the missing priority values in line with the order of the priority values in the second parent in the yet unfilled positions. The pseudo code for WMX is shown in Algorithm 8.

Insertion mutation is employed for both priority based representations. In insertion mutation, generally, a randomly picked gene (a random column in M) is removed from the chromosome and inserted back at a new random locus. The loci of genes between the place of removal and insertion change accordingly, they are shifted backward by one locus or forward by one locus depending on the inserted gene being moved forward or backward respectively within the gene. The process is illustrated in Figure 5.3, and the pseudocode is given in Algorithm 9. The p variable represents the position for insertion, and the q variable represents the position for gene deletion.

The Integer-valued priority based representation for the Multigraph MSPP using parallel arc indicators paired with WMX is abbreviated as *IP-WMX* and as *IP-PX* when it is paired with the crossover operator PX. The operators are performed on the columns of the priority-based chromosomes, such that the priority value and parallel arc indicator for a given node is derived from the same parent. The example of WMX on a two dimensional chromosome is shown in Figure 5.2.

Algorithm 8 Weight Mapping Crossover (WMX) (Lin & Gen 2008)

Input: two parents $v_1[\cdot]$, $v_2[\cdot]$, the length of chromosome n
Output: offspring $v'_1[\cdot]$, $v'_2[\cdot]$

$p \leftarrow$ random cut point between 1 and n {A random cut-point}
$k \leftarrow n - p$ {The length of right segments of chromosomes}
$v'_1 \leftarrow v_1[1:p]||v_2[p+1:n]]$ {Exchange sub strings between parents}
$v'_2 \leftarrow v_2[1:p]||v_1[p+1:n]]$
$s_1[\cdot] \leftarrow sorting(v_1[p+1:n])$ {sorting the weight of the right segments}
$s_2[\cdot] \leftarrow sorting(v_2[p+1:n])$
$i \leftarrow 1$
$j \leftarrow 1$
while $i \leq k$ **do**
  while $j \leq k$ **do**
    if $v'_1[p+1] = s_2[j]$ **then**
      $v'_1[p+1] \leftarrow s_1[j]$
    end if
  end while
  while $j \leq k$ **do**
    if $v'_2[p+1] = s_1[j]$ **then**
      $v'_2[p+1] \leftarrow s_2[j]$
    end if
  end while
end while
return $v'_1[\cdot]$, $v'_2[\cdot]$

For the random keys representation three variants are included, each with a different crossover operator. In (Gen & Lin 2006), arithmetic crossover was used, here two-point crossover and uniform crossover are also considered. Insertion mutation is used in all three cases. Both the mutation and crossover mechanisms are independent of the values of the genes and thus, they are straightforward to apply on the two dimensional genes described in Section 5.2.1. The random keys representation for the Multigraph MSPP using parallel arc indicators is abbreviated as *RK-arithX*, *RK-uniX* or *RK-2ptX* when it is paired with arithmetic, uniform or two-point crossover respectively.

5.2.3 Heuristic initialisation

Heuristic initialisation based on graph structures

Inspired by (Mohemmed et al. 2008), a novel heuristic initialisation technique is proposed here that aims to discourage detours (HeurI1). The method incorporates knowledge about the network structure and randomisation to provide a diverse initial population of high quality. The

Algorithm 9 Insertion Mutation

Input: Candidate $v[\cdot]$, length of chromosome n
Output: Mutated candidate $v'[\cdot]$

```
q ← random position between 1 and n          {Position for deletion}
p ← random position between 1 and n          {Position for insertion}
v'[·] ← copy of v[·]                         {Copy the original candidate}
if p < q then
    for i ← p + 1 to q do
        v'[i] ← v[i − 1]                     {Shift genes to the right}
    end for
    v'[p] ← v[q]
else
    for i ← q to p − 1 do
        v'[i] ← v[i + 1]                     {Shift genes to the left}
    end for
    v'[p] ← v[q]
end if
return v'[·]
```

idea is to give higher priorities to nodes closer to the destination node with higher priorities than to the nodes far from it, thereby discouraging detours.

The process starts from the random keys representation, as a chromosome from this representation is readily convertible to all the other ones. $M_{2,v}$ are initialised randomly between 0 and 1. Each node $v \in G$ is assigned a priority value according to

$$M_{1,v} = -h(v, v_D, G) + \tau, \qquad \tau \in (0, \tau_{max}). \tag{5.1}$$

$M_{1,v}$ depends on the hopcount (the minimum number of arcs in a path) from the destination node and a parameter τ_{max}. The hopcount between nodes v and v_D in G is denoted $h(v, v_D, G)$, and τ_{max} denotes the maximum value of the randomisation coefficient τ. The likelihood of detours appearing in the decoded paths can be controlled by the parameter τ_{max}. The higher τ_{max} is, the more random the priorities are, and the less prominent is the effect of the heuristic initialisation compared to a purely random one. This is illustrated in Figure 5.4. $\tau_{max} = 1.5$, and thus detours are possible, as demonstrated by the path indicated in bold, on nodes $1, 6, 2, 4, 5$. The path without detour would consist of nodes $1, 6, 4, 5$. Because the difference in the hopcounts of nodes 2 and 4 from node 5 is smaller than the difference of the random numbers associated with these nodes, a detour is formed. The hopcount information can be calculated beforehand, as it uses a simple graph and does not rely on the cost vectors and time constraints. Therefore it does

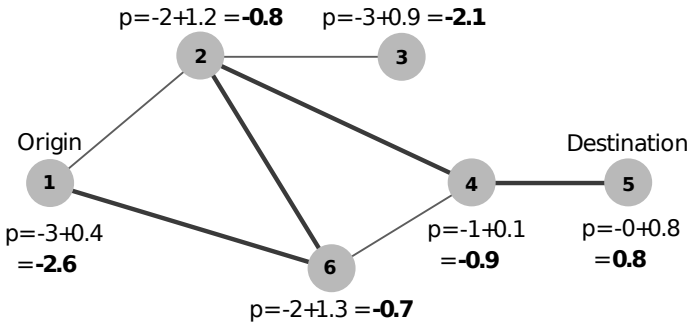


Figure 5.4: Illustration of the role of τ_{max} in heuristic initialisation.

not increase computational time.

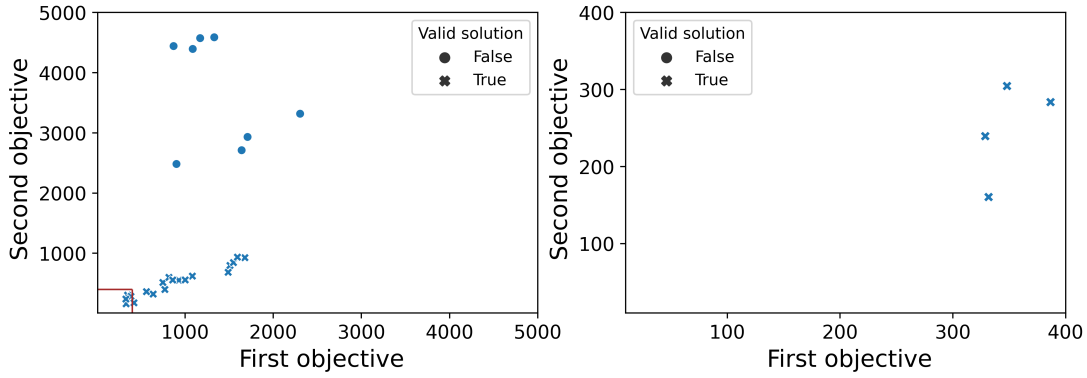
The resulting values need to be transformed according to the representations before they are fed to the algorithms as initial populations. For random keys encoding, they need to be normalised to fit the appropriate intervals. For Integer-valued priority encoding the priorities are converted to integers by sorting them into increasing order and assign to each node the rank of its priority value. For direct encodings the priorities are converted to node-based representations. This way the method can be used with all four representations. The parallel arc indicators are assigned randomly.

A crucial point is that the heuristic initialisation method should be easily computable compared to the original problem being solved. The proposed method makes use of the hopcount of each node in the graph from the destination node, which can be computed in $O(V+E)$ time. This is significantly lower than solving the Multigraph MSPP, which is in general NP-hard.

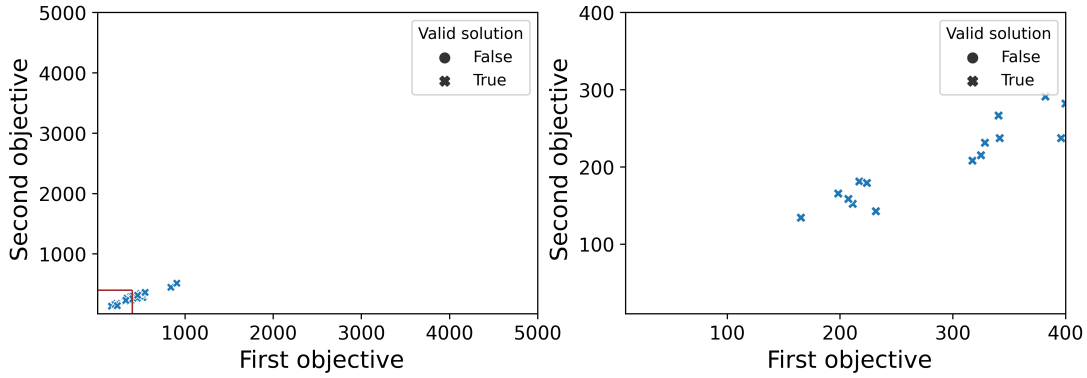
Heuristic initialisation based on single objective search

The second heuristic initialisation method (HeurI2) is adapted from (Li et al. 2013), it returns high quality solutions by using single objective search (Dijkstra's algorithm). This method is used to initialise 5 solutions for the bi-objective problem with weights for the objective values equally distributed in the interval $(0,1)$. For the triple objective problem, 7 solutions are initialised with this method. One with each single objective, one with each pair of objectives with equal weights and one considering all three objectives with equal weights. The rest of the candidates are initialised using the HeurI1 method, or a purely random initialisation.

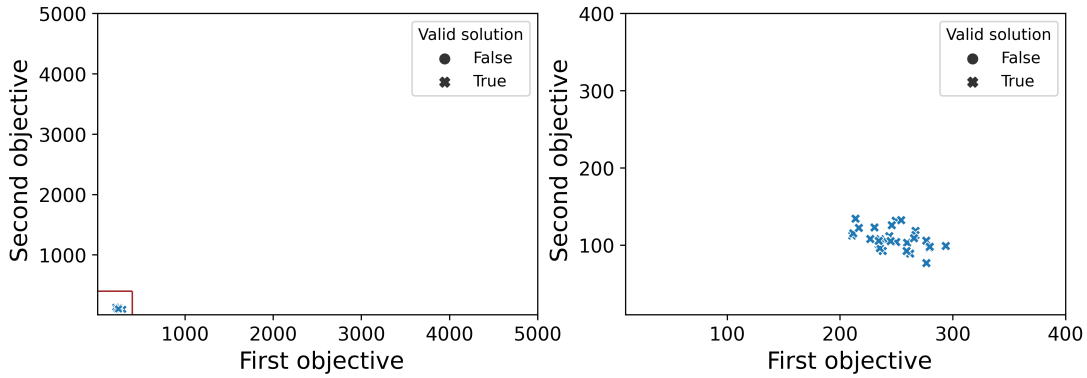
Figure 5.5 illustrates the fitness of candidates in the initial population depending on the



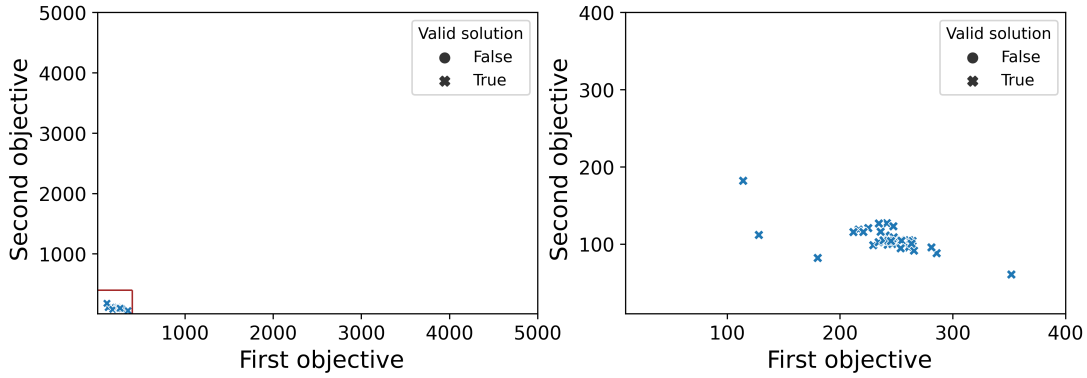
(a) Random initialisation



(b) Proposed heuristic initialisation (HeurI1), $\tau_{max} = 2$



(c) Proposed heuristic initialisation (HeurI1), $\tau_{max} = 0$



(d) Mixed initialisation (HeurI1 and HeurI2), $\tau_{max} = 0$

Figure 5.5: Fitness and validity of the initial population depending on the initialisation method and value of τ_{max}, on two different scales.

initialisation method used.

The HeurI2 method is most conveniently used with the direct variable length representation, as in (Li et al. 2013), because Dijkstra's algorithm returns a solution path. Here it is also used with the other three representations. To do this, first the solution path with the parallel indices needs to be converted to the respective representations. There are multiple possible ways to do this in each case, because only the direct variable length representation provides one-to-one encoding, the other representations do not.

For the direct fixed length representation, creating a chromosome that encodes a given solution path is relatively easy. For each of the nodes in the solution path the next node in the solution path needs to be specified, and for all other nodes a neighbour can be chosen randomly. Similarly, the parallel arc indicators need to be specified with care for the nodes in the solution path, and can be random when the node is not in the solution path.

In Algorithm 10 a method is described for producing an integer priority based chromosome that encodes a solution path specified by the node and index sequences. In the first stage, the priorities of the nodes that appear in the path are set in a loop (lines 3-7). These priorities are the highest in the graph and are increasing from the destination node towards the origin node, the origin node having the highest priority in the whole graph. This might be unintuitive, but at any point in the decoding process, the neighbour that follows the current node in the path has to have the highest priority amongst the neighbours except for nodes already in the path. The easiest way to achieve this is if it has the highest priority not only among the neighbours, but in the whole graph apart from nodes already visited. The last node in the path is not included in the loop, and therefore the values in M need to be set up separately (lines 9-10. The rest of the genes are filled up with the remaining (lower) priority values, and random parallel arc indicators in lines (12-15). These values will not be used in decoding the path, and therefore they can be filled up in an arbitrary way consistent with the given representation method. Parallel arc indicators are set to encode the required parallel indices between nodes in the solution path (lines 5-6). The translation to the remaining two representations is done along the same lines.

Algorithm 10 Encode given solution for Multigraph MSPP as integer priority based chromosome

Input: nodes, indices, G, v_D
Output: M: integer priority based chromosome
  $n \leftarrow$ The number of nodes in G
  $priority \leftarrow n$
  for $i \leftarrow 1$ to $|nodes|-1$ **do**
    $M_{1,nodes[i]} \leftarrow priority$
    $\ell \leftarrow$ number of parallel arcs between the ith and $(i+1)$th nodes in G.
    $M_{2,nodes[i]} \leftarrow \frac{indices[i]}{\ell}$
    $priority \leftarrow priority - 1$
  end for
  $M_{1,v_D} \leftarrow (priority)$
  $M_{2,v_D} \leftarrow$ random floating-point number $\in (0,1)$
  for $j \leftarrow 1$ to n **do**
    if $M_{1,j}$ is not yet specified **then**
      $M_{1,j} \leftarrow priority$
      $M_{2,j} \leftarrow$ random floating-point number $\in (0,1)$
      $priority \leftarrow priority - 1$
    end if
  end for
  return M

5.3 Implementation details

The representation and initialisation methods are compared using a multi-objective GA with non-dominated sorting and binary tournament selection with crowded-comparison and elitism (Deb et al. 2002). This strategy scales well for problems with two and three objectives (Seada & Deb 2014), however, the algorithm can be readily modified to use other multi-objective evolutionary strategies (Zhang & Li 2007, Deb & Jain 2013). The methods are implemented in Python 3, for the evolutionary computation, the inspyred package (Tonda 2020) was used. All numerical tests are performed on Queen Mary's Apocrita HPC facility (Z n.d.*b*). Parallelisation has not been utilised.

The parameters of the proposed algorithm were tuned with the use of the irace package (López-Ibáñez et al. 2016). Irace was proposed with the goal of finding the best parameter settings for an optimizer automatically. The irace package offers iterated racing procedures, a popular choice to automatically configure state-of-the-art optimisation algorithms. It samples the parameter space of the given optimisation algorithm in an intelligent manner, executing many runs of the optimisation algorithm and comparing results, which gives a more robust

Table 5.2: Values of the parameters for the different variants, tuned by the irace package.

Variants	Population size	Crossover rate	Mutation rate	τ_{max} (initialisaton randomness)
DV-dir	64	0.94	0.10	2.46
DV-indir	58	0.65	0.08	2.00
DF	28	0.89	0.04	0.63
IP-PX	44	0.62	0.12	0.30
IP-WMX	28	0.94	0.15	0.50
RK-arithX	64	0.93	0.20	0.65
RK-2ptX	38	0.95	0.17	1.02
RK-uniX	24	0.99	0.08	2.76

parameter choice than manual tuning. In this case the algorithms to be optimised and their parameters were a direct input to irace, which came back with tuned parameters as the output. The parameter tuning was performed separately for all the variants of the representations for 4 parameters: population size, crossover rate, mutation rate and τ_{max}, which controls the level of randomness in the heuristic initialisation. Tuning parameters separately for each representation achieves the best performance for each of the representations, as opposed to fixing parameters overall, which could favour one representation over another. The tuned parameters are shown in Table 5.2.

The fitness of a valid path is calculated according to Equation (5.2). It might happen that some candidates encode paths that do not reach the destination node. In these cases, the penalty term applies, which assigns a large cost to such candidates. Penalty functions for constraint handling usually include weights that control the severity of the penalty, and how much the violation of each constraint contributes to it. Here, the penalty is larger the further away the path ends from the destination node, measured by the hopcount. The fitness of an infeasible path Π' that does not reach the destination node is calculated according to Equation (5.2), where $\overline{cost_{max}}$ is the k-dimensional vector where each component equals the maximum value of any cost component in the given instance. This is used for scaling the penalty term.

$$C(\Pi') = \sum_{a \in \Pi'} \overline{cost(a)} + \overline{cost_{max}} * h(\Pi', v_D). \tag{5.2}$$

5.3.1 Test instances

The algorithms are evaluated using 32 test instances, 16 for the 2 objective problem and 16 for the 3 objective problem. These instances differ in the graph type, the maximum number of parallel arcs in the multigraph, and the correlation between the objectives. These instances are not from the generator introduced in Chapter 4, as these are unconstrained problems.

The test set includes Waxman networks (Waxman 1988) with 100 and 196 nodes and square grid networks with the same number of nodes (10 by 10 and 14 by 14 nodes). The origin and destination nodes are specified as two endpoints of a diameter (largest hopcount) of the network, to avoid setting a trivial problem. Each arc in these simple graphs is converted to a multi-arc according to a cost matrix, each row in the matrix corresponds to a cost vector for a parallel arc. The number of rows of the cost matrices is randomly chosen between 1 and l_{max}, where l_{max} the maximum allowed number of parallel arcs, 5 or 10 in this case. All parallel arcs between the same two nodes have non-dominated cost vectors. The test set includes instances with uncorrelated objectives and with negative correlation between the objectives. Negative correlation is the case where a multi-objective approach is essential for real-world applications and also these kind of instances are the most challenging for exact solution approaches (Machuca et al. 2010), which was also confirmed by the experiments conducted in Chapter 3.

The cost generation method described in Section 3.3.3 is used, where values of the two objectives are generated such that their correlation is given by the value of the correlation multiplier parameter, ρ. Instances with positive correlation are not included, because in that case, the trade-off between the objectives is smaller and therefore exact approaches can be more suitable. In the case of three objectives, the third cost component is also calculated from the first cost component with the same value of ρ.

5.3.2 Evaluation of approximate solutions

The proposed representations and their variants (listed in Table 5.1) are tested empirically and their performances are compared to a reference front using a set of performance measures. The reference front is the real Pareto front, when it is available, and an approximation of it otherwise.

Reference fronts

The real Pareto front was found by a state-of-the-art exact algorithm NAMOA* (Mandow et al. 2005), a multi-objective variant of the A* algorithm, that was adapted to the multigraph problem. This algorithm uses heuristic functions to speed up the search. Here, a heuristic function proposed in (Tung & Chew 1992) is used. It is defined as, $h_{TC}(n) = (c_1(n), c_2(n), ..., c_q(n))$, where $c_i(n)$ is the optimal scalar cost of a path from node n to the destination node, considering only the ith cost component. In the current instances, the costs are additive, and there are no time constraints, so the solutions found by NAMOA* are optimal.

One day was allowed for the execution time of NAMOA*, and when this was not enough, the Pareto front was approximated using the solutions already found by NAMOA*, if any, and the approximate solutions returned by each of the variants of the proposed metaheuristic algorithm. There are some bounds on the quality of Pareto fronts approximated this way. They contain some members of the real Pareto front, because solutions initialised with the HeurI2 method are included. When NAMOA* is stopped prematurely, it either does not return any solutions, or it returns a subset of the Pareto optimal solutions, these are also included in the reference front when they are available. Thus all other members of the approximate front are non-dominated by at least some members of the real Pareto front.

Performance measures

The multiplicative Epsilon indicator, the R3 indicator and the Relative Hypervolume (RHV) indicator are used to evaluate the approximations of the Pareto fronts, in line with the recommendations in (Fonseca et al. 2005). All three indicators signal higher qualities of the approximation front by lower values. When the approximate front is fully converged to the real Pareto front, the R3 metric and the RHV indicators have a value of 0, and the Epsilon indicator has a value of 1. For a more detailed description, see Section 2.6.

Table 5.3: The performance of NAMOA* on the 32 problem instances.

Problem Instances					NAMOA*	Reference	NAMOA*	NAMOA*
obj	graph t.	l_{max}	n	ρ	time [s]	front size	RHV 10s	RHV 20s
2	grid	5	100	-0.75	8,601.61	1247	-	-
				0.00	726.40	236	-	-
			196	-0.75	-	592	-	-
				0.00	14,917.48	472	-	-
		10	100	-0.75	62,779.48	2599	-	-
				0.00	3,536.90	416	-	-
			196	-0.75	-	506	-	-
				0.00	14,387.35	657	-	-
	waxman	5	100	-0.75	6.68	36	0.000	0.000
				0.00	11.59	48	0.072	0.000
			196	-0.75	305.88	236	0.412	0.292
				0.00	61.50	57	0.231	0.173
		10	100	-0.75	442.22	155	0.258	0.233
				0.00	9.97	69	0.000	0.000
			196	-0.75	1,787.01	368	-	-
				0.00	256.54	76	-	-
3	grid	5	100	-0.75	-	1961	-	-
				0.00	-	1316	-	-
			196	-0.75	-	2536	-	-
				0.00	-	1006	-	-
		10	100	-0.75	-	1806	-	-
				0.00	-	1355	-	-
			196	-0.75	-	3518	-	-
				0.00	-	1357	-	-
	waxman	5	100	-0.75	839.94	218	-	-
				0.00	205.06	190	-	0.405
			196	-0.75	1,654.18	567	0.609	0.590
				0.00	3,043.82	258	0.290	0.290
		10	100	-0.75	3,097.27	868	0.611	0.583
				0.00	7,178.79	651	0.445	0.390
			196	-0.75	-	946	-	-
				0.00	70,164.86	666	-	-

[a] The first five columns describe the problem instances.

[b] The column *"NAMOA* time"* specifies the running time of the NAMOA* algorithm on the given instances in seconds, if this time is less than a day.

[c] The column *"Reference front size"* specifies the number of solutions in the Pareto front, or when the algorithm does not finish in a day, the number of solutions in the approximation of the Pareto front.

[d] The last two columns describe the solution quality measured by relative hypervolume reached by NAMOA* when it is stopped prematurely after 10 or 20 seconds.

5.4 Results

5.4.1 Finding exact solutions for the Multigraph MSPP instances

Table 5.3 presents the experimental data regarding solving the 32 test instances using NAMOA*. In particular, it shows the high execution times of NAMOA*, for grid graphs this time often exceeds 24 hours, and overall it is only twice below 10 seconds. The size of the reference front is also shown, which is either the real Pareto front or its approximation, as described in 5.3.2. According to both the running time and reference front size, the bi-objective instances on Waxman graphs can be considered the easiest and the instances on grid graphs with three objectives the most difficult. The last two columns also show that if NAMOA* is stopped prematurely after 10 or 20 seconds it does not return any solutions in the case of most test instances. When it does return solutions after this short time, they are rarely a good approximation of the reference front, as measured by the RHV quality metric. These results are compared to the proposed algorithm in Section 5.4.3.

In the following, results about using the proposed algorithm with the different representations are presented and in particular it is shown that high quality approximate solutions can be found in only 10 seconds.

5.4.2 Initialisation methods

Four different initialisation procedures were used.

- *R* : Purely random initialisation
- *H1* : HeurI1 for all candidates
- *H2+R* : HeurI2 for 5 or 7 candidates as described in Section 5.2.3, and purely random initialisation for the rest
- *H1+H2* : HeurI2 for 5 or 7 candidates and HeurI1 for the rest

All eight variants of the proposed algorithm were combined with each initialisation procedure and compared the achieved solution qualities according to the performance measures in Table 5.4. H1+H2 and H2+R are visibly better than the other two in all cases. One main reason for this is that the H2 method finds the extreme solutions in the initialisation step, while these are often not found at all using a method without H2.

Apart from the direct variable length representation, the H1+H2 method significantly outperforms the H2+R. On the other hand, H2+R is significantly better than H1+H2 only in the case of the DV-dir variant. The p-values for the statistical tests are indicated in Table 5.4, compared to the best in each row. The general patterns of the results are similar in the case of the 20s time budget. For details, see Appendix Table 1.

One possible explanation for why the H1+H2 initialisation method is not better than the H2+R method in the case of the direct variable length representation is laid out in the following. In the case of the other three representations, the H1 method encodes some extra information about which direction to choose at each node in the graph, not just the nodes in the encoded path. This is possible because of the inherent ambiguity of these representations. If such a chromosome is modified through mutation or crossover, the previously redundant information might end up being used in decoding, and might have a beneficial contribution. While the direct variable length representation encodes the path without ambiguity, and thus the extra information provided by the H1 initialisation is lost, when the chromosome is converted to this representation.

5.4.3 Comparison of variants of the proposed algorithm

In the following only the best initialisations are considered for each variant. Figure 5.6 shows one example of a Pareto front found by NAMOA* in 17.4 hours compared to the solutions found by the proposed algorithm (representation: DF, initialisation: H1+H2) with a 10 second time budget. The instance used is a grid network with 100 nodes and has two objectives with strong negative correlation between them, and at most 10 parallel arcs between any nodes. It can be seen that the approximate front is very close to the real Pareto front. This example is shown as an illustration of the solution qualities in the best case for the proposed algorithm. As there were a high number of experiments conducted, showing similar plots about more instances is impractical, and in the following statistics of the performance measures are shown.

Performance of variants on average across all instances

Table 5.4 also shows that on average the RK-2ptX variant reaches the best results (with H1+H2 initialisation) considering all instances. This is confirmed by the one-sided Wilcoxon signed

Table 5.4: The role of the heuristic initialisation methods in the solution quality achieved by variants of the proposed algorithm using different genetic representations and crossover operators, described by theε, R3 and RHV metric.

	Initialisation			
	R	H1	H1+H2	H2+R
ε indicator				
DF	*** 7.7077	*** 2.7818	**1.2041**	*** 1.2487
DV-dir	*** 3.3283	*** 3.1189	* 1.2144	**1.2090**
DV-indir	3.6321	3.3199	1.2358	**1.2334**
IP-PX	*** 3.7194	*** 2.9096	**1.3228**	*** 1.3368
IP-WMX	*** 2.7993	*** 2.2668	**1.2897**	*** 1.2998
RK-2ptX	*** 3.5390	*** 2.2565	**1.1915**	*** 1.2039
RK-arithX	*** 4.0898	*** 2.7872	**1.2458**	*** 1.2731
RK-uniX	*** 4.1144	*** 2.3880	**1.2547**	*** 1.2771
R3 indicator				
DF	*** 0.6012	*** 0.2913	**0.0255**	*** 0.0300
DV-dir	*** 0.3682	*** 0.3428	** 0.0257	**0.0246**
DV-indir	0.3975	0.3625	0.0285	**0.0283**
IP-PX	*** 0.3961	*** 0.3173	**0.0439**	*** 0.0466
IP-WMX	*** 0.3122	*** 0.2389	**0.0394**	*** 0.0410
RK-2ptX	*** 0.3860	*** 0.2244	**0.0231**	*** 0.0241
RK-arithX	*** 0.4631	*** 0.3136	**0.0359**	*** 0.0396
RK-uniX	*** 0.4344	*** 0.2481	**0.0346**	*** 0.0375
RHV indicator				
DF	*** 0.5582	*** 0.1632	**0.0135**	*** 0.0180
DV-dir	*** 0.2296	*** 0.2071	* 0.0140	**0.0137**
DV-indir	0.2571	0.2231	**0.0160**	**0.0160**
IP-PX	*** 0.2652	*** 0.1864	**0.0269**	*** 0.0302
IP-WMX	*** 0.1799	*** 0.1230	**0.0227**	*** 0.0241
RK-2ptX	*** 0.2439	*** 0.1121	**0.0124**	*** 0.0132
RK-arithX	*** 0.3162	*** 0.1690	**0.0192**	*** 0.0223
RK-uniX	*** 0.2971	*** 0.1279	**0.0196**	*** 0.0229

[a] Lower values correspond to better quality.
[b] The average values are calculated from 50 runs for each of the 32 instances.
[c] The time budget is specified as 10s.
[d] The one-sided Wilcoxon signed rank test was used to decide statistical significance between H1+H2 and H2+R.
[e] Results are indicated as (*) : $p<0.05$, (**): $p<0.005$, (***): $p<0.0005$

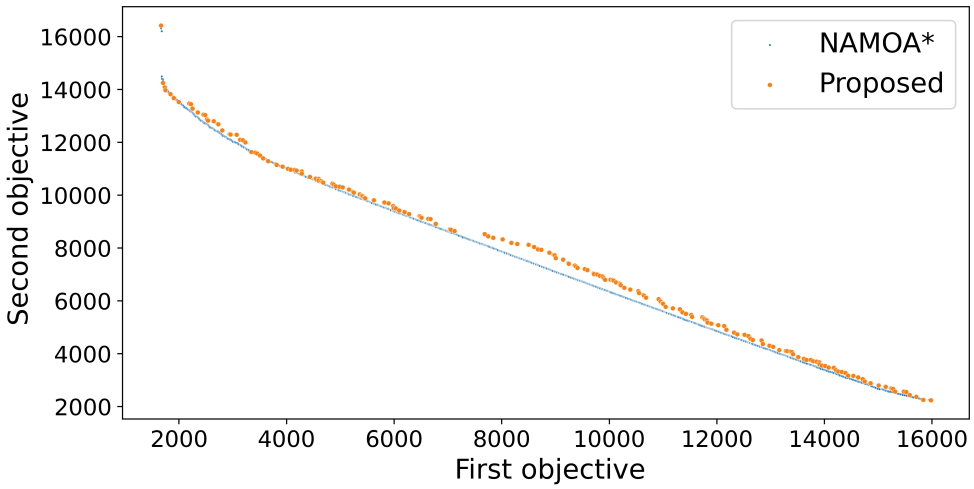


Figure 5.6: Example of a successful approximation of the Pareto front.

rank test with p-values below 10^{-7} for all three performance measures. This also holds for the 20s time budget, see Appendix8.2 Table 1 and 2.

These results might be surprising given that RK-2ptX has the most ambiguity out of the four representations. The results suggest that a different search landscape and more effective genetic operators can balance the increased ambiguity of representations. The extra information encoded in the chromosomes with some ambiguity - as discussed in Section 5.4.2 - might also contribute to the good performance.

The best variant for each of the four representations is identified based on the data presented in Table 5.4. There were two variants considered for the direct variable length representation. The DV-dir variant was better according to the averages of all three performance measures than the DV-indir. The one-sided Wilcoxon signed rank test confirms the statistical significance in all cases with p-values below 10^{-35} for both values of the time budget. This difference confirms the expectations that increasing the ambiguity of the representation of the parallel arc indices decreases the effectiveness of the search to some degree. There were two variants considered for the integer priority representation. The IP-WMX variant outperformed the IP-PX variant according to all performance measures on average. The one-sided Wilcoxon signed rank test confirms the statistical significance in all cases with p-values below 10^{-22} for both values of the time budget. This is not surprising given that WMX was specifically introduced for the shortest path problem, while PX not. There were three variants considered for the random keys

representation. The RK-2ptX variant outperformed both the RK-arithX and RK-uniX variants among all others, as it was already seen in the overall comparison of the variants.

Taking the average of the performance measures across all instances as the basis of comparison might cover up important differences between the representations. It can be the case that the RK-2ptX variant is consistently outperformed by another variant for a subgroup of the instances. This is investigated in the following.

Performance differences across the instances

In the following, only the best variant is considered for each representation with the best initialisation method. The four representations are compared in detail on the 32 instances separately. Here, the representations are compared with a time budget of 10s, based on the RHV metric. For the sake of comparison, the solution quality achieved by NAMOA* in 10s is also included. Note that the value of the Spearman's rank correlation coefficient is above 0.91 between any two of the three performance measures, this suggests that using any of the three would result in roughly the same order between the evaluated variants.

In Table 5.5, it can be seen that while RK was shown to be the best on average in Section 5.4.3, in fact three of the representations are shown to be competitive when compared separately for each of the instances. IP-WMX is outperformed by the other three in the majority of cases. DV-dir seems to be particularly suitable for grid instances with three objectives, while generally less successful in the case of Waxman networks and grids with two objectives. RK-2ptX shows an overall good performance. However, the grid instances with more than two objectives seem to be a weak point.

It can also be seen in Table 5.5, that NAMOA* outperformed the proposed algorithm with the 10s time budget in only two cases, when it found the whole Pareto front within 10s. Both of these cases are on small Waxman network. In most of the other cases no solutions are found at all in the specified time budget. Even in the cases when NAMOA* returns some solutions, the value of the RHV indicator is magnitudes higher than the variants of the proposed algorithm, indicating much poorer performance. It is worth noting that when NAMOA* is not allowed to converge, it finds solutions in the order specified by one objective, which explains the low solution quality observed with the limited time budget. Thus it can be concluded that the

Table 5.5: Comparing the best variants of the proposed algorithm for the four representations, separately for the 32 test instances with a time budget of 10s.

Problem Instance					DV-dir	DF	RK-2ptX	IP-WMX	NAMOA* 10s
obj	graph t.	l_{max}	n	ρ					
2	grid	5	100	-0.75	*** 0.0100	**0.0070**	**0.0070**	*** 0.0122	-
				0.00	*** 0.0109	*** 0.0101	**0.0062**	*** 0.0152	-
			196	-0.75	*** 0.0125	0.0078	**0.0077**	*** 0.0134	-
				0.00	*** 0.0103	*** 0.0146	**0.0079**	*** 0.0217	-
		10	100	-0.75	*** 0.0076	**0.0064**	**0.0064**	*** 0.0128	-
				0.00	**0.0058**	*** 0.0086	* 0.0067	*** 0.0169	-
			196	-0.75	0.0066	0.0066	**0.0064**	*** 0.0118	-
				0.00	**0.0083**	*** 0.0154	*** 0.0128	*** 0.0192	-
	waxman	5	100	-0.75	*** 0.0017	**0.0007**	*** 0.0012	* 0.0009	0.0000
				0.00	*** 0.0072	*** 0.0057	**0.0025**	0.0027	0.0718
			196	-0.75	*** 0.0107	0.0031	**0.0030**	*** 0.0089	0.4118
				0.00	*** 0.0052	**0.0019**	*** 0.0034	*** 0.0107	0.2314
		10	100	-0.75	*** 0.0026	** 0.0025	**0.0022**	*** 0.0026	0.2579
				0.00	*** 0.0089	*** 0.0046	**0.0037**	*** 0.0104	0.0000
			196	-0.75	*** 0.0059	*** 0.0036	**0.0031**	*** 0.0069	-
				0.00	*** 0.0130	** 0.0091	**0.0079**	*** 0.0126	-
3	grid	5	100	-0.75	**0.0082**	*** 0.0150	*** 0.0140	*** 0.0209	-
				0.00	**0.0222**	*** 0.0259	*** 0.0261	*** 0.0584	-
			196	-0.75	**0.0122**	*** 0.0183	*** 0.0176	*** 0.0267	-
				0.00	**0.0229**	*** 0.0485	*** 0.0387	*** 0.0688	-
		10	100	-0.75	**0.0129**	0.0136	*** 0.0150	*** 0.0215	-
				0.00	**0.0178**	*** 0.0374	*** 0.0397	*** 0.0659	-
			196	-0.75	**0.0072**	*** 0.0200	*** 0.0193	*** 0.0272	-
				0.00	**0.0176**	*** 0.0303	*** 0.0324	*** 0.0744	-
	waxman	5	100	-0.75	*** 0.0189	*** 0.0122	**0.0091**	*** 0.0106	-
				0.00	*** 0.0104	**0.0075**	0.0079	*** 0.0156	-
			196	-0.75	*** 0.0161	*** 0.0078	**0.0055**	*** 0.0128	0.6088
				0.00	*** 0.0366	** 0.0153	**0.0139**	*** 0.0255	0.2895
		10	100	-0.75	**0.0061**	*** 0.0103	*** 0.0082	*** 0.0117	0.6114
				0.00	**0.0169**	**0.0169**	** 0.0193	*** 0.0285	0.4449
			196	-0.75	*** 0.0421	** 0.0174	**0.0151**	*** 0.0296	-
				0.00	*** 0.0426	0.0283	**0.0276**	*** 0.0488	-

[a] The values of the average RHV performance measure of 50 runs with each representation is shown.

[b] The results of NAMOA* regarding the RHV performance measure from Table 5.3 with the same time budget of 10s are repeated in the last column, for easy comparison.

[c] The best values among the representations are shown in bold in each row.

[d] The one-sided Wilcoxon signed rank test was used to decide statistical significance between each variant compared to the best one in each row. Results are indicated as (*) : $p<0.05$, (**): $p<0.005$, (***): $p<0.0005$.

metaheuristic solution approach is preferable over NAMOA* if the time budget is small.

The general patterns of the results are similar with the other performance measures and with the time budget of 20s. For the detailed results, see Appendix8.2 Tables 3-7.

5.5 Discussion

A systematic investigation of design choices of genetic representations for the Multigraph MSPP was presented in this paper. Four main representations were investigated, some with multiple variants, resulting in eight variants in total.

Extensions to existing encoding schemes were proposed for incorporating the choice between parallel arcs through a floating-point number as parallel arc indicators, instead of including the parallel arc indices directly. This approach is necessary for the extension of priority-based representations to the Multigraph MSPP, when there are inhomogeneous numbers of parallel arcs between pairs of nodes.

Multiple different initialisation methods were also considered. Apart from purely random initialisation, two heuristic initialisation methods are investigated. One of these is an existing method based on single objective search that was previously only used with the direct variable length representation. It was adapted to be applicable to the other representations. A novel heuristic initialisation was introduced. which makes use of a priori knowledge about the graph structure and can be used with all representations.

The representations and their variants combined with different initialisation methods are compared according to three performance measures. It is found that the proposed initialisation method provides additional benefits when used together with the other heuristic initialisation methods in three out of the four representations.

Regarding the average of the performance measures across all instances, the best variant of the algorithm was the random keys representation with two-point crossover. This variant employs the novel heuristic initialisation proposed, and the parallel arc indicators to encode the choice between parallel arcs, which proves the effectiveness of the proposed methods.

When breaking down the test instances further, it was revealed that for grid based problem instances with more than two objectives, the best approach is the direct variable length represen-

tation, while the random keys representation is generally the best for the other problem instances. One future research direction is to understand this difference in depth, and design an algorithm that incorporates the strengths of both representations. The solutions obtained this way gave a good representation of the reference fronts with a time budget of only 10 seconds, while obtaining the exact Pareto front required significantly longer times for most of the investigated instances. The running times are often important in practical applications.

It was observed that the ambiguity of representing the choice of parallel arcs makes the proposed algorithm less effective. However, the inherent ambiguity of the random keys representation in encoding node sequences does not seem to cause a problem. One future direction is to understand how can the representation with the most amount of ambiguity be the most successful on average.

Chapter 6 includes the extension to problems with time constraints and refinement of the algorithm. Chapter 7 adapts the algorithm to a real-world problem, the airport ground movement problem.

Chapter 6

Genetic Algorithm for the Constrained MSPPs on Multigraphs

6.1 Introduction

In this chapter, the algorithm introduced in Chapter 5 is extended to the problem with time windows included, the MSPPMTW. The time constrained problem is less well established in the literature (reviewed in Section 2.2) compared to shortest path problems with other types of constraints. The only line of studies investigating multi-objective shortest path problems with multiple time windows for each arc (or node) are the works of Chen et. al (Chen, Weiszer, Locatelli, Ravizza, Atkin, Stewart & Burke 2016) and Weiszer et. al. (Weiszer et al. 2020) to the best of the author's knowledge, despite the relevance of the problem for a wider range of applications.

The goal in the MSPPMTW is to find the set of solution paths with non-dominated costs, among the ones that comply with the time window constraints. The time windows introduce a further level of complexity, whereby meeting a time window depends on the traversal time of the initial part of a solution path, as explained in more detail in Section 4.2.1 under non-additivity of costs. Therefore dominated partial paths might form a part of a non-dominated solution path meaning that the principle of optimality can no longer be assumed in general. Most conventional solution approaches rely on the optimality principle, notably labelling algorithms, because they eliminate dominated partial routes. Such algorithms might still be able to find good enough

solutions, but they cannot guarantee optimality. As discussed in 4.2.1, this might lead to missing optimal solutions, especially if intermediate holding is impractical or incurs costs, such as increased consumption of fuel. In this thesis, holding during the traversal of the path is not considered. A metaheuristic approach is well suited for picking up solutions that are not found by algorithms assuming the optimality principle, which is particularly important when all the solutions for a given instance are missed by such algorithms. Metaheuristics are also expected to produce results in a specified time budget, unlike the skewed running times observed for the NAMOA* algorithm in Chapter 4.

The best variant of each of the four representations from the previous chapter are used with an adapted constraint handling method to solve the MSPPMTW. In multi-objective optimisation there are multiple techniques for constraint handling as reviewed in Section 2.4.1, the approach presented here centres on the penalty function method. A local search process is introduced based on single-objective search, which is used to transform the algorithm into a multi-objective memetic algorithm. The local search operator is integrated into each variant based on the different representations. The integration of the local search operator with other operators is shown in the diagram in Figure 6.1. A small improvement in terms of the crossover operator of the direct representation is also implemented compared to the previous chapter. The algorithms are tested on instances generated with the benchmark generation framework introduced in Chapter 4. The resulting algorithm is called Memetic Algorithm for Routing in Multigraphs with Time constraints (MARMT).

Section 6.2 details the various modifications to the algorithm proposed in Section 5.2, to design MARMT. Section 6.3 compares the performance of the variants of MARMT to each other, and the best variant to NAMOA* on the same instances. It is shown that the inclusion of more parallel arcs lead to a higher chance of finding at least one solution for an instance in most cases. Secondly, it is shown that including parallel arcs with dominated costs leads to more instances being solved when the blocked intervals are rare but long. In most cases large time budgets are used, to allow enough time for convergence and give the best probability of finding solutions. The frequency and length of the blocked intervals is high enough that some instances are infeasible, therefore some part of the instances will remain without any solutions. Shorter time windows are also tested with the best performing variant.

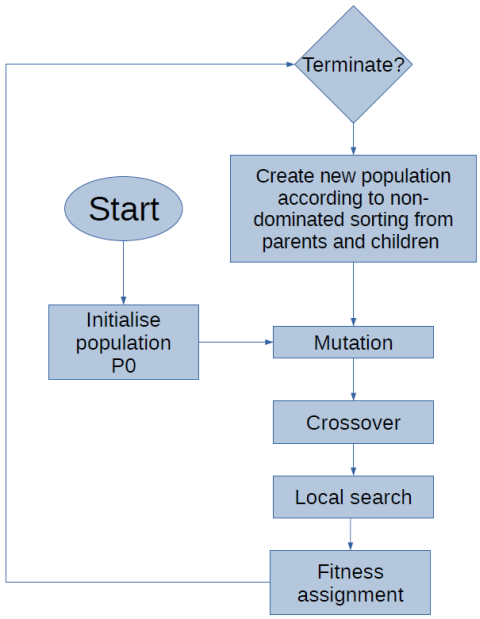


Figure 6.1: The steps of the algorithm with the local search operator.

6.2 Proposed modifications to the algorithm

The current section lays out the modifications of the algorithm compared to the descriptions in Section 5.2. The main change is the inclusion of a local search process and the change in the fitness evaluation, which all representations share. The other operators do not require any modifications. The only representation method for which two crossover operators are tested is the direct representation, where a small improvement is implemented to facilitate the exploration of parallel arcs compared to the crossover described in Section 5.2.1. The heuristic initialisation method H1+H2, H1 and R is used, as described in Section 5.4.2.

6.2.1 Constraint handling and fitness evaluation

The central difference in the problems considered in this chapter compared to the last one are the consideration of time window constraints. Combinatorial optimisation problems are inherently constrained. In Chapter 5, when time windows were not considered, the constraints were that the solution paths need to start at the origin node, follow existing arcs, and end at the destination node. These were enforced using special decoding and operators, apart from the penalty function used for ensuring that solution paths end at the destination node. In the case of incorporating

time windows, the traversal interval assigned to arcs of the path is also constrained. To account for this, the first objective is considered to reflect travel time, as seen already in Chapter 4.

The the traversal interval of the ith arc, a, in path Π, is denoted as $(T_{a,1}, T_{a,2})$, where $T_{a,1}$ is the start of the interval and $T_{a,2}$ is the end of the interval. The traversal interval can be calculated according to Equation (6.1) and (6.2).

$$T_{a,1} = \sum_{a' \in \Pi_{1,i}} c_1(a') \tag{6.1}$$

$$T_{a,2} = \sum_{a' \in \Pi_{1,i+1}} c_1(a') \tag{6.2}$$

Where $c_1(a)$ is the first cost component of the cost vector associated with arc a. The set of time windows for arc a is denoted as $\mathscr{F}_a$. For a solution path to be considered valid, the time window constraint requires the existence of a time window tw in $\mathscr{F}_a$, such that $(T_{a,1}, T_{a,2})$ fits within tw, for the solution path to be considered valid.

For solution paths, which comply with the time windows, the fitness function is defined in line with Equation (5.2). For solution paths that do not comply with time windows, the penalty term is extended. The updated fitness assignment including calculation of penalties is described in Algorithm 11. Candidates from any representations are decoded first to a path Π in the multigraph, with the respective decoding algorithms. The cost of a path Π in Line 1 is calculated by Equation (6.3).

$$C(\Pi) = \sum_{a \in \Pi} c(a) \tag{6.3}$$

To calculate the fitness value of Π, constraint violations also need to be taken into account. Static penalties are used to penalise paths that do not reach the destination node, or do not comply with time windows. The maximum value of any cost component in any cost vectors in G, denoted *maxCost* is used as a basis to establish the magnitude of the penalties (Line 2). Not reaching the destination node incurs a penalty, for which the level of violation is measured as the minimum distance of the solution path and the destination node (Line 3). For time window constraints, the level of violation is measured as the number of arcs where time windows are violated (Line 4). The level of constraint violation and *maxCost* are multiplied to give the base penalty, p_0. There are two objectives and two constraints, accordingly, four weights are set up

$(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ to calculate the fitness values (Lines 7 and 11).

These parameters were tuned by irace with respect to the algorithm and the problem at hand as with the other parameters, as explained in Section 5.3, and the values $1, 7, 5$ and 3 are used respectively in the experiments. Note that this setting can be understood to put the weight of the minimum hopcount penalty from the destination on the second objective, and the weight of the time window conflict penalty on the first objective.

Algorithm 11 FitnessAssignment(*candidate*)

Input: $\Pi :=$ decoded candidate
Output: $fitness :=$ fitness value
1: $fitness \leftarrow C(\Pi)$
2: $maxCost \leftarrow$ Maximum value of any cost component in G.
3: $minHop \leftarrow$ Hopcount between Π and v_D
4: $conflicts \leftarrow$ The number of arcs violating time windows
5: **if** $v_D \neq v_{|\Pi|}$ **then**
6: $p_0 \leftarrow maxCost * minHop$
7: $fitness \leftarrow fitness + (p_0 * \alpha_1, p_0 * \alpha_2)$
8: **end if**
9: **if** $conflicts > 0$ **then**
10: $p_0 \leftarrow maxCost * conflicts$
11: $fitness \leftarrow fitness + (p_0 * \alpha_3, p_0 * \alpha_4)$
12: **end if**
13: **return** $fitness$

6.2.2 Local search process

Memetic algorithms (MA) supplement the evolutionary process with a local search process (Moscato & Norman 1992). This is a popular extension to GAs, in order to avoid premature convergence and guide the population towards promising areas of the search space. Local search is costly in terms of computational resources and running time, and might lead to premature convergence, for these reasons, it should be employed infrequently. MA approach has been proposed for the dynamic shortest path problem in simple graphs in (Dib et al. 2018). In local search, all possible alternative partial routes were explored to replace a single arc in a route, and the one that dominated the most other alternative routes was chosen. The disadvantage of this approach is that it only replaces a single arc, and also that there might be a very high number of alternative routes, especially in a multigraph.

The local search operator employed in this work is based on Dijkstra's algorithm (Dijkstra

et al. 1959), which can solve single objective shortest path problems efficiently. In the local search, the objective values are aggregated to a single objective with random weights. A single-objective shortest path respecting time windows is found between two randomly chosen nodes in a candidate, and the newly found partial solution replaces the part between the two nodes of the original candidate. The part of solution path being overwritten is restricted to be at most a certain percentage l_{rel} of the whole length measured in hopcounts. Conducting a local search in the case of trajectories shorter than a certain minimum length l_{min} is also avoided. If no solution is found by the local search, the initial part of the candidate is returned by the local search process, up to the node from where the local search is started. The solution will be highly penalised for not reaching the destination in the fitness evaluation, as explained in Section 6.2.1. The above local search process can easily be incorporated into the direct representation. However its use is less straight-forward in other representations, just as in the case of using Dijkstra's algorithm for initialisation, seen in Section 5.2.3.

The operators are performed in the following order: mutation, crossover, local search. This way the diversity of the population is increased before crossover and the results of local search always reach evaluation without further modification.

6.2.3 Search based on direct variable length representation

Apart from the local search and the fitness function, a change in the crossover operator is proposed, to further adapt it to the multigraph case. The crossover operator is based on finding crossing sites between two parent chromosomes. A crossing site is one node or a list of sequential nodes that appear in both parents other than v_O or v_D. If there are differences in the node sequences of the parent solution paths both before and after a given crossing site, the node sequences of the offspring can differ from both parents. Such crossing sites are called *ideal crossing sites*, illustrated by Example I. in Figure 6.2. In simple graph problems only ideal crossing sites produce offspring that are novel compared to the parents. In section 5.2.2 it was mentioned that opposed to the simple graph case, the crossover adapted from (Ahn & Ramakrishna 2002) can be executed on parents with identical node sequences. As long as the parallel arcs of the parents differ both before and after the crossing site, the offspring will be novel. This is illustrated by Example III. in Figure 6.2. The above two cases have been covered

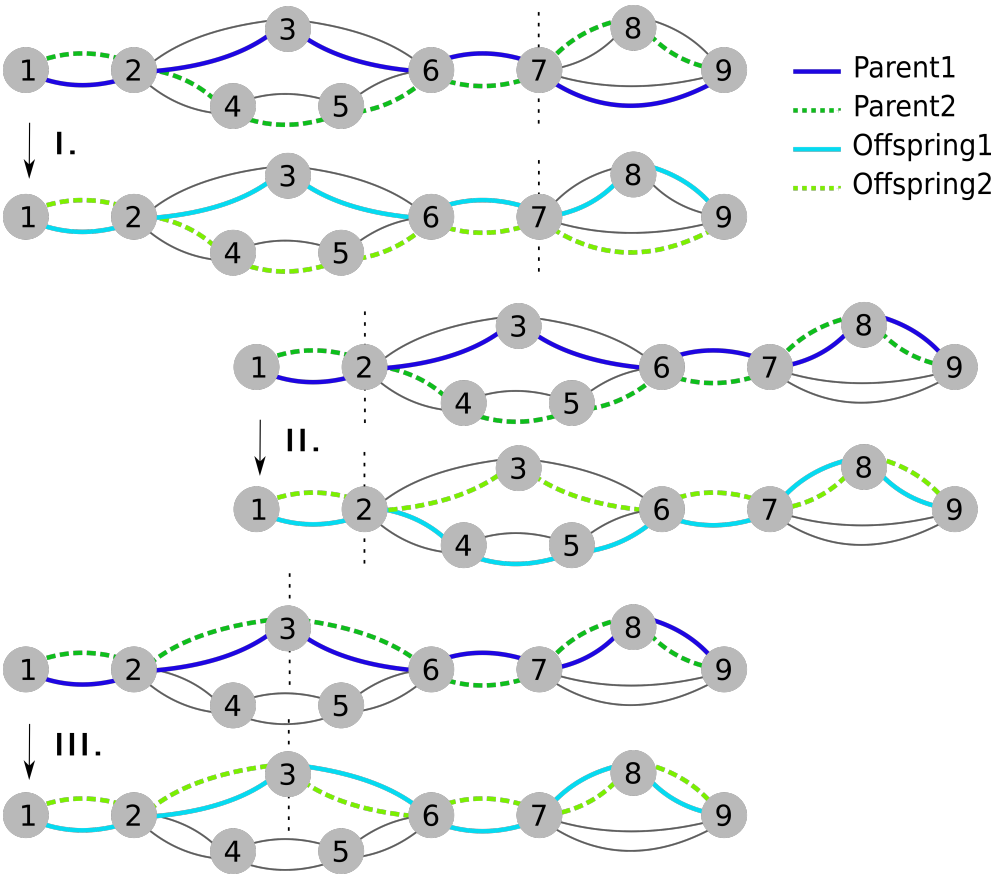


Figure 6.2: Illustration of direct crossover in multigraphs.

by the crossover operator employed in Chapter 5. However, there is a third case of interest in the multigraph problem, illustrated by Example II. in Figure 6.2. The node sequences are not identical and there is not an ideal crossing site, as the node sequences do not differ on both sides of the crossing site, only the parallel arc indices. This is sufficient for creating novel solutions by crossover in the multigraph case.

Algorithm 12 describes the different cases for the crossover process, with the above mentioned third case incorporated. The cases are based on the comparison of the two parents, which determines how the crossing site is chosen. When the parents are identical, a crossover is not possible. If only the node sequences are identical, a crossover can be performed at a randomly chosen site (Line 3), as shown in Figure 6.2 (III). If the node sequence of the parents is not identical, a randomly chosen ideal crossing site is used (Line 7). If there are no ideal crossing sites, other not ideal crossing sites are considered. There can be at most two of these at this point in the algorithm, but it is possible that there is only one. If there is only one, detected at Line 9,

this crossing site is used. If there are two, the one closer to the destination node (v_D) is used (Line 9). This is because the area close to the origin node is likely better explored, given the path generation starts from the origin node. Algorithm 12, uses the subroutine described in Algorithm 13 for executing crossovers once the crossing site is fixed. Lines 1 and 2 refer to following one parent path (including parallel arc indices) up to the crossing site and then switching to the other parent. Just like in the unconstrained case, loops may form during crossover, and these are eliminated in Line 3. The proposed modification to the crossover for the direct variable length representation is differentiated by denoting the representation with the modified crossover as "DV-1" and the representation with the unchanged crossover as "DV-2".

Algorithm 12 CrossoverOutline(parent1, parent2)

Input: P_1 := parent1, P_2 := parent2
Output: $ch1, ch2$:= Offspring
1: **if** node sequences of P_1, P_2 are identical **then**
2: $site \leftarrow$ randomly chosen node from P_1
3: $ch1, ch2 \leftarrow$ Recombine($P_1, P_2, site$)
4: **else**
5: $sites \leftarrow$ crossing sites
6: $idealSites \leftarrow$ ideal crossing sites out of $sites$
7: $site \leftarrow$ random choice from $idealSites$
8: $ch1, ch2 \leftarrow$ Recombine($P_1, P_2, site$)
9: **if** there is only one crossing site **then**
10: $site \leftarrow$ the only crossing site
11: $ch1, ch2 \leftarrow$ Recombine($P_1, P_2, site$)
12: **else if** there are two crossing sites **then**
13: $site \leftarrow$ crossing site closer to v_D
14: $ch1, ch2 \leftarrow$ Recombine($P_1, P_2, site$)
15: **else**
16: $ch1, ch2 \leftarrow P_1, P_2$
17: **end if**
18: **end if**
19: **return** $ch1, ch2$

Algorithm 13 Recombine(*parent1, parent2, site*)

Input: P_1 := parent1, P_2 := parent2, $site$
Output: $ch1, ch2$:= Offspring
1: $ch1 \leftarrow P_1$ up to $site$, and P_2 from $site$
2: $ch2 \leftarrow P_2$ up to $site$, and P_1 from $site$
3: Remove loops from $ch1$ and $ch2$
4: **return** $ch1, ch2$

6.2.4 Other representations

The encoding scheme and the genetic operators proposed for the Multigrapph MSPP in Chapter 5 do not need to be modified in order to be used for the MSPPMTW, except for the fitness evaluation already laid out in Section 6.2.1. However, the integration of the local search operator elicits some further discussion. Dijkstra's algorithm operates on a direct representation of the graph, therefore it cannot be used directly with any of the fixed length representation methods. The techniques proposed in Chapter 5 can be employed to convert chromosomes from direct variable length representation to the other representations, thereby making it possible to use the local search operator with them. All fixed-length chromosomes are decoded before executing local search on them by their respective decoding algorithms. After the local search has been conducted, the candidate is converted back to its original representation. For priority-based representations this is achieved by Algorithm 10, and for the direct fixed-length representation, it is done in a similar way.

It remains to be seen if the computational overhead of converting candidates between different representation schemes is worthwhile in the case at hand. One could imagine local search processes that operate within the bounds of priority based representations, discovering the neighbourhood of the current chromosome with one or two changes in the priority values. However, such exploration becomes expensive when a high number of genes are being changed at a time, moreover, there is no guarantee that a better solution is within reach of such a simple operator. In contrast, at the price of converting the candidate between representations, there is a good chance of finding some useful solution with Dijkstra's algorithm. The same argument also holds for the direct fixed-length representation.

6.3 Results

Multiple variants of MARMT are considered with differences in the representation method, initialisation and local search. Local search might render the heuristic initialisation methods redundant, especially the one based on single objective search. To measure the contribution of initialisation and local search, four variants considered for each representation: H1+H2 with and without local search, H1 with local search and random initialisation with local search. There

were 20 variants considered in this experiment, four different combinations of initialisation and local search paired with each of the 5 representation methods, where DV-1 and DV-2 are counted as separate representations.

The enumerative NAMOA* algorithm is also employed to solve the same instances, without the use of a heuristic function. Two days were allocated to solve each instance with NAMOA* and after the two days the algorithm is terminated regardless of its success. Reference fronts are created along the lines described in the last chapter (Section 5.3.2), by creating a unified Pareto front from the solutions returned by either NAMOA* or any variants of MARMT.

A test set created with the benchmark generator proposed in Chapter 4 is used. The test set is comprised of instances with the following properties:

- Four network types (NM2, BA, Prox, Grav from Chapter 4),
- Three sizes of networks (100, 300, 500 number of nodes),
- Three values for number of parallel arcs (1, 5, 10),
- Two values of the correlation multiplier ρ_1 (0.0, 0.7),
- Eleven different settings of the two time window parameters, including one without time windows, and 5 with varied values of the parameter of frequency of time blocked intervals (p_{freq}) and 5 with varied values of the parameter characterising the length of time blocked intervals (p_{len}),
- All instances have two objectives,
- All instances have origin and destination nodes specified as end nodes of a diameter of a network.

In total 36 multigraph network structures are considered, each with two different sets of cost vectors. Each network with given cost-vectors is paired with 11 sets of time windows.

The methods are implemented in Python 3, for the evolutionary computation, the inspyred package (Tonda 2020) was used. All numerical tests are performed on Queen Mary's Apocrita HPC facility (Z n.d.*b*). Parallelisation has not been utilised.

Table 6.1: Number of runs returning at least one valid solution with each of the 20 considered variant.

Representation	LS rate	Initialisation	Feasible found
DV-1	0.0	H1+H2	4126
DV-1	0.02	H1+H2	*4871*
DV-1	0.02	H1	*4846*
DV-1	0.02	Random	***4881***
DV-2	0.0	H1+H2	4132
DV-2	0.02	H1+H2	*4868*
DV-2	0.02	H1	4842
DV-2	0.02	Random	*4872*
DF	0.0	H1+H2	3532
DF	0.02	H1+H2	4499
DF	0.02	H1	4480
DF	0.02	Random	4608
IP	0.0	H1+H2	3738
IP	0.02	H1+H2	4477
IP	0.02	H1	4476
IP	0.02	Random	4355
RK	0.0	H1+H2	3986
RK	0.02	H1+H2	4591
RK	0.02	H1	4564
RK	0.02	Random	4558

6.3.1 Comparing variants

The performance of different variants of MARMT are compared. Some of the instances with high values of p_{freq} are expected to be infeasible. Some feasible instances are expected to be difficult to find solutions for. Therefore, the primary criteria for comparison is the proportion of instances solved by the variants of MARMT. Secondarily, the quality of the solutions found is considered, as judged by the performance measures. Comparison is presented based on both convergence, and a time budget of 30 seconds for some variants of MARMT.

Table 6.1 shows how many times the algorithm returned at least one valid solution. For each of the 20 variants, there are 10 runs conducted for each of the 792 instances, resulting in 7920 runs for each variant. The best five values are highlighted in italic and the best value is highlighted in bold. It can be observed that the direct variable length representation finds at least one solution in the largest proportion of the runs, with DV-1 consistently outperforming DV-2, when local search is used. All variants perform better with local search than without local search, as expected.

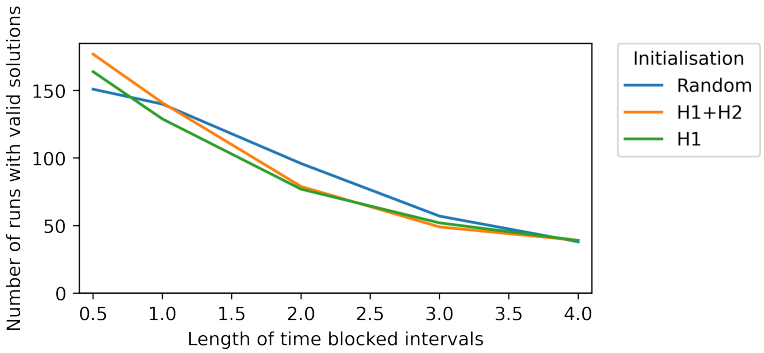
Somewhat surprisingly, the random initialisation with local search finds at least one solution most frequently among the considered variants. This might be because the information used

to initialise the populations is less useful in case of the instances with more time windows, because detours are needed to satisfy time windows. To visualise the effect of the initialisation methods at different values of the time window parameters, line plots are shown in Figure 6.3 and Figure 6.4. The sub figures show different network types, as there are differences in the results depending on the network types.

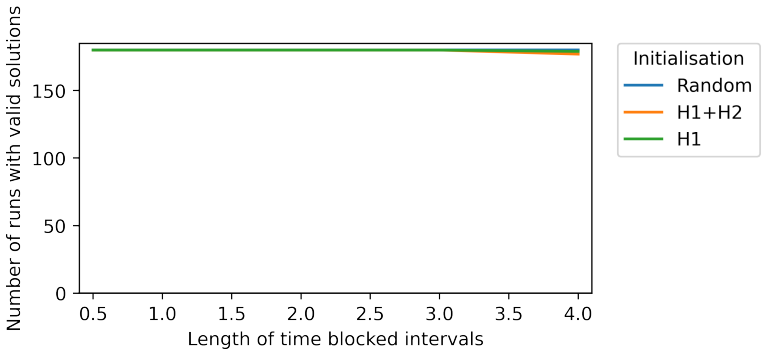
Figure 6.3 includes the set of instances with fixed p_{freq}, so the number of runs producing any valid solutions can be visualised with the change in p_{len}. All sub figures are at the same scale. It is visible that the NM2 type network structure is the hardest to find solutions for, and this is the only case where the difference between the initialisation methods is apparent, random initialisation producing the best results at the mid values of p_{len}. The difference among the network types might be explained by the large diameter of NM2 graphs. With the same number of time blocked intervals in different types of networks, a higher proportion of blocked intervals will affect solution paths in networks where the hopcount between origin and destination is higher. As the blocked intervals become longer, it only affects the existence of valid solutions if the blocked intervals are positioned on arcs of interest, in between origin and destination. There are simply more of these arcs in the NM2 case.

Figure 6.4 includes the set of instances with fixed p_{len}, so the number of runs producing any valid solutions can be visualised with the change in the values of p_{freq}. Time blocked interval frequencies have a stronger impact on finding valid solutions for all types of graphs, however, the superiority of the random initialisation is again only obvious in the case of NM2. The findings do not support the hypothesis that the time windows make the random initialisation better suited for the problem.

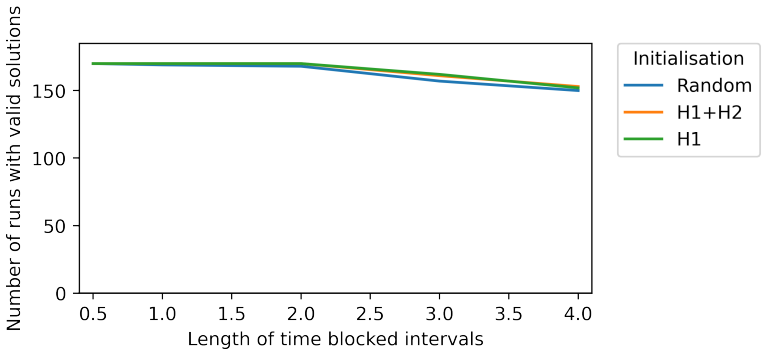
It is also important to know the quality of the returned solutions. For this, the results regarding the average of the performance measures are shown in Table 6.2. The DV-1 representation with local search and H1+H2 initialisation method is the best according to all perfromance measures (highlighted in bold). For each performance measure, the best five values are highlighted in italic. This includes only the direct variable length representations, apart from two values of the relative hypervolume with random keys and direct fixed-length representations. Still, generally there is a high level of agreement between the three performance measures regarding average of the performance measures, and more also with the number of runs producing a valid



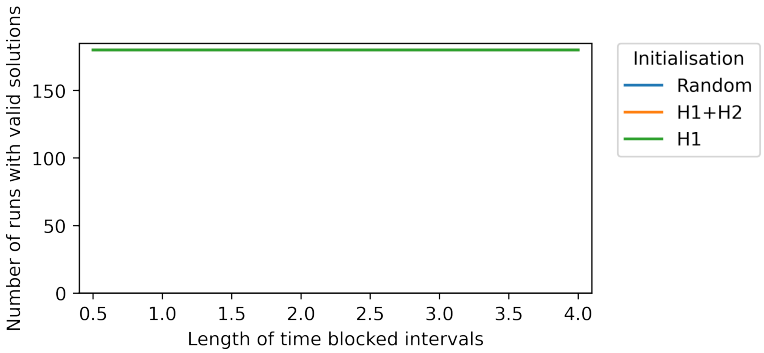
(a) "NM2" network



(b) "Prox" network

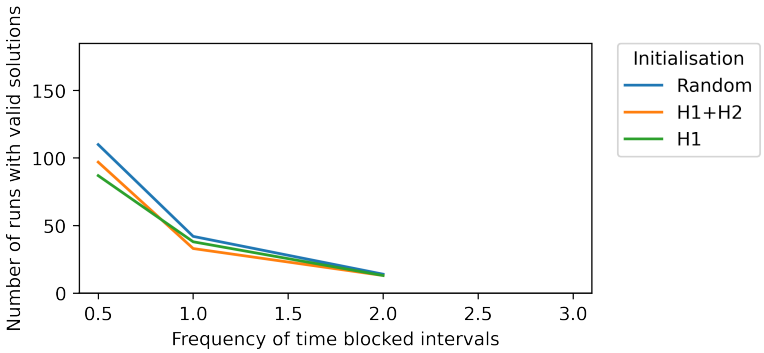


(c) "Grav" network

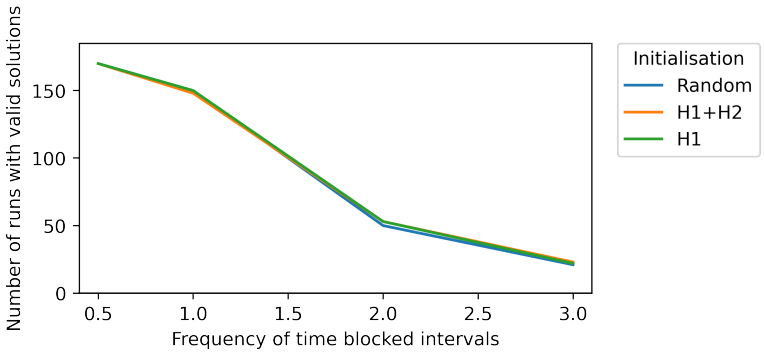


(d) "BA" network

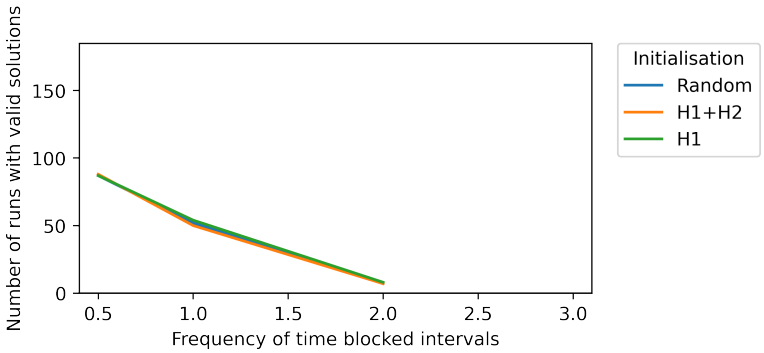
Figure 6.3: Effects of initialisation on number of runs of MARMT producing at least one valid solution for different network types, by time blocked interval length.



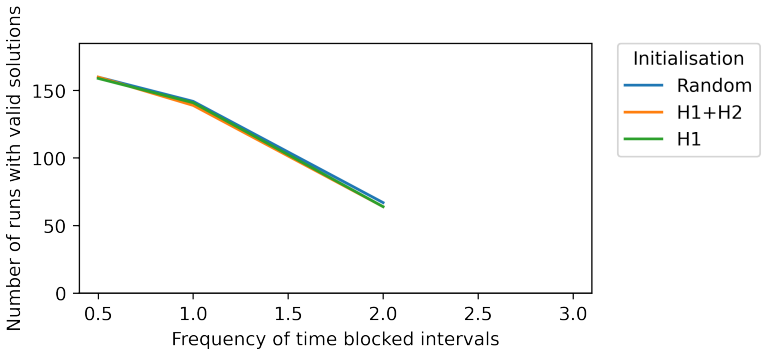
(a) "NM2" network



(b) "Prox" network



(c) "Grav" network



(d) "BA" network

Figure 6.4: Effects of initialisation on number of runs of MARMT producing at least one valid solution for different network types, by time blocked interval frequency.

solution, seen in Table 6.1. This means that generally, the variants finding at least one valid solution most often are also the ones providing the best solution quality on average.

Performance measures cannot be interpreted when there are no solutions found, therefore the set of instances included in the averages are not the same for each variant in Table 6.2. Comparison of the performance measures across only the solved instances might favour variants that solved only the easier instances. The alignment is not perfect, the variant finding at least one solution most often uses random initialisation, while the best solution quality is reached with the H1+H2 initialisation method. To see if this is caused by the easier instances being solved by the H1+H2 initialisation variant, the performance measures of runs not producing any valid solutions is assigned a value twice the highest value of the given performance measure in the dataset. The new average values of the performance measures are shown in Table 6.3 for the two variants in question.

Table 6.2: Average performance measures for the 20 variants, only considering cases where at least one valid solution was found.

Representation	LS rate	Initialisation	ε	R3	RHV
DV-1	0.0	H1+H2	1.6597	0.0711	0.5411
DV-1	0.02	H1+H2	***1.3275***	***0.0235***	***0.3805***
DV-1	0.02	H1	*1.3474*	*0.0292*	*0.4691*
DV-1	0.02	Random	*1.4627*	*0.0339*	*0.4953*
DV-2	0.0	H1+H2	2.0638	0.0929	0.7464
DV-2	0.02	H1+H2	*1.4038*	*0.0292*	0.5262
DV-2	0.02	H1	1.6607	0.0433	0.6072
DV-2	0.02	Random	*1.6489*	*0.0432*	0.5199
DF	0.0	H1+H2	2.2238	0.0906	*0.5045*
DF	0.02	H1+H2	1.8562	0.0519	0.5197
DF	0.02	H1	1.9708	0.0648	0.6362
DF	0.02	Random	2.4948	0.0918	0.7191
IP	0.0	H1+H2	2.2933	0.0994	0.5176
IP	0.02	H1+H2	1.7534	0.0493	0.5639
IP	0.02	H1	1.7986	0.0608	0.6509
IP	0.02	Random	2.4663	0.0993	1.0984
RK	0.0	H1+H2	2.1288	0.0925	0.6698
RK	0.02	H1+H2	1.6896	0.0432	*0.4878*
RK	0.02	H1	2.0191	0.0659	0.7024
RK	0.02	Random	2.2219	0.0794	0.8994

Even with the high values of the performance measures substituted for unsolved instances, the H1+H2 initialisation method is better than the random initialisation according to all three performance measures. Therefore, it can be concluded that there is a trade-off between minimising leaving instances unsolved, and finding high quality solution sets. This makes intuitive sense, if one considers that finding a solution requires exploration, and finding good quality solutions requires exploitation.

In the following, only the best two variants are considered, DV-1 with local search and random or H1+H2 initialisation.

Table 6.3: Average performance measures for the two best variants, the cases where no solutions are found are incorporated with a penalty.

Representation	LS rate	Initialisation	ε	R3	RHV
DV-1	0.02	H1+H2	**19.003**	**0.606**	**0.619**
DV-1	0.02	Random	19.028	0.61	0.689

Table 6.4: Number of runs returning at least one valid solution with each of the 20 considered variants, with a time budget of 30 seconds.

Representation	LS rate	Initialisation	Instances with solutions
DV-1	0.02	H1+H2	4714
DV-1	0.02	Random	**4743**

6.3.2 Smaller time budgets

Experiments were conducted with a time budget of 30 seconds to consider the case of shorter time budgets as opposed to termination based on the number of generation passed without improvement in the Pareto front.

Table 6.4 shows the number of runs producing at least one valid solution. The numbers are close to the ones seen in Table 6.1 with convergence based termination, the decrease in the number of runs with at least one valid solution is less than 3%. Regarding the quality of the solutions, Table 6.5 shows the three performance measures. These are better compared to the values seen in Table 6.2 with convergence based termination. With smaller time budgets, the same pattern can be observed, the random initialisation is better for finding solutions and the H1+H2 initialisation is better for finding good quality solutions, but it leaves more instances unsolved. When high values are substituted for unsolved instances, the difference between the two initialisations becomes small, as can be seen in Table 6.6, and the performance measures are no longer aligned. Therefore it is less obvious if the H1+H2 is superior in solution quality.

Table 6.5: Average performance measures for the 2 best variants, only including cases where at least one valid solution was found, with a time budget of 30 seconds.

Representation	LS rate	Initialisation	ε	R3	RHV
DV-1	0.02	H1+H2	**1.143**	**0.019**	**0.377**
DV-1	0.02	Random	1.186	0.027	0.389

Table 6.6: Average performance measures for the two best variants, cases where no solutions found are incorporated with a penalty, with a time budget of 30 seconds.

Representation	LS rate	Initialisation	ε	R3	RHV
DV-1	0.02	H1+H2	9.704	0.593	**0.629**
DV-1	0.02	Random	**9.653**	**0.593**	0.634

Table 6.7: Average and maximum running times of NAMOA*, in hours.

Parallel arcs	Mean running time	Max running time
1	0.15	11.23
5	0.62	18.08
10	1.12	46.33

6.3.3 Comparing to enumerative approach

This section compares the best variants of MARMT to the NAMOA* algorithm. The NAMOA* algorithm assumes the optimality principle, and therefore does not find all Pareto-optimal solutions for the constrained problem.

Running times

NAMOA* is an enumerative algorithm, that runs until all Pareto-optimal solutions are found (assuming the optimality principle) and therefore its running time is highly variable depending on the instance. NAMOA* It was found in (Machuca Sánchez et al. 2012), that Two days were allowed for solving each instance, to save computational resources. The observed running times are summarised in Table 6.7. The average running time ranges from 10 minutes to more than an hour depending on the number of parallel arcs in the multigraph, which is high compared to the 30 second time budget seen in Section 6.3.2, that produced reasonable results. Even in the case of convergence based termination, the average running time was 125s for the memetic algorithm, while the maximum was 50 minutes, where it was capped. The maximum running times with NAMOA* range from 11 hours to more than two days, although at two days any remaining run was terminated, and therefore it cannot be known how long they would have run otherwise. There were 45 instances for which this happened out of the 792.

Solutions missed by NAMOA*

There are 40 instances for which MARMT has found at least one valid solution, and NAMOA* did not, even though it halted within the allocated time. The distribution of these instances

Table 6.8: Number of cases when NAMOA* failed to find any solution, when some solutions were found by MARMT, by time window parameters.

p_{freq}	p_{len}	Number of cases
0	0	0
0.5	0.1	1
1.0	0.1	8
2.0	0.1	9
3.0	0.1	0
4.0	0.1	0
0	0	0
0.1	0.5	3
0.1	1.0	2
0.1	2.0	5
0.1	3.0	7
0.1	4.0	5

regarding the time window parameters are shown in Table 6.8. It can be observed that generally NAMOA has more problems with higher values of the time window parameters, up to the point where some solutions can be found by the memetic algorithm. For the highest values of the time window parameters the instances might be infeasible. It is worth mentioning that even if some instance is solved by MARMT it does not mean that the solution is consistently found, it could have been found only one run of a single variant. Nevertheless, Table 6.8 demonstrates that NAMOA* is no longer an exact algorithm for problems with time windows, and that the memetic algorithm is able to find solutions that NAMOA* will not find.

In turn, there are only 9 instances for which none of the variants of MARMT find a valid solution, and NAMOA* has found one. This is of course not a fair comparison, given that one instance is solved 10 times by each of the 20 variants considered. Still, given that NAMOA* is not a stochastic algorithm, it does not matter how many times it runs, it will produce the same results, while the memetic algorithm can find solutions for more instances in the unrealistic case of unlimited time and computational resources, and therefore has better potential for future research.

A more fair comparison is comparing the first run of the DV-1 variant with random initialisation with NAMOA*. Then it is found that 42 instances unsolved by the best variant of the memetic algorithm are solved by NAMOA*, and 20 instances are solved by the best variant of the memetic algorithm and not by NAMOA*. However, in this comparison, the running times are unbalanced, and with a significantly shorter time budget than 2 days, the memetic algorithm

Table 6.9: Median values of the performance measures.

Graph type	Parallel arcs	ε	R3	RHV
BA	1	1.000	0.000	0.000
BA	5	1.033	0.001	0.002
BA	10	1.047	0.002	0.003
NM2	1	1.046	0.008	0.054
NM2	5	1.120	0.021	0.051
NM2	10	1.104	0.025	0.041
Grav	1	1.002	0.000	0.000
Grav	5	1.008	0.000	0.001
Grav	10	1.020	0.001	0.003
Prox	1	1.026	0.001	0.010
Prox	5	1.084	0.006	0.013
Prox	10	1.154	0.010	0.027

Table 6.10: Mean values of the performance measures.

Graph type	Parallel arcs	ε	R3	RHV
BA	1	1.257	0.022	0.031
BA	5	1.102	0.018	0.660
BA	10	1.132	0.019	0.643
NM2	1	1.064	0.023	0.100
NM2	5	1.176	0.037	0.088
NM2	10	1.327	0.042	0.267
Grav	1	1.027	0.003	0.669
Grav	5	1.032	0.003	0.218
Grav	10	1.087	0.003	0.021
Prox	1	1.100	0.016	0.743
Prox	5	1.164	0.029	0.115
Prox	10	1.271	0.029	0.586

can be expected to be preferable.

Quality of solutions found

Table 6.9 shows the median values of the performance measures for groups of instances with the 30 s time budget, including only the instances with valid solutions. The median value of the performance measures is good, with worse results for the NM2 networks and higher numbers of parallel arcs. These instance types are also more difficult for the NAMOA* algorithm, as seen in Chapter 4, therefore lower quality solutions can be acceptable. The mean values of the same data is shown in Table 6.10, which is higher in most cases, suggesting a skewed distribution, with bad solution quality for some instances.

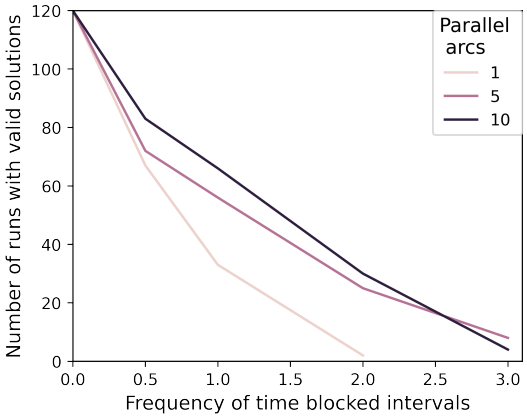
Table 6.11: Instances with at least one valid solution found for different numbers of parallel arcs.

ρ_1	Parallel arcs	Instances with solutions
0.0	1	737
0.0	5	770
0.0	10	809
0.7	1	765
0.7	5	910
0.7	10	880

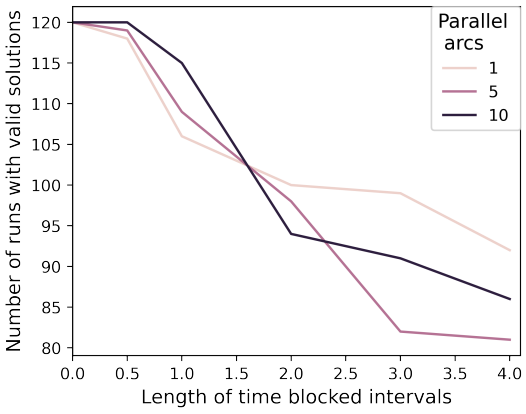
6.3.4 Higher numbers of parallel arcs allow more solutions

The main motivation of including higher numbers of parallel arcs in the instances apart from better representation of the trade-offs between the objectives is the better chances of satisfying time windows. Table 6.11 shows the number of instances with at least one valid solution with convergence based termination broken down based on number of parallel arcs and correlation between objectives. As expected, five parallel arcs lead to more instances with at least one solution for both values of the correlation multiplier than the simple graph case. However, the case of 10 parallel arcs is only better than 5 parallel arcs in the case of no correlation between the objectives. It can also be observed that the case of no correlation between the objectives seems to be harder to find valid solutions for with all three values of the parallel arcs. This is surprising, given that the time windows and the network structure are generated the same way in both cases, and only the time windows should lead to instances being infeasible, even if the cost of the solutions is different. One possible explanation is that when the trade-off between the objectives is less clear, the selection process of the memetic algorithm is less effective.

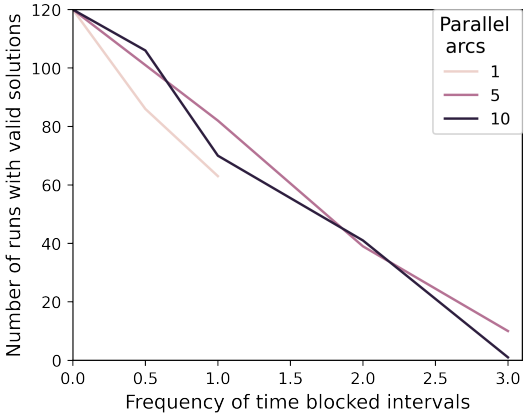
Figure 6.5 shows the number of instances with at least one valid solution for four categories of instances, separated based on the value of the correlation multiplier, and the time window parameters. When there are no time windows all instances are solved independent of the number of parallel arcs. The line associated with single parallel arcs disappears when no valid solutions were found for any of the instances in the category. It can be seen that including more parallel arcs is not always beneficial. For the case of higher lengths of time blocked intervals, using a simple graph seems to be the best. In all other cases, including five parallel arcs is better. Including ten parallel arcs, however, is rarely better than five. A possible explanation for simple graphs being preferable for long time blocked intervals is that more parallel arcs allow only a



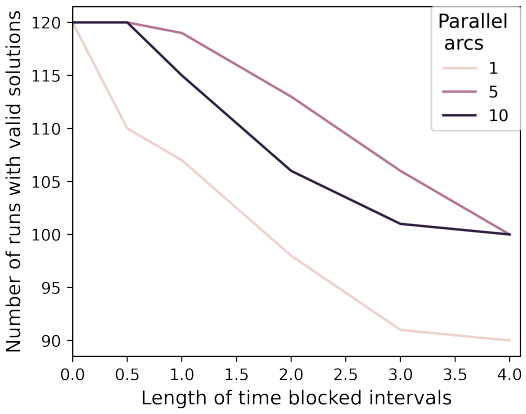
(a) $\rho_1 = 0$, p_{freq} varied



(b) $\rho_1 = 0$, p_{len} varied



(c) $\rho_1 = 0.7$, p_{freq} varied



(d) $\rho_1 = 0.7$, p_{len} varied

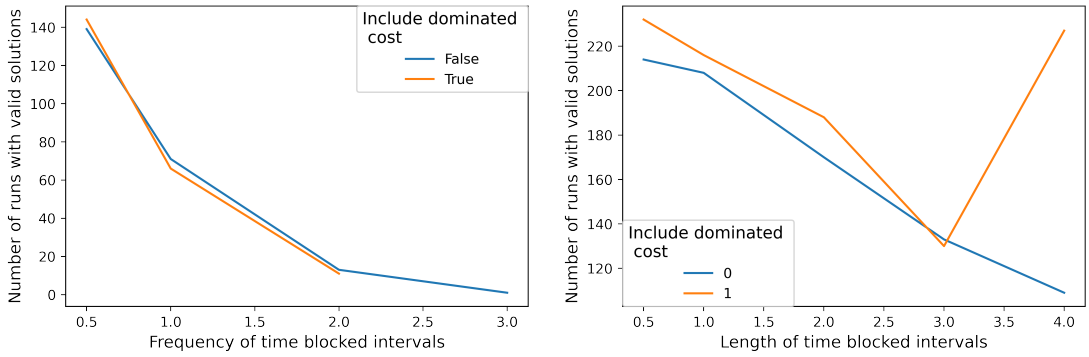
Figure 6.5: Effect of parallel arcs on number of instances with any valid solutions, note that a positive value of ρ_1 implies negative correlation between the objectives (see Section 4.1.1)

small variation in the time of traversing arcs. In the next section dominated arcs with larger variability in traversal times are considered.

6.3.5 Dominated costs

This section examines if including more arcs with dominated cost vectors help with satisfying time windows. It is conjectured that when intermediate holding is not permitted, the arcs with dominated cost vectors offer flexibility and therefore might be useful. To test the hypothesis that dominated costs help with finding valid solutions, the costs of one parallel arc of the instances is modified to be twice its initial value in both objectives. In practice, travel time can often be adjusted by going slower, and this also might serve as a rough model of intermediate holding, where stopping might incur costs in other objectives too, not just time, such as extra fuel for acceleration. This approach is not used for instances with a single parallel arc. It is worth mentioning that dominated costs might be present already with the original instance generation method. However, they are similar in their objective values - most importantly travel time - to the other parallel arcs, and therefore there is only a low level of flexibility. Including a parallel arc with a twice as long traversal time provides more flexibility to satisfy any time windows later on in the path.

The same instances are used with 5 and 10 parallel arcs as before, only the experiments are repeated on the instances modified as explained above with the variant of the DV-1 representation and random initialisation. Figure 6.6 breaks down the number of runs with at least one valid solution found according to the time window parameters and the inclusion of the dominated costs. The inclusion of the arc with the dominated cost leads to more instances being solved in the case of longer time blocked intervals (Figure 6.6b), and not in the case of more frequent time blocked intervals (Figure 6.6a), where the results for including or not including dominated costs is similar. In Figure 6.6b, there is a large increase in the number of solved instances for the highest value of length of time blocked intervals, which is unexplained. In conclusion, doubling the objective values of an arc increases the chance of finding a valid solution on average for instances with longer time blocked intervals with relatively low frequency.



(a) p_{freq} varied **(b)** p_{len} varied

Figure 6.6: Effect of including arcs with dominated costs on number of instances with any valid solutions

6.4 Discussion

In this chapter, the metaheuristic algorithm introduced in Chapter 5 has been adapted to the MSPPMTW, and empirically tested on problem instances generated by the benchmark generator proposed in Chapter 4. The adaptation includes the incorporation of the time window constraints into the fitness evaluation by penalty functions and the proposal of a local search operator based on single objective search. The best variant of MARMT was found to be the direct variable length representation with the local search operator and improved crossover operator. The initialisation method seems less important in the performance, however the highest number of instances were solved with using random initialisation, and the best solution quality was seen with heuristic initialisation.

The best variants of the memetic algorithm produces at least one solution with a similar chance as the NAMOA* algorithm, in a smaller amount of time, however, usually with some compromise in solution quality. For most instances the median of the performance measures are showing a good representation of the reference front, with the average values showing slightly worse quality. It could also be observed that increasing the number of parallel arcs in the multigraph leads to finding at least one solution for more instances, than with lower values. This was the case in all considered values of correlation between objectives and time window parameter values, except for the case of no correlation between objectives and relatively long time blocked intervals.

The benefit of increasing the number of parallel arcs from five to ten is less clear, considering

both proportion of instances solved and average quality of solutions. However, it is possible that better results would be reached with a larger population size. The main disadvantage of including ten parallel arcs over five is expected to be the potential for filling up the population with highly similar candidates, increasing the chances of premature convergence. Increasing the costs of one of the parallel arcs to twice their original value was shown to lead to more instances being solved in the case of relatively long time blocked intervals, but not in the case of short but frequent time blocked intervals. This is not surprising given that there is a better chance of finding a longer time window when the time blocked intervals are concentrated. This chapter has answered some questions that are important in designing models and solution approaches for MSPPMTWs, however further research is needed to support the findings and give answers to questions raised by the results, such as the reason behind the difference made by including dominated costs at the highest value of p_{len}.

Chapter 7

Airport Ground Movement Problem as a Real World Application

7.1 Introduction

In this chapter, the memetic algorithm introduced in Chapter 6, MARMT, is employed to solve the airport ground movement problem (Chen, Weiszer, Locatelli, Ravizza, Atkin, Stewart & Burke 2016), which is a real world application exemplifying the MSPPMTW. As reviewed in 2.5.1, the airport ground movement problem concerns the efficient routing and scheduling of aircraft between gates and runways and vice versa, and can be viewed as a special case of the energy-efficient driving problem. Airport infrastructure is difficult to develop, and thus it is important to be as efficient as possible in utilising resources such as the taxiways. It is also important to balance economic and environmental concerns through careful consideration of energy consumption and emissions. There is a trade-off between taxi time and fuel consumed, as engines are more efficient at certain speeds and on routes with fewer turns (Ravizza et al. 2013). Time windows are imposed on aircraft by other aircraft moving on the airport surface at the same time. Therefore, the goal is to find the *trajectories* for the aircraft that are Pareto-optimal regarding their taxi time and fuel consumption among the trajectories satisfying the time windows. The term trajectory is used to signal the difference from a path. A trajectory defines the movement of a vehicle in space and time, or equivalently it prescribes a speed value for each point of a route.

In the presence of time constraints, apart from affecting the objective values of a trajectory, the choice of speed can affect feasibility of the solution, therefore it is important to manage routing and scheduling in an integrated way. The multigraph representation makes this possible by including the choice of speed profiles as discrete decision variables for certain sections of route. The choice of speed profile is restricted to efficient speed profiles stored in a pre generated database. Therefore, a series of connected arcs in a multigraph can represent a trajectory, describing the movement of the vehicle in terms of time (hence scheduling) and space (hence routing), whereas a route only describes the movement in space. Trajectories are required to be realistic, so that they can be followed precisely, facilitating the efficient operation of the system composed of multiple aircraft at the airport. Realism of the trajectories restricts the level of acceleration and deceleration possible, taking into account the geometry of the route (Weiszer et al. 2020).

With the above considerations, a specialised version of the MSPPMTW is formulated to solve the airport ground movement problem. There are two important differences compared to the general case of the MSPPMTW, (1) the consideration of realistic trajectories, and (2) the time windows are defined for subunits of the arcs of the multigraph used for routing. The memetic algorithm described in the last chapter (MARMT) is employed for solving the airport ground movement problem formulated as an MSPPMTW. The adapted algorithm is called Airport-MARMT. A metaheuristic solution approach is expected to provide multiple benefits for solving the airport ground movement problem, including (1) returning solutions in pre-specified time budgets, (2) handling non-additive costs and (3) potential for global optimisation, including optimising the order of the aircraft. The variants of Airport-MARMT are evaluated using real data about airport operations at the Hong Kong airport (see Figure 7.1). Airport-MARMT is compared to the AMOA* algorithm (Weiszer et al. 2020), a state-of-art solution approach for the airport ground movement problem. The first two points will be shown to be realised in this chapter, while the third point remains to be explored and is outside the scope of this thesis.

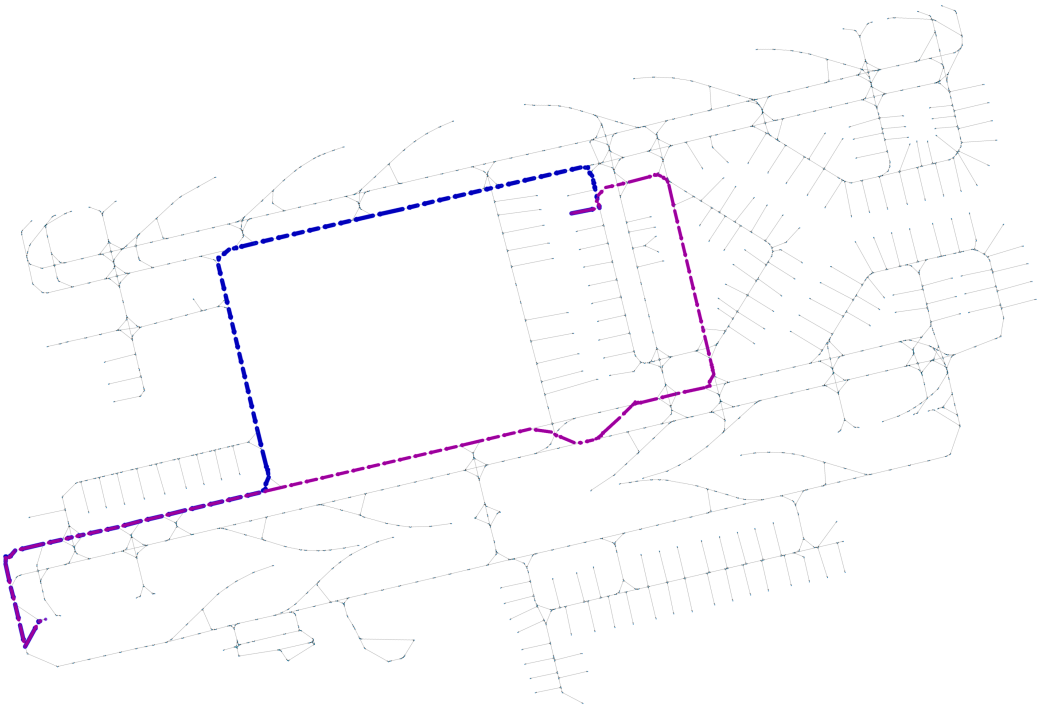


Figure 7.1: Illustration of the layout of the Hong Kong airport with two alternative routes for an aircraft.

7.2 Airport ground movement as a combinatorial optimisation problem

The ground movement problem is decomposed into a series of MSPPMTWs by the framework introduced in (Chen, Weiszer, Locatelli, Ravizza, Atkin, Stewart & Burke 2016). Each aircraft is routed by solving a MSPPMTWs, with respect to constraints inflicted by already routed aircraft.

Optimising speed profiles does not lend itself directly to discrete optimisation techniques, as the decision variables describing a speed profile are real numbers. Realistic speed profiles, however, can be precomputed for certain sections of taxiways based on their geometry, called segments (defined in Section 7.2.3). A specified number of speed profiles and the corresponding cost values can be stored in a database, saving computation time (Weiszer et al. 2015). Trajectories for each aircraft are then found by specifying consecutive segments between the origin node (v_O) and destination node (v_D) and choosing a speed profile for each of them, or equivalently, by specifying a path in the multigraph. The physical constraints regarding the aircraft manoeuvring such as the maximum speed and maximum acceleration rate are handled by the speed-profile

generation algorithm (Weiszer et al. 2015), and are not detailed here.

7.2.1 Sequential routing of aircraft

The sequential routing of aircraft are described in Algorithm 14 (Weiszer et al. 2020). The corresponding notations are explained in Table 7.1. Aircraft are routed on a first come first serve basis. A set of non-dominated solutions Θ_i is found by procedure *Route*. *Route* can be any routing and scheduling algorithm that can handle time window constraints. Here, AMOA* (Weiszer et al. 2020) and the variants of Airport-MARMT (see Section 7.3 are considered. AMOA* is an adaptation of the NAOMA* algorithm for the airport ground movement problem. Intermediate holding during taxiing is not considered in this thesis. Instead, aircraft are held at the gate for 1 min before *Route* is reattempted if there aren't any solutions found.

Algorithm 14 Sequential routing of aircraft

1: Sort *AircraftSequence* according to t_i
2: **for all** $aircraft_i \in AircraftSequence$ **do**
3: $\Theta_i \leftarrow Route(aircraft_i)$
4: **if** Θ_i is empty **then**
5: $t_i \leftarrow t_i + 60$s {1 min postponement}
6: Go to line 3
7: **end if**
8: $\theta \leftarrow$ Preferred solution from Θ_i
9: Reserve route θ and adjust corresponding time constraints
10: **end for**

Even though there is only a single trajectory that will be used by the current aircraft, it is important to find the whole Pareto front or a good representation of it. This way the Decision Maker (DM) can have a good idea about the available trade-offs between the objectives, which facilitates better decision making. The focus in this Chapter is to solve the routing problem for each aircraft efficiently, and not to examine the decision regarding the trajectory chosen from Θ_i. For this reason, a simple strategy is used to simulate the role of a DM, without aiming to be realistic. A single trajectory is chosen out of Θ_i, according to a weighted sum of the objective values and reserved for the current aircraft. The weights of the two objectives are denoted w_1 and w_2, such that $w_2 = 1 - w_1$.

The airport ground movement problem for a given aircraft can be described by two graphs, one depicting the layout of the airport and the other one depicting all possible speed profiles for

Table 7.1: Notations for the current chapter.

Notation	Description
$aircraft_i$	The ith aircraft.
t_i	Start time of $aircraft_i$.
Θ_i	Set of feasible trajectories with non-dominated cost vectors found for $aircraft_i$.
θ	A trajectory.
$\theta(nodes, indices)$	The trajectory defined by *nodes* and *indices*.
w_1, w_2	The weights of the first and second objective when choosing a single trajectory to reserve for $aircraft_i$ from Θ_i.
$pred((v, v_{i+1}), \theta)$	Predecessor edge of segment (v, v_{i+1}) in trajectory θ.
$Route$	The routing procedure that finds trajectories.
$G_0 = (V_0, E)$	Layout graph (simple graph).
$e_i \in E$	An edge in the layout graph.
$\mathscr{F}_e = \{(t_{e,i,start}, t_{e,i,end}) \mid 0 < i < \lvert\mathscr{F}_e\rvert\}$.	Set of time windows assigned to edge e.
$G = (V, A)$	Speed-profile graph (multigraph), $V \subset V_0$.
v_O	Origin node.
v_D	Destination node.
u	Number of considered speed profiles in multigraph reduction.
$(v, w)^k \in A$	The kth arc representing the kth speed profile between nodes v and w in G.
$I(e, (v, w))$	The set of indices k, such that speed profile $(v, w)^k$ is a valid continuation of the route r that has e as its last edge.
$c_{(v,w)^k}$=(c_1, c_2)	Cost vector associated with speed profile $(v, w)^k$.
c_1	Cost component associated with taxi time.
c_2	Cost component associated with fuel consumption.
$C(\theta)$	Sum of cost vectors associated with trajectory θ.
M	Priority based chromosome. For node v, $M_{1,v}$ encodes the priority value and $M_{2,v}$ encodes parallel arc index.

a given aircraft. These two graphs are both relevant for the design of the *Route* procedure, and they are described in detail in the following.

7.2.2 The layout graph

The first important graph in the modelling of the ground movement problem is the *layout graph*, which contains the geographical information about all available taxiways in the airport. The layout graph is a directed graph $G_0 = (V_0, E)$, where the set of nodes V_0 represent gates, stands, runway exits, taxiway intersections and intermediate points. Intermediate points are distributed in such a way that taxiways between two nodes in the layout graph are at most as long as the minimum safe separation of aircraft. This minimum safe separation is set to 60m (Weiszer et al. 2020).

In line with the established terminology for ground movement operations, the sections of taxiways between two nodes in the layout graph are called *edges* $E = \{e_1, e_2, ..., e_{|E|}\}$. To avoid confusion, the term "arc" is used in general when talking about addjacent pairs of nodes in a graph and the term "edge" is reserved for the airport layout graph.

Edges are used for governing the scheduling component of the problem that stems from multiple aircraft moving on the airport ground at the same time. Avoiding conflict between aircraft is ensured by (1) allowing at most one aircraft at a time on each edge and (2) allowing no aircraft on edges that are conflicting with an occupied edge at any time. The set of conflicting edges with edge e consists of each edge e', such that the distance of e and e' is smaller than the minimum safe separation (as measured along taxiways). To keep track of occupation of the edges, a set of time windows are assigned to each edge ($\mathscr{F}_e$). Time windows are corresponding to time intervals when the edge is free, that is, when not occupied and not conflicting with occupied edges.

7.2.3 The speed profile graph

The second important graph in the modelling of ground movement is the *speed profile graph*. Before turning, aircraft generally need to slow down, and after turning accelerate. Thus for speed profile generation, it makes sense to group together sequences of edges depending on their geometry. For this reason, to model speed-profiles, the literature defines *straight* and *turning*

segments (Khadilkar & Balakrishnan 2012, Ravizza et al. 2013). An edge belongs to a turning segment if the angle of the edge with the previous edge in the given route (*predecessor edge*) is above 30 degrees. Otherwise it belongs to a straight segment. Sequential edges that belong to the same type of segment are grouped together. The segments are generated in a way to cover all possible edge-sequences in the layout graph (Weiszer et al. 2015).

The speed profile graph contains information about the pre-computed efficient speed-profiles for all segments. For the same segment multiple alternative speed-profiles are possible. Therefore, the speed profile graph is a multigraph $G = (V,A)$. The nodes of G are the endpoints of segments, $V \subset V_0$, $V = \{1,2,...|V|\}$. Arcs in G are associated with a sequence of edges in G_0. The arcs $(v,w)^k \in A$ are defined by their endpoints $v,w \in V$ and a parallel arc index $1 \leq k \leq |A_{(v,w)}|$. Arcs imply speed profiles, and thus the predecessor edge to the segment (v,w) in a given trajectory θ affects which speed profiles out of $\{(v,w)^1,...(v,w)^{|A_{(v,w)}|}\}$ are available in θ. The set of indices of speed profiles between nodes v and w that can follow a given predecessor edge, e, are denoted $I(e,(v,w))$.

There is a cost vector associated with each speed profile $c_{(v,w)^k} = (c_1,c_2)$, which describes the taxi time (c_1) and fuel consumption (c_2). Speed profiles of the same type (straight or turning) for the same segment can be thought of as a cost matrix, which includes non-dominated cost vectors as its rows.

The number of alternative speed profiles considered for each segment greatly influences the size of the search space. For this reason, multigraph reduction techniques are introduced in (Weiszer et al. 2020) to provide a reduced number of speed profiles from the database. Here, the technique based on including the first u speed profiles is employed. It is important to note that the number of parallel arcs in G sometimes can be different than u for some pairs of nodes. This is because in rare cases two segments might connect the same two nodes, but use different edges. An example of this is shown on Figure 7.2. In this case, there will be u speed profiles for each segment between the same nodes. All of those speed profiles show up in G as parallel arcs, resulting in twice as many parallel arcs as the pre-specified number.

The speed profiles and costs also depend on the weight category of aircraft. Also, if a segment is the first or last one in a trajectory, it implies greater acceleration or declaration than if it is in the middle. This means that speed profile graphs differ for aircraft in different weight

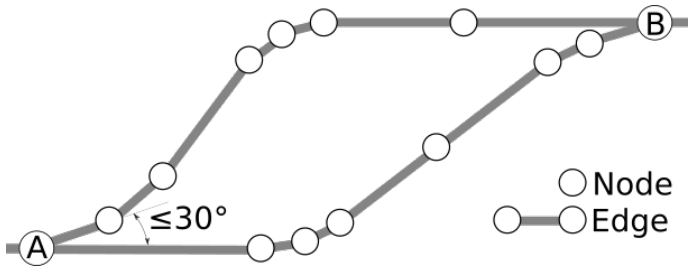


Figure 7.2: The reason behind inhomogeneous numbers of parallel arcs in the speed profile graph. There are two straight segments (angles below 30 degrees) between the same two nodes, *A* and *B*.

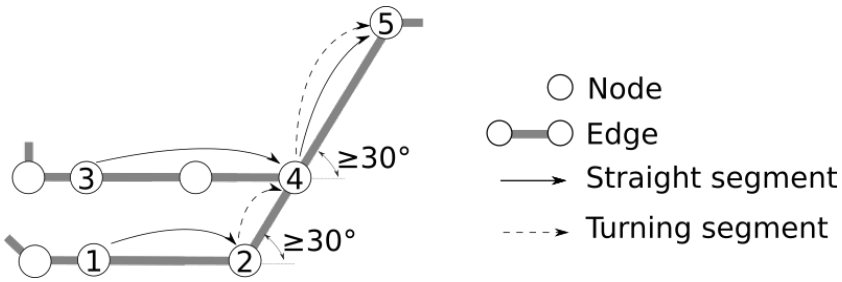


Figure 7.3: Illustration of non-additivity property.

category. However, within the same weight category, the difference is small, and can be quickly modified before routing each aircraft.

7.2.4 Non-additivity of costs

Which speed profiles are available for a given segment depend on the predecessor edge in the given trajectory as discussed above. Straight or turning speed profiles can be associated with the same segment. This characteristic gives rise to the non-additivity property of costs. Labelling approaches for the MSPP only find all possible solutions when the costs satisfy the additivity property, because they eliminate dominated partial solutions. Metaheuristic approaches can easily overcome this challenge.

A detailed example is shown in Figure 7.3. The segment 4-5 is a turning segment, when approached via segment 3-4, and is a straight segment when approached via segment 2-4. Depending on the direction, the cost vector of segment 4-5 is different, a turning segment is more costly. It is possible that up to node 4 the trajectory via node 3 dominates the other trajectory via node 2, and at the same time up to node 5 the trajectory via node 3 no longer dominates the other one.

7.2.5 Routing problem considering a single aircraft

A trajectory for the *i*th aircraft can be specified as a path in the speed profile graph (multigraph). Such a trajectory $\theta \in \Theta_i$ in general has the following form:

$$(v_1, v_2)^{k_1}, (v_2, v_3)^{k_2}, \ldots, (v_{|\theta|-1}, v_{|\theta|})^{k_{|\theta|-1}}, \tag{7.1}$$

$$\text{s.t.} \quad (v_j, v_{j+1})^{k_j} \in A, \quad \forall j \in (1, 2, \ldots, |\theta|-1) \tag{7.2}$$

However, not all paths in the multigraph correspond to a feasible trajectory. A feasible trajectory needs to satisfy the following constraints:

1. Respecting predecessor edges.

$$k_j \in I(e, (v_j, v_{j+1})), \ \forall j \in (1, 2, \ldots, |\theta|-1),$$

   where

$$e = pred((v_j, v_{j+1}), \theta)$$

2. Respecting time windows. For each edge e in trajectory θ, exists a time window $tw \in \mathscr{F}_e$, such that the traversal period of edge e according to θ falls into tw.

3. Not containing any loops in the layout graph. Listing the end nodes of all edges in θ should not contain duplicates.

4. The trajectory should start with the origin node and end with the destination node. $v_O = v_1$ and $v_D = v_{|\theta|}$.

Constraint 1 ensures that the trajectory describes a realistic speed profile in terms of acceleration and deceleration. Constraint 2 ensures the trajectory complies with the available time windows for each edge. Constraint 3 prohibits routes with loops in the layout graph, as a practical consideration. Although loops could be a way of achieving compliance with time windows, holding before taxiing or during taxiing is generally a better choice. Note, that this is a stronger statement than $v_1, v_2, v_3 \ldots, v_{|\theta|-1}, v_{|\theta|}$ being all distinct, which only concerns end nodes of segments. Constraint 4 ensures that the end points of the trajectory are as required.

Constraint 1 and 2 are highly specific to the ground movement problem. In other applications, time constraints might be defined for nodes or for arcs of the multigraph.

The goal is to find the set of feasible trajectories Θ_i with Pareto optimal costs. For a feasible trajectory θ, the corresponding cost vector can be calculated according to Equation (7.3).

$$C(\theta) = \sum_{(v_j,v_{j+1})^k \in \theta} c_{(v_j,v_{j+1})^{k_j}}. \tag{7.3}$$

7.3 The adaptation of the memetic algorithm

This section describes how MARMT (see 6.2) is adapted to the airport ground movement problem. Three representation methods are used, the direct variable length representation and the two priority-based representations. The direct fixed length representation is not included, because evolutionary operators are expected to often lead to invalid offspring. This is because the node specified as the next one for each node in the graph cannot be guaranteed to provide a valid continuation of any trajectory reaching that node. In comparison, in priority based representations the priorities specify an order between the neighbours of any given node. If the neighbour with highest priority is not a valid continuation, the neighbour with second highest priority can be used, and so on.

The variant based on direct variable length representation is implemented in line with the representation method and operators described in Section 7.3.1. Similarly, the description of the variants with priority-based representation schemes are described in Section 7.3.2. The description of the representation methods is repeated in the following at a high level, because of the difference in optimising a path and a trajectory, and to emphasise the role of the two graphs G and G_0.

7.3.1 Direct variable length representation

The direct variable length representation specifies a trajectory by listing node IDs (v) that form a path in the speed profile graph (G) and the corresponding parallel arc indices (k_i) in the following form, that is equivalent to the form described in Section 5.2.1: $[v_1, k_1, v_2, k_2, ...v_{|\theta|}]$.

Decoding a candidate to a trajectory in G is straightforward, with an exception of handling

Constraint 3. To avoid loops in the layout graph, once the decoding reaches a speed profile that includes an edge with an end node already in the decoded part of the trajectory, the decoding is stopped. The already decoded part is returned, which will be penalised for not reaching the destination node in fitness evaluation (see Section 7.3.3). Repair in general would be difficult, because the search process operates at the level of segments (G), while repeated nodes appear at the level of edges (G_0).

The operators from Section 5.2.1 are modified to take into account predecessor edges. In mutation, a node in the candidate path is chosen at random. Then, part of the chromosome is regenerated by a random walk starting from the chosen node, taking predecessor edges into account. The crossover is similar to the process described in Section 6.2.3, except, recombine is sometimes unsuccessful, because of the predecessors. Algorithm 15 is based on Algorithm 12 and includes only a small modification of checking ideal crossing sites until one produces viable offspring (Line 9). If none of the ideal crossing sites produced offspring satisfying Constraint 1, other crossing sites are considered.

Unlike in the previous chapter, a repair mechanism aimed at eliminating loops is not enough to ensure feasibility of the offspring in the ground movement problem, for the reason mentioned earlier, as Constraint (1) can still be violated. Therefore, when a potential crossing site is found in Algorithm 15, the next step is to execute the modified one-point crossover according to Algorithm 16 to remove any loops from the offspring and check for violation of Constraint 1. If a violation occurs, the part up to the violation site is returned as a new candidate (Line 6 of Algorithm 16), which will be penalised in fitness assignment accordingly. Note, that in applications without Constraint 1, removing loops without further penalising is sufficient.

7.3.2 Search based on priority based representations

In priority based representations, the trajectories are encoded indirectly as a priority value assigned to each node in the speed profile graph and a parallel indicator, as before. The algorithm used for the benchmark problem instances can be adopted with very little modification, which illustrates the advantage of using generic representations as opposed to ones closely specialised for the problem. For the adaptation to the airport ground movement problem, the only modification needed is taking predecessor nodes into account in the decoding process, as

Algorithm 15 CrossoverOutline(parent1, parent2)

Input: P_1 := parent1, P_2 := parent2
Output: $ch1, ch2$:= Offspring

if node sequences of P_1, P_2 are identical **then**
  $site \leftarrow$ randomly chosen node from P_1
  $ch1, ch2 \leftarrow$ Recombine($P_1, P_2, site$)
else
  $sites \leftarrow$ crossing sites
  $idealSites \leftarrow$ ideal crossing sites out of $sites$
  for all $site$ in $idealSites$ **do**
    $ch1, ch2 \leftarrow$ Recombine($P_1, P_2, site$)
    if at least one child is feasible **then**
      return $ch1, ch2$
    end if
  end for
  if there is only one crossing site **then**
    $site \leftarrow$ crossing site
    $ch1, ch2 \leftarrow$ Recombine($P_1, P_2, site$)
  else if there are two crossing sites **then**
    $site \leftarrow$ crossing site closer to v_D
    $ch1, ch2 \leftarrow$ Recombine($P_1, P_2, site$)
  else
    $ch1, ch2 \leftarrow P_1, P_2$
  end if
end if
return $ch1, ch2$

Algorithm 16 Recombine(*parent1, parent2, site*)

Input: P_1 := parent1, P_2 := parent2, $site$
Output: $ch1, ch2$:= Offspring

$ch1 \leftarrow P_1$ up to $site$, and P_2 from $site$
$ch2 \leftarrow P_2$ up to $site$, and P_1 from $site$
Remove loops from $ch1, ch2$ {regarding speed profile graph}
for all speed profile in ch_i for $i = 1, 2$ **do**
  if speed profile violates predecessor edges **then**
    Remove nodes from the ch_i starting from the end node of the speed profile
  end if
end for
return $ch1, ch2$

detailed in the following.

The decoding process iteratively finds the neighbour with the highest priority among the ones satisfying Constraint 1 and 3, and adds them to the decoded trajectory. The process is detailed in Algorithm 17. The loop in Lines 4-15 first identifies the *allowed* neighbour list (Line 6) that consists of the nodes that are (1) directly reachable from the last node of the already decoded part of the trajectory, (2) do not introduce loops in G_0 and (3) satisfy predecessor edges. If there are no such nodes, the already decoded part is returned (Line 8). Otherwise, the node with the highest priority is identified, and the lists *nodes* and *indices* defining the trajectory are updated (Lines 12 and 15).

Algorithm 17 DecodingPriorityBased(M)

Input: $M :=$ priority based chromosome
Output: $nodes, indices$
1: $nodes \leftarrow$ list with a single element: v_O
2: $indices \leftarrow$ empty list
3: $predEdge \leftarrow$ None
4: **while** Last element of $nodes \neq v_D$ **do**
5: $neighbours \leftarrow$ Set of nodes reachable from last element in *nodes* in G
6: $allowed \leftarrow$ Set of nodes in $neighbours \notin nodes$, that comply with the $predEdge$, and do not introduce loops in G_0
7: **if** $allowed = \emptyset$ **then**
8: **return** $nodes, indices$
9: **else**
10: $v_{prev} \leftarrow$ Last element of *nodes*
11: $v_{next} \leftarrow$ Node with maximum priority in $allowed$ according to M
12: $nodes = nodes \cup v_{next}$
13: $x \leftarrow \mid I(predEdge, v_{prev}, v_{next}) \mid$
14: $currentIndex \leftarrow \lfloor M_{2,v_{prev}} * x \rfloor + 1$
15: $indices = indices \cup currentIndex$
16: $predEdge = pred((v_{prev}, v_{next}), \theta(nodes, indices))$
17: **end if**
18: **end while**
19: **return** $nodes, indices$

As seen in Section 7.2.3, $|I(e,(v,w)|$ depends on the nodes v and w and the predecessor edge. Therefore, it would be difficult to refer to parallel arcs based on parallel arc indices directly, because the genetic operators would lead to infeasible offspring. This further motivates the argument for the use of parallel arc indicators for the priority based representations put forward in Section 5.2.1.

The other notable difference compared to the last chapter regards the local search. Local

search is conducted as described in Section 6.2.2. Afterwards, the candidate is converted back to priority-based representations using Algorithm 18, a modified version of 10. The modification is needed to ensure that predecessor edges are taken into consideration.

Algorithm 18 directToPriority($nodes, indices$)

Input: $nodes, indices$
Output: $M :=$ priority based chromosome
1: $n \leftarrow$ number of nodes in G
2: $priority \leftarrow n$
3: **for** $i \leftarrow 1$ to $\mid nodes \mid$ **do**
4: $\quad M_{1,nodes[i]} \leftarrow priority$
5: $\quad predEdge \leftarrow$ predecessor edge for segment $nodes[i], nodes[i+1]$
6: $\quad M_{2,nodes[i]} \leftarrow \frac{indices[i]}{|I(predEdge,(nodes[i],nodes[i+1]))|}$
7: $\quad priority \leftarrow priority - 1$
8: **end for**
9: **for** $j \leftarrow 1$ to n **do**
10: $\quad$ **if** $M_{1,j}$ is not yet specified **then**
11: $\quad\quad M_{1,j} \leftarrow priority$
12: $\quad\quad M_{2,j} \leftarrow$ Random floating-point number $\in [0,1]$
13: $\quad\quad priority \leftarrow priority - 1$
14: $\quad$ **end if**
15: **end for**
16: **return** M

Algorithm 18 takes the node sequence (*nodes*) and the index sequence (*indices*) as input, together defining a trajectory. It returns a priority-based chromosome that encodes the same trajectory. For this, it needs to specify the value of each element in the 2 by n matrix, M, that is the priority-based chromosome. In Lines 3-8, the priorities of the nodes that appear in the trajectory are set. These priorities are increasing from the destination node towards the origin node. This ensures that in the decoding process, the neighbour with the highest priority is the next node in the trajectory for each node, as all other unvisited nodes in the graph have lower priorities. In Lines 9-13, the rest of the genes are filled up with the lower priority values, and random parallel indices. Converting to random keys representation is done along the same lines. The only difference is that the priority values are floating point numbers.

7.3.3 Fitness function and constraint handling

The fitness assignment for candidates encoding trajectories, including calculation of penalties is done along the same lines as laid out in Section 6.2.1 for solving benchmark instances. The

algorithm is restated with notations of a trajectory in Algorithm 19. The objective values of trajectory θ is calculated according to Equation (7.3) (Line 1). To get the fitness value of θ, the penalties need to be added for violation of Constraint 2 and 4. The other two constraints will not be violated by any candidates that reach fitness evaluation. The maximum value of any cost component in any speed profiles in G, *maxCost* is used as a basis to establish the magnitude of the penalties (Line 2). For not reaching the destination node, the level of violation is measured as the minimum distance of the trajectory and the destination node (Line 3). For violating time windows, the level of violation is measured as the number of time-windows violated (Line 4). The level of constraint violation and *maxCost* are multiplied to give the base penalty, p_0.

Algorithm 19 FitnessAssignment(θ)

Input: θ := trajectory
Output: $fitness$:= fitness value
1: $fitness \leftarrow C(\theta)$
2: $maxCost \leftarrow$ Maximum value of any cost component in G.
3: $minHop \leftarrow$ Hopcount between θ and v_D
4: $conflicts \leftarrow$ The number of time window violations
5: **if** $v_D \neq v_{|\theta|}$ **then**
6: $\quad p_0 \leftarrow maxCost * minHop$
7: $\quad fitness \leftarrow fitness + (p_0 * \alpha_1, p_0 * \alpha_2)$
8: **end if**
9: **if** $conflicts > 0$ **then**
10: $\quad p_0 \leftarrow maxCost * conflicts$
11: $\quad fitness \leftarrow fitness + (p_0 * \alpha_3, p_0 * \alpha_4)$
12: **end if**
13: **return** $fitness$

As in Section 6.2.1, four weights are set up respectively for the two objectives and two constraints, $\alpha_1, \alpha_2, \alpha_3, \alpha_4$. These weights for the penalty function have been tuned by irace (López-Ibáñez et al. 2016), and their values are set to be $1, 7, 5, 3$ respectively. The weights are applied in Lines 7 and 11 to calculate the penalty for each objective. One possible advantage of the values of this weight set-up is that the first objective value is penalised more for violating time windows and the second for not reaching the destination. Therefore, the population can be expected to not be biased towards any of the two constraints.

Table 7.2: Variants of Airport-MARMT with different representations.

Representation	Description	Abbr.	Operators
Direct	Node IDs listed with directly encoded parallel arc indices	DV	Modified one point crossover, Random walk mutation, Local search
Integer priority	Integer priority values with indirectly encoded parallel arc indices	IP	WMX, Insertion mutation, Local search
Random keys	Floating point priority values with indirectly encoded parallel arc indices	RK	Two point crossover, Insertion mutation, Local search

Table 7.3: Tuned values of the parameters.

Variants	Pop. size	Cross. rate	Mut. rate	τ_{max}	l_{min}	l_{rel}
DV	120	0.90	0.19	4.5	3	0.8
IP	120	0.95	0.29	4.5	3	0.8
RK	120	0.83	0.13	4.5	3	0.8

7.4 Implementation details

The variants of Airport-MARMT considered in this chapter are described in Table 7.2. The H1 method for heuristic initialisation is used, as described in section 5.2.3.

7.4.1 Parameters

For tuning the parameters of the algorithms, the irace package (López-Ibáñez et al. 2016) was used. The description of irace and how it was used can be found in Section 5.3. The minimum and maximum length of candidate trajectories for local search was set to $l_{min} = 3$ and $l_{rel} = 80\%$ in all experiments, as tuned by irace. The value of τ_{max} controlling the randomisation of the initial population is also the same in all experiments, so that all variants start with approximately the same quality of initial population. Tuning was carried out respectively for the different representations for values of crossover and mutation rates. Population size is kept the same across all variants, in order to ensure that the same local search rate will lead to approximately equal number of local search to be performed per generation. The tuned parameters are shown in Table 7.3. The value of the local search rate is examined in Section 7.5.

7.4.2 Problem instance

All algorithms are tested using the real data of a day of operations at the Hong Kong International Airport (7.1.2017, 0:00–24:00), that includes 506 aircraft. This taxiway layout can be categorised as medium complexity, with 1309 nodes, 1491 edges, 160 gates and 38 runway exits. In this study, 506 aircraft will be routed sequentially, as described in Algorithm 14, with time windows inflicted on later aircraft due to already routed aircraft.

7.4.3 Evaluation of solutions

The most straightforward way of comparison is the overall travel time and fuel consumption realised for the whole day of operation. This is an important practical measure of efficiency for longer intervals of airport operations. Apart from the overall taxi time, it is also important to report how often there were no solutions found and a postponement by one minute was necessary until a solution becomes available. For this reason, the *adjusted taxi time* is used to account for the total postponements. For trajectory θ, the adjusted taxi time $C_{1,\theta,adj}$ in seconds can be calculated from the taxi time of the trajectory ($C_{1,\theta}$), and the number of postponements P for the given aircraft according to Equation (7.4).

$$C_{1,\theta,adj} = C_{1,\theta} + 60 * P \tag{7.4}$$

In most cases the fuel consumption and adjusted taxi time is reported in an aggregated way, as this aggregation represents the real operational cost of airport after a decision is made by the air traffic controller. Therefore, the weights (w_1, w_2) for choosing the reserved trajectory from the Pareto front for each aircraft are used here as the surrogates of the operational cost coefficients. Note that using any other weights could skew the results. The *weighted aggregate* ($C_{aggr,\theta}$) of a trajectory θ is calculated according to Equation (7.5).

$$C_{aggr,\theta} = C_{1,\theta,adj} * w_1 + C_{2,\theta,adj} * w_2 \tag{7.5}$$

To also show comparison to AMOA* in a concise way, *relative weighted aggregate* is also introduced. The relative weighted aggregate characterises how the variants of Airport-

MARMT perform compared to AMOA* regarding the reserved trajectories. The relative weighted aggregate for the ith aircraft is calculated according to Equation (7.6), from the weighted aggregate of the trajectory reserved by Airport-MARMT ($C_{aggr,\theta_{i,MARMT}}$) and by AMOA* ($C_{aggr,\theta_{i,AMOA*}}$).

$$C_{aggr,i,rel} = \frac{C_{aggr,\theta_{i,MARMT}}}{C_{aggr,\theta_{i,AMOA*}}} \tag{7.6}$$

Apart from the reserved trajectories, finding a close approximation of the real Pareto front for each aircraft is also of interest. The ε indicator is one of the most popular quality indicators for assessing proximity to the real Pareto front (Liefooghe & Derbel 2016). It signals higher qualities of the approximate front by lower values. When the approximated front is the same as the reference Pareto front, the value of the ε indicator equals 1. The unimpeded Pareto fronts - obtained without considering time windows - are used as reference fronts. This way, for the same aircraft, the same and best possible reference front is used regardless of the current strategy for choosing the reserved trajectory from the Pareto front.

Another relevant metric is the size of the Pareto front. It is preferred to have more and uniformly distributed solutions (Chitra & Subbaraj 2012), so that the trade-offs between the objectives can be assessed by air traffic controllers. Also, when more solutions are found there is a better chance for finding solutions that do not violate time windows. However, it is generally easier to find many low-quality solutions than many high-quality ones, and thus the ε metric is more important.

7.5 Results

The methods are implemented in Python 3, for the evolutionary computation, the inspyred package (Tonda 2020) was used. All numerical tests are performed on Queen Mary's Apocrita HPC facility (Z n.d.*b*). Parallelisation has not been utilised.

First, the results obtained by the state-of-the-art enumerative solution approach are described, which is used as a basis for comparison. Then, results of Airport-MARMT are presented with a focus on the effect of local search. Two different termination criteria are explored for Airport-MARMT: (1) 10 generations without change in the Pareto optimal solutions found so far, to evaluate convergence properties and (2) 10 seconds time budget, to evaluate the potential use

Table 7.4: Computational times of AMOA* (u=3) routing a single aircraft.

w_1	mean [s]	median [s]	min [s]	max [s]	std [s]	sum [h]
1	80.1	41.4	0.9	602.1	106.3	11.26
0.5	45.3	22.9	0.4	338.8	61.6	6.37
0	40.9	20.1	0.4	322.8	56.1	5.74

Table 7.5: Number of solutions found by AMOA* (u=3) for a single aircraft.

w_1	mean	median	min	max	std
1	13.78	9	1	91	14.04
0.5	13.89	9	1	91	14.15
0	13.86	9	1	91	14.13

for real-time decision support.

7.5.1 Results based on the enumerative solution approach

AMOA* with $u = 3$ is used to route all 506 aircraft, because that is the highest number of speed profiles per segment that can be solved in a reasonable time. Table 7.4 describes the distribution of the running times for all aircraft. The running times of AMOA* range from 0.4 seconds to 602 seconds. Higher running times are observed when the fastest trajectory ($w_1 = 1$) is reserved for each aircraft than in the other two cases. The mean of the running times is approximately twice of the median, showing a skewed distribution with most aircraft being routed in shorter times, while a smaller number of them taking significantly longer. The average running time is much higher than 10 seconds, which is the limit acceptable for on-line decision support (Z n.d.*a*).

Table 7.5 describes the number of Pareto optimal solutions found by AMOA*. The distribution of the number of solutions is very similar in all three cases. This is also a skewed distribution, the average number of solutions found is 13 and the maximum is 91.

7.5.2 Results based on convergence based termination

First, the case when the algorithm is allowed to run until convergence is considered. Convergence is assumed when there is no improvement in the Pareto front found so far for 10 consecutive generations. For the purpose of comparing to AMOA*, $u = 3$ is used and $u = 10$ is also included to show how Airport-MARMT scales to higher numbers of parallel arcs.

Quality of reserved trajectories

In Table 7.6 results are shown for relative weighted aggregate of objectives for the whole day of operation. In almost all cases the direct representation outperforms the priority based ones, and random key representation is the worst of the three. There are only a few cases where the statistical significance of the difference between MARMT-IP and MARMT-D cannot be established. Therefore, it can be concluded that the direct representation performs the best in terms of relative weighted aggregate, when the computational time is not limited.

Table 7.6 also shows that with the local search rate value of 0.02, MARMT-D is able to reach the same or slightly better results as AMOA*, when it is used with the same strategy for reserving routes for individual aircraft. This is possible, because of the non-additivity property and the presence of time windows.

The increasing of the local search rate brings decreasing marginal improvement, and at the same time increases computational time. Figure 7.4 shows the relationship between local search rate, relative weighted aggregate and computational time. There is a sharp improvement in solution quality until the local search rate reaches 0.02. The sum of computational time with the local search rate of 0.02 is 6.13 hours for the whole day of operations with $u = 3$ and 7.29 hours with $u = 10$. This is close to the computational times observed with AMOA*, that is between 5.6 hours and 11.2 hours for $u = 3$. With the local search rate of 0.1, the computational time is above 21 hours, but there is only a modest improvement in relative weighted aggregate over the local search rate of 0.02. Note, that the improvement upon the highest previous local search rate is statistically significant with $p = 0.005$ until the local search rate 0.06 for $u = 3$ and until the local search rate 0.04 for $u = 10$.

Regarding the number of parallel arcs, the relative weighted aggregate is lower for all local search rates with $u = 10$, at least with MARMT-D, as can be seen in Table 7.6. The same trend

Table 7.6: Relative weighted aggregate of objectives for sequential routing of the 506 aircraft with different local search rates and different numbers of parallel arcs.

		$u = 3$			$u = 10$		
	local s. r.	RK	D	IP	RK	D	IP
$w = 0.0$	0	1.0267 **	**1.0231**	1.0265 **	1.0163 **	**1.0083**	1.0163 **
	0.001	1.0226 **	**1.0160**	1.0193 **	1.0133 **	**0.9998**	1.0098 **
	0.005	1.0136 **	**1.0029**	1.0083 **	1.0042 **	**0.9883**	0.9960 **
	0.01	1.0092 **	**1.0002**	1.0026 **	0.9985 **	**0.9850**	0.9900 **
	0.02	1.0055 **	**0.9983**	0.9993 **	0.9957 **	**0.9834**	0.9867 **
	0.04	1.0033 **	**0.9979**	0.9983 *	0.9932 **	**0.9831**	0.9846 **
	0.06	1.0023 **	**0.9977**	0.9981 **	0.9915 **	**0.9829**	0.9842 **
	0.08	1.0019 **	**0.9978**	0.998	0.9916 **	**0.9828**	0.9836 **
	0.1	1.0017 **	**0.9977**	0.9978	0.9906 **	**0.9829**	0.9836 **
$w = 0.5$	0	1.0207 *	**1.0180**	1.0198 *	1.0207 **	**1.0148**	1.0204 **
	0.001	1.0145 **	**1.0117**	1.0127	1.0140 **	**1.0075**	1.0116 **
	0.005	1.0054 **	**1.0013**	1.0032 **	1.0059 **	**0.9988**	1.0021 **
	0.01	1.0026 **	**0.9991**	1.0000 **	1.0027 **	**0.9967**	0.9988 **
	0.02	1.0004 **	**0.9982**	0.9986 **	1.0007 **	**0.9961**	0.9969 **
	0.04	0.9995 **	**0.9979**	0.9982 *	0.9992 **	**0.9959**	0.9962
	0.06	0.9994 **	**0.9978**	0.9981 *	0.9988 **	**0.9960**	0.9961
	0.08	0.9992 **	**0.9978**	0.9982 **	0.9983 **	**0.9959**	**0.9959**
	0.1	0.9990 **	**0.9977**	0.9980 **	0.9984 **	**0.9958**	0.9959
$w = 1.0$	0	1.0194 **	**1.0174**	1.0188 *	1.0210 **	**1.0161**	1.0196 **
	0.001	1.0128 *	1.0112	**1.0111**	1.0149 **	**1.0105**	1.0132 *
	0.005	1.0056 **	1.0034	**1.0031**	1.0073 **	**1.0024**	1.0046 **
	0.01	1.0025 **	**1.0007**	1.0016	1.0045 **	**1.0000**	1.0019 **
	0.02	1.0012 **	**0.9998**	1.0001 *	1.0031 **	**0.9993**	1.0001 **
	0.04	1.0005 **	**0.9994**	0.9996	1.0022 **	**0.9988**	0.999
	0.06	1.0002 **	**0.9992**	0.9994 *	1.0015 **	**0.9988**	0.9988
	0.08	1.0004 **	**0.9992**	0.9993	1.0014 **	**0.9987**	0.9988
	0.1	1.0003 **	**0.9991**	0.9994 *	1.0013 **	**0.9987**	0.9988

[a] Stopping criteria is 10 generations without improvement.

[b] The one-sided Wilcoxon signed rank test was used to decide statistical significance between the best result (bold) and the others in each sub-row. Results are indicated as (*) : $p < 0.05$, (**): $p < 0.005$, (***): $p < 0.0005$.

[c] RK: MARMT_RK with Random key representation.

[d] D: MARMT_D with Direct representation.

[e] IP: MARMT_IP with Integer priority representation.

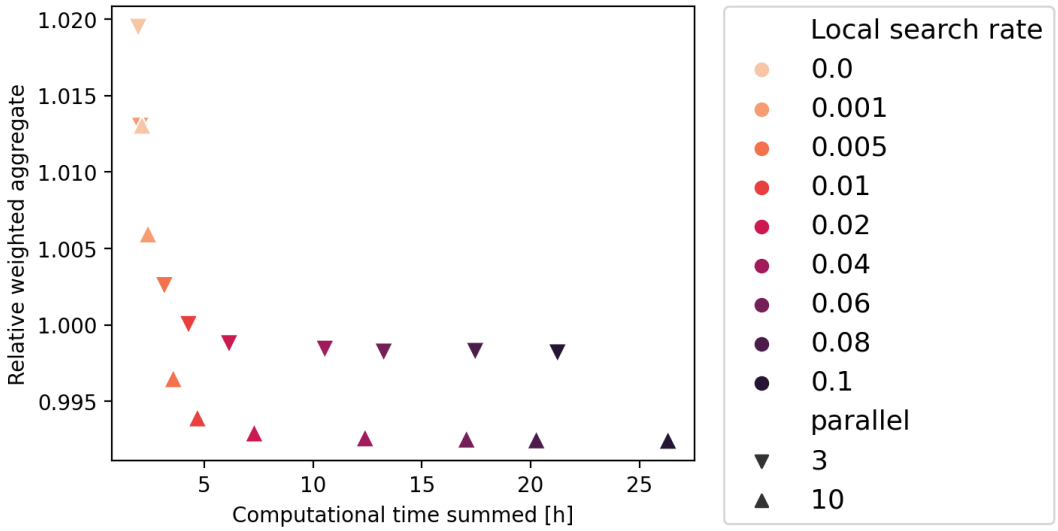


Figure 7.4: Relationship between solution quality as measured by the relative weighted aggregate of objectives, local search rate, sum of computational time, and number of speed profiles for the 506 aircraft, with the direct representation. Experiments with different w_1 values are grouped together.

is shown in Figure 7.4, additionally, it can be observed that the additional computational time required is low, especially compared to AMOA*. For $u = 10$, AMOA* would take longer than 10 days (Weiszer et al. 2020).

The results shown so far only consider relative weighted aggregate of objectives. Figure 7.5 shows the two objectives separately for AMOA* and Airport-MARMT with different representations and the local search rates 0 and 0.02. It can be observed that MARMT-D dominates the results achieved by AMOA* with a local search rate of 0.02.

Quality of Pareto fronts found

To quantify the quality of the Pareto front found by the variants of Airport-MARMT for the individual aircraft, results regarding the number of solutions found and the ε indicator are presented.

In Table 7.7 results regarding the ε indicator are presented. MARMT-D performs the best again among the three regardless of how often local search is performed. The statistical significance of this is stronger in the case of $u = 10$. In the case of $u = 3$, MARMT-IP can also be competitive. This is especially the case when $w_1 = 1$, that is when the fastest trajectories are reserved for each aircraft.

Table 7.7: Mean ε indicator for sequential routing of the 506 aircraft with different local search rates and different numbers of parallel arcs.

		$u = 3$			$u = 10$		
	local s. r.	RK	D	IP	RK	D	IP
$w_1 = 0.0$	0.000	1.0348 *	**1.0325**	1.0351 *	1.0344 **	**1.0291**	1.0344 **
	0.001	1.0324 **	**1.0269**	1.0291 *	1.0325 **	**1.0222**	1.0294 **
	0.005	1.0236 **	**1.0155**	1.0188 **	1.0249 **	**1.0137**	1.0185 **
	0.010	1.0198 **	**1.0127**	1.0138 *	1.0197 **	**1.0103**	1.0140 **
	0.020	1.0162 **	**1.0105**	1.0109	1.0180 **	**1.0087**	1.0117 **
	0.040	1.0145 **	**1.0098**	1.0100	1.0163 **	**1.0082**	1.0101 **
	0.060	1.0135 **	**1.0095**	1.0098 *	1.0146 **	**1.0081**	1.0095 **
	0.080	1.0134 **	**1.0097**	**1.0097**	1.0150 **	**1.0079**	1.0090 **
	0.100	1.0130 **	**1.0094**	**1.0094**	1.0139 **	**1.0079**	1.0089 **
$w_1 = 0.5$	0.000	1.0341 *	**1.0327**	1.0343	1.0334 **	**1.0279**	1.0343 **
	0.001	1.0315 **	**1.0262**	1.0286 *	1.0293 **	**1.0206**	1.0264 **
	0.005	1.0213 **	**1.0147**	1.0165 *	1.0213 **	**1.0111**	1.0159 **
	0.010	1.0174 **	**1.0108**	1.0120 *	1.0175 **	**1.0079**	1.0110 **
	0.020	1.0144 **	**1.0090**	1.0094 *	1.0151 **	**1.0066**	1.0088 **
	0.040	1.0129 **	**1.0082**	1.0087 **	1.0128 **	**1.0062**	1.0071 **
	0.060	1.0122 **	**1.0081**	1.0084 *	1.0122 **	**1.0062**	1.0068 **
	0.080	1.0119 **	**1.0080**	1.0085 **	1.0115 **	**1.0061**	1.0065 **
	0.100	1.0117 **	**1.0079**	1.0082 *	1.0117 **	**1.0060**	1.0064 **
$w_1 = 1.0$	0.000	1.0345 **	**1.0320**	1.0346 **	1.0331 **	**1.0281**	1.0325 **
	0.001	1.0311 **	**1.0252**	1.0280 **	1.0287 **	**1.0223**	1.0271 *
	0.005	1.0213 **	**1.0153**	1.0158	1.0208 **	**1.0120**	1.0156 **
	0.010	1.0176 **	**1.0109**	1.0123 *	1.0172 **	**1.0087**	1.0112
	0.020	1.0150 **	**1.0093**	1.0100 **	1.0148 **	**1.0073**	1.0085 **
	0.040	1.0131 **	**1.0086**	1.0089 *	1.0125 **	**1.0065**	1.0070 **
	0.060	1.0122 **	**1.0084**	1.0085	1.0116 **	**1.0063**	1.0067 *
	0.080	1.0124 **	**1.0082**	1.0084 *	1.0114 **	**1.0062**	1.0066 *
	0.100	1.0121 **	**1.0082**	1.0084	1.0112 **	**1.0063**	1.0066

[a] Stopping criteria is 10 generations without improvement.
[b] The one-sided Wilcoxon signed rank test was used to decide statistical significance between the best result (bold) and the others in each sub-row. Results are indicated as (*) : $p < 0.05$, (**): $p < 0.005$, (***): $p < 0.0005$.
[c] RK: MARMT_RK with Random key representation.
[d] D: MARMT_D with Direct representation.
[e] IP: MARMT_IP with Integer priority representation.

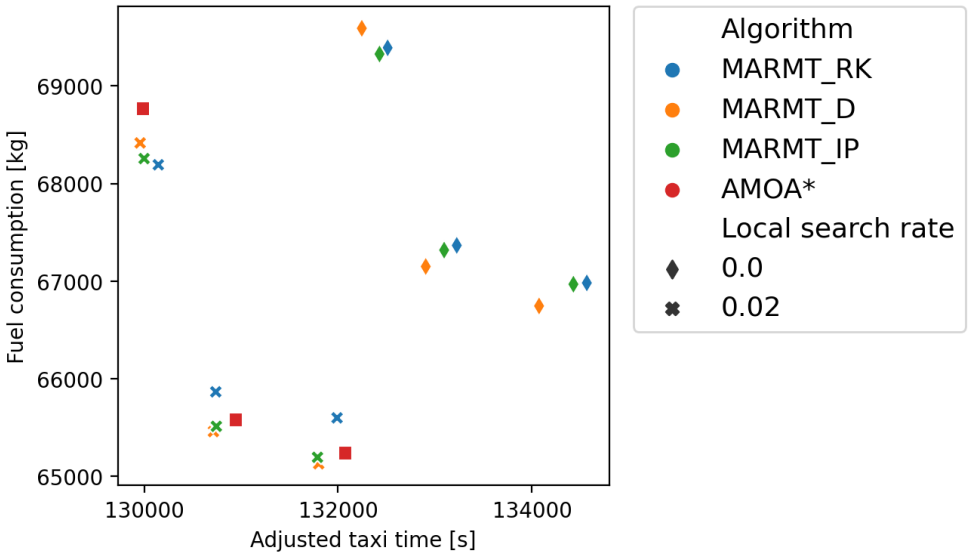


Figure 7.5: Difference between Airport-MARMT with and without local search and AMOA*. Three different weights are used for reserving trajectories for individual aircraft. Each marker represents the average of 10 data points.

Table 7.8 shows the average number of Pareto optimal solutions obtained. This metric also supports the superiority of MARMT-D. With the increasing local search rate the number of Pareto optimal solutions increases, with the exception that the variants with the lowest local search rates produce less solutions than the ones without local search. In the case of $u = 3$, the number of solutions found remains under the value set by AMOA*, which is between 13 and 14 for all strategies for reserving a single trajectory. In the case of $u = 10$, the number of solutions obtained with the direct representation is nearly twice as high as with $u = 3$.

So far it has been shown that when the computational time of Airport-MARMT is not limited, the direct representation performs the best in terms of overall objective values, the ε indicator and the number of Pareto optimal solutions. Higher local search rates generally lead to better solution quality. It could also be observed that better solution quality can be reached with more parallel arcs (alternative speed profiles) in the multigraph.

These results are important as they establish that Airport-MARMT is able to find Pareto fronts close to or better than those of AMOA*, when given enough time.

Table 7.8: Mean number of Pareto optimal solutions found for the 506 aircraft with different local search rates and different numbers of parallel arcs. Stopping criteria is 10 generations without improvement.

		$w_1 = 0.0$			$w_1 = 0.5$			$w_1 = 1.0$		
		RK	D	IP	RK	D	IP	RK	D	IP
	local search r.									
$u = 3$	0.000	5.4	**9.9**	5.0	5.4	**9.8**	5.0	5.4	**9.9**	5.0
	0.001	5.2	**9.3**	4.9	5.2	**9.3**	4.9	5.2	**9.3**	4.9
	0.005	5.1	**9.5**	5.3	5.1	**9.4**	5.3	5.1	**9.5**	5.4
	0.010	5.3	**10.2**	5.9	5.3	**10.3**	6.0	5.3	**10.4**	6.0
	0.020	5.6	**11.1**	6.9	5.7	**11.3**	7.0	5.7	**11.2**	7.0
	0.040	6.2	**11.9**	7.9	6.2	**11.9**	7.9	6.3	**11.9**	7.9
	0.060	6.5	**12.1**	8.4	6.6	**12.2**	8.3	6.6	**12.3**	8.4
	0.080	6.7	**12.4**	8.6	6.6	**12.4**	8.6	6.7	**12.5**	8.7
	0.100	6.7	**12.5**	8.8	6.7	**12.5**	8.8	6.8	**12.6**	8.8
$u = 10$	0.000	6.7	**19.4**	6.1	6.8	**19.5**	6.1	6.8	**19.7**	6.2
	0.001	6.5	**17.9**	5.9	6.5	**18.1**	6.0	6.6	**18.0**	6.0
	0.005	6.4	**17.9**	6.5	6.4	**18.0**	6.5	6.5	**18.1**	6.6
	0.010	6.7	**18.9**	7.4	6.7	**19.2**	7.5	6.7	**19.4**	7.4
	0.020	7.2	**20.4**	8.7	7.2	**20.8**	8.8	7.3	**20.9**	8.9
	0.040	8.1	**21.8**	10.2	8.1	**22.3**	10.3	8.2	**22.1**	10.3
	0.060	8.4	**22.5**	11.1	8.5	**22.6**	11.3	8.5	**22.7**	11.3
	0.080	8.6	**22.7**	11.7	8.7	**23.1**	11.8	8.7	**23.0**	11.8
	0.100	8.6	**22.9**	12.2	8.6	**23.2**	12.2	8.7	**23.3**	12.2

[a] RK: MARMT_RK with Random key representation.
[b] D: MARMT_D with Direct representation.
[c] IP: MARMT_IP with Integer priority representation.

Table 7.9: Mean relative weighted aggregate for the 506 aircraft with the 10 seconds time budget. Different local search rates and different numbers of parallel arcs are included for the direct representation.

	local search rate								
	0.0	0.001	0.005	0.01	0.02	0.04	0.06	0.08	0.1
$u = 3$									
$w_1 = 0.0$	1.0239	1.0194	1.0126	1.0094	1.0096	1.0103	1.0096	1.0095	**1.0091**
$w_1 = 0.5$	1.0182	1.0139	1.0060	1.0039	**1.0031**	1.0034	1.0033	**1.0031**	1.0040
$w_1 = 1.0$	1.0180	1.0131	1.0065	1.0049	**1.0034**	1.0038	1.0040	1.0044	1.0040
$u = 10$									
$w_1 = 0.0$	1.0096	1.0051	0.9974	0.9965	**0.9948**	0.9966	0.9975	0.9992	0.9978
$w_1 = 0.5$	1.0155	1.0110	1.0043	1.0026	**1.0025**	**1.0025**	1.0035	1.0030	1.0034
$w_1 = 1.0$	1.0163	1.0124	1.0062	1.0054	**1.0045**	**1.0045**	1.0054	1.0058	1.0047

7.5.3 Potential for real time decision support

Promptness of routing decisions is important in real world transportation problems. In the airport ground movement problem, a trajectory needs to be found for each aircraft under 10 seconds for on-line decision support (Z n.d.*a*). This time budget is used as the termination criteria in the following experiments. The 10 seconds time budget amounts to 1.4 hours for routing the 506 aircraft, which is lower than the computational times observed in Figure 7.4, even without local search. Therefore, compromised solution quality can be expected. Only results of MARMT-D are shown, without loss of generality.

Table 7.9 shows the relative weighted aggregate with the 10 seconds time budget and different local search rates. With increasing local search rates, statistical significance of marginal improvement in relative weighted aggregate can be shown until the value of 0.02, for both $u = 3$ and $u = 10$. Even with the 10 seconds time budget, the relative weighted aggregate is within 1% of the relative weighted aggregate obtained by AMOA*.

The average number of Pareto optimal solutions found is shown in Table 7.10. The number of solutions decrease as the local search rate increases, which is expected as local search is computationally intensive and thus less evaluations can be performed in the same time budget. However, the lower local search rates below 0.02 lead to poorer objective values, which are the primary consideration. With the local search rate of 0.02, the average number of optimal solutions found is approximately half of the the number of optimal solutions found without limiting the computation time.

It is not obvious from our experiments if including more speed profiles is worthwhile with

Table 7.10: Mean number of Pareto optimal solutions for the 506 aircraft with the 10 seconds time budget. Different local search rates and different numbers of parallel arcs are included for the direct representation.

	local search rate								
	0.0	0.001	0.005	0.01	0.02	0.04	0.06	0.08	0.1
$u = 3$									
$w_1 = 0.0$	9.0	8.7	7.6	7.2	6.5	5.9	5.5	5.3	5.3
$w_1 = 0.5$	8.9	8.6	7.7	7.3	6.7	6.0	5.6	5.4	5.2
$w_1 = 1.0$	9.0	8.7	7.7	7.3	6.6	6.1	5.6	5.3	5.2
$u = 10$									
$w_1 = 0.0$	16.2	15.4	12.7	11.3	9.7	8.3	7.3	6.7	6.6
$w_1 = 0.5$	16.1	15.1	12.6	11.3	9.7	8.2	7.2	6.9	6.6
$w_1 = 1.0$	16.4	15.2	12.8	11.2	9.7	8.5	7.2	6.9	6.8

small time budgets. The relative weighted aggregate is similar in the cases of $u = 3$ and $u = 10$. However, there are more solutions found in the case of $u = 10$.

7.6 Discussion

In this chapter the metaheuristic solution approach proposed in the earlier chapters of this thesis has been applied for the airport ground movement problem, as a representative of transportation problems where the multigraph representation is relevant. The adaptation included modifying existing operators for the specific problem, namely searching for realistic trajectories and not just paths. The variants of Airport-MARMT were evaluated using real data about one day of operation at Hong Kong International airport.

Three genetic representation schemes were compared including the direct, the integer priority based and the random key representations. Based on the converged solution quality, the direct variable length representation proved to be the most effective at exploring the search space. The difference was the most striking regarding the number of Pareto optimal solutions found. This is likely due to (1) the direct representation being the only one that provides 1-to-1 encoding and (2) the crossover operator exploring variations of parallel arcs even when no novel node sequences are found. This finding is in line with the findings of Chapter 6, where also the direct variable length representation was found to be the most effective.

The performance of the best variant of Airport-MARMT was found to be very close to a state-of-the-art enumerative solution approach even with a time budget as short as 10 seconds. Furthermore, on average outperformed the said enumerative approach when allowed to converge,

which can be attributes to the fact that the principle of optimality does not hold for the airport problem, because of the non-additivity of costs and time windows. Note that the running time of the enumerative approach was 40 seconds on average. The results put forward in this chapter support the possibility of using the proposed algorithm in a reactive way, where trajectories are recomputed in a short time after disturbances occur (Weiszer et al. 2020).

The local search operator proposed in the last chapter enhances the search capability of Airport-MARMT by introducing new high quality candidates into the population based on single objective search. Great improvements in solution quality were observed with the inclusion of the local search operator for the case of convergence based termination. The improvements were seen in terms of objective values, number of solutions found and the ε quality indicator. In the case of the 10 seconds time budget as the stopping criteria, objective values improved with the inclusion of local search. However, the number of Pareto optimal solutions decreased with the increasing local search rate, which can be explained by local search taking up more time.

It was found that including more speed profiles slightly improves solution quality with convergence termination in terms of all measures considered for evaluation. The same consistent conclusion could not be drawn for the 10 seconds time budget, even though the number of Pareto optimal solutions found increases with higher number of parallel arcs. Section 6.3.4 showed that the benefit of higher numbers of parallel arcs is dependent on the properties of the time windows and the correlation of objectives. Also, there it is possible that there is an optimal value of speed profiles to be included. Possibly, the flexibility allowed by more speed profiles cannot be fully capitalised, because the order of aircraft is fixed beforehand, as explained in (Weiszer et al. 2021). These questions remains to be explored in future research.

Chapter 8

Conclusion and Future Directions

8.1 Summary

This thesis investigated the Multi-objective Shortest Path Problem on Multigraphs with Time Windows. The investigation covers the increased computational burden posed by the multigraph formulation, a benchmark set proposed for the specific problem, and the proposal of a meta-heuristic solution approach thoroughly tested on real and benchmark problem instances. The main achievements are summarised as follows.

Computational time analysis The most important features behind the high computational effort are identified through a thorough empirical study and a predictive model. A large and diverse set of artificial problem instances have been solved by the NAMOA* algorithm adapted to multigraphs. Properties of the instances were quantified and used as features in the predictive model. Through feature selection, it was found that features with the largest effect on computational effort of Multigraph MSPP instances are (1) hopcount between origin and destination node, (2) number of parallel arcs in the multi-arcs of the multigraph, (3) correlation of the third objective to the first, (4) closeness centrality of the origin node. These are followed by other features describing the structure and costs of the instance. The predictive power of the proposed model is high, supporting the significant role of the selected features. The multigraph formulation clearly increases the computational burden, as highlighted by the number of parallel arcs being high on the list of important features. The relevance of the hopcount for the multigraph

problem is also not surprising, given the number of possible ways to traverse H multi-arcs, with each including u parallel arcs is u^H. Reasonable agreement was observed between the features selected for predicting computational effort and number of Pareto-optimal solutions, which supports the robustness of the findings against the specific choice of the algorithm used. The thesis made a modest contribution to the long standing research problem of generating instances for the MSPP with high numbers of Pareto-optimal solutions, by identifying important features for predicting the number of Pareto-optimal solutions. Among these, multiple features based on the *range of cost directions* were included in the ten most important features, with higher importance than the correlation of objectives. This is a novel proposal of the thesis, as previously the correlation of objectives was the only property related to costs that was known to affect problem difficulty. The two features are not independent, however, the *range of cost directions* might capture information that is more important.

Benchmark for the MSPPMTW A benchmark generation framework was proposed to offer a standard test set for the MSPPMTW, a problem relevant for many real-world applications. The generation technique is able to provide a wide variety of problem instances with different structural and cost properties, shown to affect computational effort by the predictive model. The stochastic generation approach makes it possible to create high number of instances with similar properties, which might be useful in future research for identifying additional features affecting problem difficulty. The temporal aspect of integrated routing and scheduling problems is incorporated into the instances by nominating the first objective to represent travel time. Travel time is an important objective in all transportation related problems, and it is necessary to incorporate it into the instances to be able to represent time windows. A method for assigning time windows to arcs is included in the generator, where the properties of the time blocked intervals (between time windows) can be influenced by two parameters. These parameters control the number and the length of the blocked intervals.

Experiments on the proposed benchmark set Experiments were conducted on an extensive MSPPMTW instance set generated by the proposed approach, to determine the efficiency of the NAMOA* algorithm and the proposed memetic algorithm on instances with different properties. It was found that the structure, objective correlation values, and the time window parameters

have non-trivial effects on the number of instances solved. For example, in the case of relatively long blocked intervals and no correlation between the objectives including higher numbers of parallel arcs does not increase the number of instances with at least one solution found. At the same time, higher number of parallel arcs are beneficial in the case of shorter blocked intervals and also in the case of any values of p_{len} when there is negative correlation between the objectives. Including cost vectors that are twice as expensive as other ones among parallel arcs between the same nodes was also investigated. Improvements were observed in terms of number of instances solved in the case of longer time blocked intervals, but not in the case of short, but more frequent blocked intervals. This finding suggests that (1) including dominated costs can be beneficial depending on the properties of the time windows and (2) intermediate holding can expected to be beneficial in some cases, even if it incurs large costs.

Memetic algorithm proposed A memetic algorithm (MARMT) is developed for the multigraph MSPP with time windows. Three different genetic representation methods are adapted to the multigraph problem and compared to each other, and to NAMOA*. The direct variable length representation is found to perform the best in almost all experiments. A local search operator is proposed based on single objective search. Integrating the local search operator to the metaheuristic significantly improves solution quality as measured by multiple quality indicators. Heuristic initialisation is observed to improve solution quality, especially for the fixed length representations. Even when local search is applied, heuristic initialisation still improves solution quality as measured by performance measures. However, a slight drop in the number of solved instances is observed, which suggests premature convergence. MARMT is shown to handle the higher number of parallel arcs better compared to the enumerative approach in terms of running times. MARMT can also find solutions that are missed by enumerative approaches that rely on the principle of optimality, which does not hold for the MSPPMTW. MARMT is further tailored to a representative real-world application, the airport ground movement problem, where it is shown to be a good candidate for on-line routing, thanks to its good compromise between solution quality and running time.

8.2 Outlook on future work

A number of future directions are still open for research in relation to the work presented in this thesis, which are summarised below.

The predictive model for the computational effort might be improved by considering features that describe the graph structure and the costs in a more detailed way. The current work only includes a handful of features that characterise the costs in the network, which consist of hundreds of cost vectors. This is justified, because all the cost vectors are generated in a similar way. However, cost vectors in real-world networks might not be well characterised by a few features. Perhaps all cost vectors and all arcs of the underlying graph can be used as inputs to a predictive model, this might require a deep learning approach.

The problem of generating difficult instances for both the MSPP and MSPPMTW is far from being exhausted. Due to the rarity of extreme outlier instances in terms of running times and number of solutions, it is hard to obtain difficult instances to investigate. Future work could explicitly look for difficult instances, for example by evolving them through the framework put forward in (Smith-Miles & Bowly 2015). Evolving graphs can be informed by the process in (Barry et al. 2015). If more difficult instances could be obtained in larger quantities, it would be easier to identify what makes them special.

There are many possible ways that the proposed benchmark generator can be used or improved. The most obvious one is designing better algorithms for the MSPPMTW, through testing them on the proposed benchmark set. The literature lacks an exact approach that can find all Pareto-optimal solutions for the problem (without assuming the principle of optimality) or approximate methods that can provide good solutions in short time budgets. The memetic algorithm proposed in the thesis aims to serve the latter purpose, although there are many other promising metaheuristic algorithms that can be applied to the same problem. Improvements of the benchmark generation method are possible by including more ways to generate costs and time windows, perhaps influenced by needs of particular real world applications.

There is a high interest in many-objective shortest path problems, driven by the more realistic and detailed modelling of routing problems. A recent benchmark suite is provided in (Weise & Mostaghim 2021) for simple graph problems. The multigraph modelling approach investigated

in the thesis can be readily extended to include many objectives and the proposed solution approaches pave the way to solve such problems effectively.

The improvement of operators might become a fruitful area of further research for priority based representations. Often, the changes introduced in the chromosome are not sufficient to modify the encoded solution. This limits the exploration capabilities of these algorithms and leads to slow convergence. Strategies aimed at ensuring the modification of the encoded solution in genetic operators might mitigate some of the disadvantage due to the the ambiguity associated with the priority based representations. One recent method for reducing the search space with random keys representation was proposed in (Krömer et al. 2021). Such methods might also help by shrinking the parts of the search space that correspond to the same solution, thereby making a change in the encoded solution more probable.

The work presented in this thesis establishes the value of metaheuristic solution methods for this family of problems. Using different metaheuristic algorithms, in particular Ant Colony Optimisation would also be worthwhile to investigate, because it is often used for graph problems (Glabowski et al. 2012, Ghoseiri & Nadjari 2010, Hassan et al. 2020). The benchmark problem instances introduced in this thesis provide a basis for any future comparison of different algorithms.

The study of the airport ground movement problem is based on a medium sized airport and includes the objectives of taxi time and fuel consumption. The proposed algorithm shows great potential for reaching real-time decision support. A natural extension of this work is to investigate larger airports, scenarios with denser traffic and considering emissions as the third objective. Another interesting future direction is including the order of the aircraft in the optimisation problem (Brownlee et al. 2018). Given that the routes available to different aircraft depend on the previously routed aircraft, this is a much more complex problem than the MSPPMTW, making metaheuristic algorithms uniquely suitable.

Appendix

Supplementary material for Chapter 5

Table 1: Illustration of the role of the heuristic initialisation methods in the solution quality achieved by variants of NSGA-II using different genetic representations and crossover operators, described by the R3 metric.

	Initialisation			
	R	H1	H1+H2	H2+R
ε indicator				
DirFixL	*** 7.4730	*** 2.7997	**1.1934**	1.2363
DV-dir	*** 3.1227	*** 2.9739	*** 1.2048	**1.2000**
DV-indir	3.4750	3.1787	1.2263	**1.2225**
IP-PX	*** 3.2724	*** 2.6662	**1.2909**	*** 1.3045
IP-WMX	*** 2.5375	*** 2.1416	**1.2738**	*** 1.2787
RK-2ptX	*** 3.4757	*** 2.1760	**1.1765**	*** 1.1853
RK-arithX	*** 3.9342	*** 2.7162	**1.2371**	*** 1.2621
RK-uniX	*** 3.9197	*** 2.3856	**1.2358**	*** 1.2511
R3 indicator				
DF	*** 0.5846	*** 0.2904	**0.0236**	*** 0.0281
DV-dir	*** 0.3485	*** 0.3323	*** 0.0239	**0.0231**
DV-indir	0.3825	0.3495	0.0268	**0.0262**
IP-PX	** 0.3548	** 0.2861	**0.0389**	** 0.0406
IP-WMX	*** 0.2778	*** 0.2192	**0.0364**	*** 0.0376
RK-2ptX	*** 0.3822	*** 0.2171	**0.0202**	*** 0.0212
RK-arithX	*** 0.4504	*** 0.3077	**0.0342**	*** 0.0380
RK-uniX	*** 0.4230	*** 0.2481	**0.0315**	*** 0.0340
RHV indicator				
DF	*** 0.5353	*** 0.1633	**0.0124**	*** 0.0163
DV-dir	*** 0.2111	*** 0.1957	*** 0.0129	**0.0124**
DV-indir	0.2415	0.2111	**0.0148**	**0.0148**
IP-PX	*** 0.2238	*** 0.1632	**0.0232**	*** 0.0254
IP-WMX	*** 0.1525	*** 0.1095	**0.0208**	*** 0.0217
RK-2ptX	*** 0.2418	*** 0.1065	**0.0107**	*** 0.0113
RK-arithX	*** 0.2998	*** 0.1634	**0.0181**	*** 0.0210
RK-uniX	*** 0.2840	*** 0.1264	**0.0174**	*** 0.0201

[a] Lower values correspond to better quality.

[b] The average values are calculated from 50 runs for each of the 32 instances.

[c] The time budget for NSGA-II is specified as 20s.

[d] The one-sided Wilcoxon signed rank test was used to decide statistical significance between H1+H2 and H2+R . Results are indicated as (*) : $p<0.05$, (**): $p<0.005$, (***): $p<0.0005$.

Table 2: Comparing variants of NSGA-II with their best initialisation methods for both the 10s and 20s time budgets. Average solution qualities on the 32 instances are shown according to the three quality indicators.

	ε indicator		$R3$ indicator		RHV indicator	
Representation	10 s	20 s	10 s	20 s	10 s	20 s
DirFixL	*** 1.2041	*** 1.1934	*** 0.0255	*** 0.0236	*** 0.0135	*** 0.0124
DV-dir	*** 1.2090	*** 1.2000	*** 0.0246	*** 0.0231	*** 0.0137	*** 0.0124
DV-indir	*** 1.2334	*** 1.2225	*** 0.0283	*** 0.0262	*** 0.0160	*** 0.0148
IP-PX	*** 1.3228	*** 1.2909	*** 0.0439	*** 0.0389	*** 0.0269	*** 0.0232
IP-WMX	*** 1.2897	*** 1.2738	*** 0.0394	*** 0.0364	*** 0.0227	*** 0.0208
RK-2ptX	**1.1915**	**1.1765**	**0.0231**	**0.0202**	**0.0124**	**0.0107**
RK-arithX	*** 1.2458	*** 1.2371	*** 0.0359	*** 0.0342	*** 0.0192	*** 0.0181
RK-uniX	*** 1.2547	*** 1.2358	*** 0.0346	*** 0.0315	*** 0.0196	*** 0.0174

[a] Lower values correspond to better quality.

[b] The average values are calculated from 50 runs for each of the 32 instances.

[c] The one-sided Wilcoxon signed rank test was used to decide statistical significance between each variant compared to the best one in each row. Results are indicated as (*) : $p<0.05$, (**): $p<0.005$, (***): $p<0.0005$.

Table 3: Comparing the best variants of NSGA-II with a time budget of 20s for the four representations, separately for the 32 test instances. The values of the average RHV quality indicator of 50 runs with each representation is shown.

Problem Instance					DV-dir	DF	RK-2ptX	IP-WMX	NAMOA* 20s
obj	graph t.	l_{max}	n	ρ					
2	grid	5	100	-0.75	*** 0.0093	*** 0.0070	**0.0053**	*** 0.0109	-
				0.00	*** 0.0101	*** 0.0092	**0.0050**	*** 0.0133	-
			196	-0.75	*** 0.0102	* 0.0064	**0.0059**	*** 0.0120	-
				0.00	*** 0.0097	*** 0.0123	**0.0062**	*** 0.0202	-
		10	100	-0.75	*** 0.0068	0.0056	**0.0054**	*** 0.0112	-
				0.00	* 0.0056	*** 0.0077	**0.0050**	*** 0.0159	-
			196	-0.75	*** 0.0060	*** 0.0055	**0.0046**	*** 0.0101	-
				0.00	**0.0082**	*** 0.0137	*** 0.0114	*** 0.0185	-
	waxman	5	100	-0.75	*** 0.0013	**0.0008**	0.0011	* **0.0008**	0.0000
				0.00	*** 0.0062	*** 0.0046	**0.0023**	**0.0023**	0.0000
			196	-0.75	*** 0.0083	*** 0.0033	**0.0024**	*** 0.0074	0.2918
				0.00	*** 0.0045	**0.0018**	*** 0.0030	*** 0.0098	0.1730
		10	100	-0.75	** 0.0024	** 0.0026	**0.0022**	* 0.0023	0.2328
				0.00	*** 0.0094	*** 0.0045	**0.0034**	*** 0.0100	0.0000
			196	-0.75	*** 0.0061	*** 0.0034	**0.0028**	*** 0.0056	-
				0.00	*** 0.0117	*** 0.0092	**0.0067**	*** 0.0116	-
3	grid	5	100	-0.75	**0.0069**	*** 0.0144	*** 0.0124	*** 0.0194	-
				0.00	**0.0204**	*** 0.0239	*** 0.0229	*** 0.0543	-
			196	-0.75	**0.0108**	*** 0.0165	*** 0.0156	*** 0.0234	-
				0.00	**0.0205**	*** 0.0425	*** 0.0316	*** 0.0637	-
		10	100	-0.75	0.0125	0.0128	**0.0124**	*** 0.0192	-
				0.00	**0.0157**	*** 0.0348	*** 0.0351	*** 0.0601	-
			196	-0.75	**0.0062**	*** 0.0176	*** 0.0173	*** 0.0252	-
				0.00	**0.0165**	*** 0.0259	*** 0.0274	*** 0.0707	-
	waxman	5	100	-0.75	*** 0.0162	*** 0.0121	**0.0084**	*** 0.0098	-
				0.00	*** 0.0094	0.0069	**0.0067**	*** 0.0143	0.4047
			196	-0.75	*** 0.0122	*** 0.0077	**0.0049**	*** 0.0118	0.5902
				0.00	*** 0.0344	*** 0.0158	**0.0120**	*** 0.0219	0.2895
		10	100	-0.75	**0.0052**	*** 0.0105	*** 0.0077	*** 0.0106	0.5829
				0.00	** 0.0165	**0.0151**	*** 0.0190	*** 0.0258	0.3898
			196	-0.75	*** 0.0376	*** 0.0176	**0.0127**	*** 0.0259	-
				0.00	*** 0.0403	** 0.0269	**0.0238**	*** 0.0462	-

[a] The results of NAMOA* regarding the RHV quality indicator with the same time budget of 20s are shown in the last column, for comparison.

[b] The best values among the representations are shown in bold in each row.

[c] The one-sided Wilcoxon signed rank test was used to decide statistical significance between each variant compared to the best one in each row. Results are indicated as (*) : p<0.05, (**): p<0.005, (***): p<0.0005.

Table 4: Comparing the best variants of NSGA-II for the four representations, separately for the 32 test instances with a time budget of 10s. The values of the average Epsilon quality indicator of 50 runs with each representation is shown.

Problem Instance					DV-dir	DF	RK-2ptX	IP-WMX	NAMOA* 10s
obj	graph t.	l_{max}	n	ρ					
2	grid	5	100	-0.75	*** 1.0654	**1.0450**	1.0452	*** 1.0638	-
				0.00	*** 1.1896	*** 1.2170	**1.1509**	*** 1.3325	-
			196	-0.75	*** 1.1080	* 1.0571	**1.0527**	*** 1.0735	-
				0.00	*** 1.2094	*** 1.3056	**1.1666**	*** 1.3539	-
		10	100	-0.75	*** 1.0680	1.0551	**1.0532**	*** 1.0849	-
				0.00	**1.1630**	*** 1.2795	*** 1.2115	*** 1.3453	-
			196	-0.75	**1.0476**	*** 1.0533	*** 1.0523	*** 1.0749	-
				0.00	**1.3301**	*** 1.7687	*** 1.6615	*** 1.7445	-
	waxman	5	100	-0.75	*** 1.0634	**1.0320**	*** 1.0466	1.0339	1.0000
				0.00	*** 1.1912	*** 1.1465	1.0939	**1.0934**	2.3609
			196	-0.75	*** 1.1301	**1.0536**	** 1.0595	*** 1.0987	4.7292
				0.00	*** 1.1884	**1.1003**	*** 1.1601	*** 1.2524	5.1258
		10	100	-0.75	*** 1.1088	**1.1058**	** 1.1075	*** 1.1071	4.2204
				0.00	*** 1.4441	**1.3008**	1.3205	*** 1.5825	1.0000
			196	-0.75	*** 1.0818	1.0554	**1.0533**	*** 1.0698	-
				0.00	*** 1.2413	* 1.1597	**1.1504**	*** 1.2407	-
3	grid	5	100	-0.75	**1.0476**	*** 1.0596	*** 1.0599	*** 1.0778	-
				0.00	**1.2156**	*** 1.2329	1.2189	*** 1.3434	-
			196	-0.75	**1.0781**	* 1.0810	* 1.0798	*** 1.1374	-
				0.00	**1.2009**	*** 1.3451	*** 1.2689	*** 1.3512	-
		10	100	-0.75	**1.1489**	1.1598	*** 1.2072	*** 1.2144	-
				0.00	**1.2249**	*** 1.3985	*** 1.3746	*** 1.6090	-
			196	-0.75	**1.0541**	*** 1.0989	*** 1.1010	*** 1.1653	-
				0.00	**1.2516**	*** 1.4188	*** 1.3974	*** 1.6710	-
	waxman	5	100	-0.75	*** 1.2318	**1.2007**	*** 1.2124	*** 1.2142	-
				0.00	*** 1.2017	**1.1554**	1.1567	*** 1.2294	-
			196	-0.75	*** 1.1151	*** 1.0546	**1.0458**	*** 1.0704	3.1893
				0.00	*** 1.4183	1.2640	**1.2527**	*** 1.3235	4.7317
		10	100	-0.75	1.0675	1.0700	**1.0673**	*** 1.0797	3.7121
				0.00	**1.5491**	1.5555	* 1.6199	*** 1.7931	15.2834
			196	-0.75	*** 1.3641	* 1.1619	**1.1292**	*** 1.2875	-
				0.00	*** 1.8879	**1.5401**	1.5517	*** 2.1514	-

[a] The results of NAMOA* regarding the Epsilon quality indicator with the same time budget of 10s are shown in the last column, for comparison.

[b] The best values among the representations are shown in bold in each row.

[c] The one-sided Wilcoxon signed rank test was used to decide statistical significance between each variant compared to the best one in each row. Results are indicated as (*) : p<0.05, (**): p<0.005, (***): p<0.0005.

Table 5: Comparing the best variants of NSGA-II for the four representations, separately for the 32 test instances with a time budget of 20s. The values of the average Epsilon quality indicator of 50 runs with each representation is shown.

Problem Instance					DV-dir	DF	RK-2ptX	IP-WMX	NAMOA* 20s
obj	graph t.	l_{max}	n	ρ					
2	grid	5	100	-0.75	*** 1.0629	*** 1.0460	**1.0372**	*** 1.0585	-
				0.00	*** 1.1763	*** 1.2075	**1.1276**	*** 1.3225	-
			196	-0.75	*** 1.0827	*** 1.0512	**1.0431**	*** 1.0660	-
				0.00	*** 1.2008	*** 1.2637	**1.1276**	*** 1.3542	-
		10	100	-0.75	*** 1.0631	*** 1.0503	**1.0446**	*** 1.0765	-
				0.00	1.1610	*** 1.2572	**1.1558**	*** 1.3360	-
			196	-0.75	1.0456	1.0464	**1.0444**	*** 1.0643	-
				0.00	**1.3280**	1.7837	1.6194	*** 1.7287	-
	waxman	5	100	-0.75	*** 1.0540	1.0348	*** 1.0440	**1.0316**	1.0000
				0.00	*** 1.1723	*** 1.1248	1.0847	**1.0843**	1.0000
			196	-0.75	*** 1.0988	**1.0536**	1.0549	*** 1.0878	3.8131
				0.00	*** 1.1739	**1.0944**	*** 1.1550	*** 1.2450	4.1623
		10	100	-0.75	*** 1.1087	**1.1035**	* 1.1074	*** 1.1069	3.9561
				0.00	*** 1.4674	**1.2797**	1.2969	*** 1.5488	1.0000
			196	-0.75	*** 1.0860	**1.0520**	1.0555	*** 1.0611	-
				0.00	*** 1.2275	*** 1.1622	**1.1344**	*** 1.2238	-
3	grid	5	100	-0.75	**1.0434**	*** 1.0562	*** 1.0556	*** 1.0750	-
				0.00	** 1.2128	* 1.2135	**1.2009**	*** 1.3173	-
			196	-0.75	1.0731	* 1.0714	**1.0683**	*** 1.1146	-
				0.00	**1.1776**	*** 1.3229	*** 1.2379	*** 1.3307	-
		10	100	-0.75	* 1.1483	**1.1471**	* 1.1608	*** 1.1888	-
				0.00	**1.2214**	*** 1.3661	*** 1.3490	*** 1.5684	-
			196	-0.75	**1.0466**	*** 1.0926	*** 1.0846	*** 1.1476	-
				0.00	**1.2390**	*** 1.3881	*** 1.3575	*** 1.6481	-
	waxman	5	100	-0.75	*** 1.2193	**1.1998**	1.2012	** 1.2095	-
				0.00	*** 1.1976	**1.1468**	* 1.1512	*** 1.2200	5.4526
			196	-0.75	*** 1.0910	*** 1.0543	**1.0450**	*** 1.0670	3.1091
				0.00	*** 1.3924	*** 1.2732	**1.2353**	*** 1.3196	4.7317
		10	100	-0.75	**1.0640**	** 1.0734	1.0649	*** 1.0748	3.6113
				0.00	** 1.5479	**1.4930**	*** 1.6691	*** 1.7472	13.0856
			196	-0.75	*** 1.3506	** 1.1594	**1.1117**	*** 1.2578	-
				0.00	*** 1.8648	**1.5212**	1.5232	*** 2.0806	-

[a] The results of NAMOA* regarding the Epsilon quality indicator with the same time budget of 20s are shown in the last column, for comparison.

[b] The best values among the representations are shown in bold in each row.

[c] The one-sided Wilcoxon signed rank test was used to decide statistical significance between each variant compared to the best one in each row. Results are indicated as (*) : p<0.05, (**): p<0.005, (***): p<0.0005.

Table 6: Comparing the best variants of NSGA-II for the four representations, separately for the 32 test instances with a time budget of 10s. The values of the average Epsilon R3 quality indicator of 50 runs with each representation is shown.

Problem Instance					DV-dir	DF	RK-2ptX	IP-WMX	NAMOA* 10s
obj	graph t.	l_{max}	n	ρ					
2	grid	5	100	-0.75	*** 0.0123	**0.0091**	**0.0091**	*** 0.0173	-
				0.00	*** 0.0283	*** 0.0253	**0.0170**	*** 0.0327	-
			196	-0.75	*** 0.0156	* 0.0108	**0.0100**	*** 0.0177	-
				0.00	*** 0.0275	*** 0.0341	**0.0226**	*** 0.0485	-
		10	100	-0.75	*** 0.0170	**0.0149**	0.0152	*** 0.0297	-
				0.00	**0.0285**	*** 0.0371	0.0301	*** 0.0596	-
			196	-0.75	**0.0120**	0.0125	0.0123	*** 0.0198	-
				0.00	**0.0361**	*** 0.0511	*** 0.0432	*** 0.0722	-
	waxman	5	100	-0.75	*** 0.0016	**0.0007**	*** 0.0012	*** 0.0010	0.0000
				0.00	*** 0.0094	*** 0.0076	**0.0035**	0.0036	0.0633
			196	-0.75	*** 0.0169	**0.0054**	**0.0054**	*** 0.0137	0.2933
				0.00	*** 0.0141	**0.0056**	*** 0.0094	*** 0.0299	0.1820
		10	100	-0.75	*** 0.0074	** 0.0074	**0.0066**	*** 0.0078	0.2288
				0.00	*** 0.0294	*** 0.0181	**0.0103**	*** 0.0320	0.0000
			196	-0.75	*** 0.0110	*** 0.0077	**0.0067**	*** 0.0142	-
				0.00	*** 0.0253	** 0.0205	**0.0177**	*** 0.0247	-
3	grid	5	100	-0.75	**0.0094**	*** 0.0161	*** 0.0147	*** 0.0223	-
				0.00	**0.0280**	*** 0.0341	*** 0.0330	*** 0.0639	-
			196	-0.75	**0.0120**	*** 0.0194	*** 0.0190	*** 0.0296	-
				0.00	**0.0303**	*** 0.0637	*** 0.0520	*** 0.0831	-
		10	100	-0.75	**0.0248**	0.0251	*** 0.0289	*** 0.0391	-
				0.00	**0.0378**	*** 0.0817	*** 0.0815	*** 0.1084	-
			196	-0.75	**0.0107**	*** 0.0290	*** 0.0279	*** 0.0413	-
				0.00	**0.0349**	*** 0.0592	*** 0.0618	*** 0.1177	-
	waxman	5	100	-0.75	*** 0.0198	*** 0.0152	**0.0118**	*** 0.0132	-
				0.00	*** 0.0136	0.0092	**0.0091**	*** 0.0189	-
			196	-0.75	*** 0.0154	*** 0.0075	**0.0051**	*** 0.0116	0.2852
				0.00	*** 0.0477	0.0152	**0.0149**	*** 0.0403	0.2317
		10	100	-0.75	**0.0076**	*** 0.0160	*** 0.0124	*** 0.0191	0.3233
				0.00	** 0.0485	**0.0404**	0.0416	*** 0.0825	0.4407
			196	-0.75	*** 0.0353	*** 0.0165	**0.0140**	*** 0.0261	-
				0.00	*** 0.1204	** 0.0995	**0.0899**	*** 0.1190	-

[a] The results of NAMOA* regarding the R3 quality indicator with the same time budget of 10s are shown in the last column, for comparison.

[b] The best values among the representations are shown in bold in each row.

[c] The one-sided Wilcoxon signed rank test was used to decide statistical significance between each variant compared to the best one in each row. Results are indicated as (*) : p<0.05, (**): p<0.005, (***): p<0.0005.

Table 7: Comparing the best variants of NSGA-II for the four representations, separately for the 32 test instances with a time budget of 20s. The values of the average Epsilon R3 quality indicator of 50 runs with each representation is shown.

Problem Instance					DV-dir	DF	RK-2ptX	IP-WMX	NAMOA* 20s
obj	graph t.	l_{max}	n	ρ					
2	grid	5	100	-0.75	*** 0.0115	0.0090	**0.0070**	*** 0.0157	-
				0.00	*** 0.0265	*** 0.0237	**0.0143**	*** 0.0283	-
			196	-0.75	*** 0.0130	* 0.0088	**0.0077**	*** 0.0159	-
				0.00	*** 0.0264	*** 0.0316	**0.0195**	*** 0.0456	-
		10	100	-0.75	*** 0.0154	0.0135	**0.0131**	*** 0.0265	-
				0.00	0.0281	*** 0.0338	**0.0237**	*** 0.0563	-
			196	-0.75	0.0109	0.0105	**0.0091**	*** 0.0171	-
				0.00	**0.0348**	*** 0.0400	*** 0.0382	*** 0.0694	-
	waxman	5	100	-0.75	*** 0.0013	**0.0008**	*** 0.0011	*** 0.0009	0.0000
				0.00	*** 0.0083	*** 0.0062	0.0032	**0.0030**	0.0000
			196	-0.75	*** 0.0137	0.0055	**0.0042**	*** 0.0113	0.2190
				0.00	*** 0.0122	**0.0056**	*** 0.0080	*** 0.0280	0.1389
		10	100	-0.75	*** 0.0071	** 0.0076	**0.0066**	*** 0.0069	0.2063
				0.00	*** 0.0313	*** 0.0182	**0.0097**	*** 0.0319	0.0000
			196	-0.75	*** 0.0112	*** 0.0071	**0.0063**	*** 0.0118	-
				0.00	*** 0.0234	** 0.0208	**0.0153**	*** 0.0232	-
3	grid	5	100	-0.75	**0.0080**	*** 0.0155	*** 0.0130	*** 0.0206	-
				0.00	**0.0261**	*** 0.0318	*** 0.0297	*** 0.0605	-
			196	-0.75	**0.0110**	*** 0.0174	*** 0.0168	*** 0.0257	-
				0.00	**0.0280**	*** 0.0561	*** 0.0445	*** 0.0779	-
		10	100	-0.75	0.0240	**0.0237**	*** 0.0240	*** 0.0348	-
				0.00	**0.0342**	*** 0.0766	*** 0.0751	*** 0.0999	-
			196	-0.75	**0.0093**	*** 0.0255	*** 0.0249	*** 0.0384	-
				0.00	**0.0328**	*** 0.0517	*** 0.0549	*** 0.1132	-
	waxman	5	100	-0.75	*** 0.0165	*** 0.0151	**0.0111**	*** 0.0121	-
				0.00	*** 0.0124	0.0084	**0.0076**	*** 0.0173	0.2317
			196	-0.75	*** 0.0116	*** 0.0071	**0.0044**	*** 0.0106	0.2754
				0.00	*** 0.0456	0.0157	**0.0126**	*** 0.0329	0.2317
		10	100	-0.75	**0.0063**	*** 0.0165	*** 0.0118	*** 0.0176	0.3053
				0.00	** 0.0503	0.0400	**0.0371**	*** 0.0750	0.3919
			196	-0.75	*** 0.0311	*** 0.0169	**0.0120**	*** 0.0233	-
				0.00	*** 0.1160	** 0.0938	**0.0793**	*** 0.1148	-

[a] The results of NAMOA* regarding the R3 quality indicator with the same time budget of 20s are shown in the last column, for comparison.

[b] The best values among the representations are shown in bold in each row.

[c] The one-sided Wilcoxon signed rank test was used to decide statistical significance between each variant compared to the best one in each row. Results are indicated as (*) : p<0.05, (**): p<0.005, (***): p<0.0005.

Bibliography

Abbaspour, R. A. & Samadzadegan, F. (2010), 'An evolutionary solution for multimodal shortest path problem in metropolises.', *Comput. Sci. Inf. Syst.* **7**(4), 789–811. 48, 110, 111, 113

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. 24, 45, 46, 48, 109, 110, 113, 114, 143

Aliakbary, S., Habibi, J. & Movaghar, A. (2015), 'Feature extraction from degree distribution for comparison and analysis of complex networks', *The Computer Journal* **58**(9), 2079–2091. 74, 75

Barabási, A.-L. (2009), 'Scale-free networks: a decade and beyond', *science* **325**(5939), 412–413. 59

Barabási, A.-L. & Albert, R. (1999), 'Emergence of scaling in random networks', *science* **286**(5439), 509–512. 61, 64, 70

Barry, A., Griffith, J. & O'Riordan, C. (2015), An evolutionary and graph-rewriting based approach to graph generation, *in* '2015 7th International Joint Conference on Computational Intelligence (IJCCI)', Vol. 1, IEEE, pp. 237–243. 194

Bean, J. C. (1994), 'Genetic algorithms and random keys for sequencing and optimization', *ORSA journal on computing* **6**(2), 154–160. 47

Beke, L., Uribe, L., Lara, A., Coello, C. A. C., Weiszer, M., Burke, E. K. & Chen, J. (2023), 'Routing and scheduling in multigraphs with time constraints-a memetic approach for airport ground movement', *IEEE Transactions on Evolutionary Computation* . 1, 27

Beke, L., Weiszer, M. & Chen, J. (2021), 'A comparison of genetic representations and initialisation methods for the multi-objective shortest path problem on multigraphs', *SN Computer Science* **2**(3), 1–22. 1, 27, 86

Bellman, R. (1958), 'On a routing problem', *Quarterly of applied mathematics* **16**(1), 87–90. 35, 40

Bertsekas, D. P., Guerriero, F. & Musmanno, R. (1996), 'Parallel asynchronous label-correcting methods for shortest paths', *Journal of Optimization Theory and Applications* **88**(2), 297–320. 58

Bleviss, D. L. (2021), 'Transportation is critical to reducing greenhouse gas emissions in the united states', *Wiley Interdisciplinary Reviews: Energy and Environment* **10**(2), e390. 22

Blum, C. & Roli, A. (2003), 'Metaheuristics in combinatorial optimization: Overview and conceptual comparison', *ACM computing surveys (CSUR)* **35**(3), 268–308. 24, 38, 42, 43

Boldi, P. & Vigna, S. (2014), 'Axioms for centrality', *Internet Mathematics* **10**(3-4), 222–262. 75

Bonacich, P. (1987), 'Power and centrality: A family of measures', *American journal of sociology* **92**(5), 1170–1182. 75

Breugem, T., Dollevoet, T. & van den Heuvel, W. (2017), 'Analysis of fptases for the multi-objective shortest path problem', *Computers & Operations Research* **78**, 44–58. 60

Brownlee, A. E., Woodward, J. R., Weiszer, M. & Chen, J. (2018), A rolling window with genetic algorithm approach to sorting aircraft for automated taxi routing, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference', pp. 1207–1213. 195

Brumbaugh-Smith, J. & Shier, D. (1989), 'An empirical investigation of some bicriterion shortest path algorithms', *European Journal of Operational Research* **43**(2), 216–224. 56, 58, 60, 71, 72, 74

Burke, E. K., Newall, J. P. & Weare, R. F. (1998), 'Initialization strategies and diversity in evolutionary timetabling', *Evolutionary computation* **6**(1), 81–103. 48

Calvete, H. I., del Pozo, L. & Iranzo, J. A. (2017), 'The energy-constrained quickest path problem', *Optimization Letters* **11**(7), 1319–1339. 60

Caramia, M., Giordani, S. & Iovanella, A. (2010), 'On the selection of k routes in multiobjective hazmat route planning', *IMA Journal of Management Mathematics* **21**(3), 239–251. 35, 60

Carraway, R. L., Morin, T. L. & Moskowitz, H. (1990), 'Generalized dynamic programming for multicriteria optimization', *European journal of operational research* **44**(1), 95–104. 51

Chatterjee, A., Manohar, M. & Ramadurai, G. (2016), 'Statistical analysis of bus networks in india', *PloS one* **11**(12), e0168478. 62

Chen, B. Y., Chen, X.-W., Chen, H.-P. & Lam, W. H. (2020), 'Efficient algorithm for finding k shortest paths based on re-optimization technique', *Transportation Research Part E: Logistics and Transportation Review* **133**, 101819. 60

Chen, B. Y., Lam, W. H., Sumalee, A., Li, Q., Shao, H. & Fang, Z. (2013), 'Finding reliable shortest paths in road networks under uncertainty', *Networks and spatial economics* **13**(2), 123–148. 58

Chen, J., Weiszer, M., Locatelli, G., Ravizza, S., Atkin, J. A., Stewart, P. & Burke, E. K. (2016), 'Toward a more realistic, cost-effective, and greener ground movement through active routing: A multiobjective shortest path approach', *IEEE Transactions on Intelligent Transportation Systems* **17**(12), 3524–3540. 24, 35, 38, 39, 50, 51, 138, 163, 165

Chen, J., Weiszer, M., Stewart, P. & Shabani, M. (2016), 'Toward a more realistic, cost-effective, and greener ground movement through active routing—part i: Optimal speed profile generation', *IEEE Transactions on Intelligent Transportation Systems* . 51

Chen, T. & Guestrin, C. (2016), Xgboost: A scalable tree boosting system, *in* 'Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining', pp. 785–794. 63

Cherkassky, B. V., Goldberg, A. V. & Radzik, T. (1996), 'Shortest paths algorithms: Theory and experimental evaluation', *Mathematical programming* **73**(2), 129–174. 35, 60, 71

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. 35, 38, 42, 45, 57, 109, 180

Climaco, J. C. N. & Martins, E. Q. V. (1982), 'A bicriterion shortest path algorithm', *European Journal of Operational Research* **11**(4), 399–404. 37

Clímaco, J. C. & Pascoal, M. M. (2012), 'Multicriteria path and tree problems: discussion on exact algorithms and applications', *International Transactions in Operational Research* **19**(1-2), 63–98. 37

Coello, C. A. C. (2021), Constraint-handling techniques used with evolutionary algorithms, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference Companion', pp. 692–714. 49

Coello Coello, C. A. (2009), 'Evolutionary multi-objective optimization: some current research trends and topics that remain to be explored', *Frontiers of Computer Science in China* **3**(1), 18–30. 49

Coltin, B. & Veloso, M. (2014), Ridesharing with passenger transfers, *in* '2014 IEEE/RSJ International Conference on Intelligent Robots and Systems', IEEE, pp. 3278–3283. 35, 52

Danloup, N., Mirzabeiki, V., Allaoui, H., Goncalves, G., Julien, D. & Mena, C. (2015), 'Reducing transportation greenhouse gas emissions with collaborative distribution: a case study', *Management Research Review* . 22

De Benedictis, L. & Taglioni, D. (2011), The gravity model in international trade, *in* 'The trade impact of European Union preferential policies', Springer, pp. 55–89. 62, 65

de las Casas, P. M., Sedeño-Noda, A. & Borndörfer, R. (2021), 'An improved multiobjective shortest path algorithm', *Computers & Operations Research* p. 105424. 58

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. 44, 125

Deb, K., Mohan, M. & Mishra, S. (2003), Towards a quick computation of well-spread pareto-optimal solutions, *in* 'Evolutionary Multi-Criterion Optimization: Second International Conference, EMO 2003, Faro, Portugal, April 8–11, 2003. Proceedings 2', Springer, pp. 222–236. 45

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. 44, 125

Deng, W., Li, W., Cai, X. & Wang, Q. A. (2011), 'The exponential degree distribution in complex networks: Non-equilibrium network theory, numerical simulation and empirical data', *Physica A: Statistical Mechanics and its Applications* **390**(8), 1481–1485. 65, 66, 70

Deng, Y., Liu, Y. & Zhou, D. (2015), 'An improved genetic algorithm with initial population strategy for symmetric tsp', *Mathematical Problems in Engineering* **2015**. 48

Desrochers, M. & Soumis, F. (1988), 'A generalized permanent labelling algorithm for the shortest path problem with time windows', *INFOR: Information Systems and Operational Research* **26**(3), 191–212. 38

Desrosiers, J., Pelletier, P. & Soumis, F. (1983), 'Plus court chemin avec contraintes d'horaires', *RAIRO-Operations Research* **17**(4), 357–377. 38

Dib, O., Dib, M. & Caminada, A. (2018), 'Computing multicriteria shortest paths in stochastic multimodal networks using a memetic algorithm', *International Journal on Artificial Intelligence Tools* **27**(07), 1860012. 42, 142

Dijkstra, E. W. et al. (1959), 'A note on two problems in connexion with graphs', *Numerische mathematik* **1**(1), 269–271. 23, 35, 39, 142

Ehrgott, M. & Gandibleux, X. (2000), 'A survey and annotated bibliography of multiobjective combinatorial optimization', *OR-Spektrum* **22**(4), 425–460. 34, 37, 57, 58, 86

Enzi, M., Parragh, S. N. & Puchinger, J. (2020), 'The bi-objective multimodal car-sharing problem', *arXiv preprint arXiv:2010.10344* . 35, 52

Erdös, P. & Rényi, A. (1959), 'On random graphs, i', *Publicationes Mathematicae (Debrecen)* **6**, 290–297. 59, 64, 70

Fernandez, S. A., Juan, A. A., de Armas Adrián, J., e Silva, D. G. & Terrén, D. R. (2018), 'Metaheuristics in telecommunication systems: network design, routing, and allocation problems', *IEEE Systems Journal* **12**(4), 3948–3957. 109

Fonseca, C. M., Knowles, J. D., Thiele, L. & Zitzler, E. (2005), A tutorial on the performance assessment of stochastic multiobjective optimizers, *in* 'Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)', Vol. 216, p. 240. 54, 128

Fruchterman, T. M. & Reingold, E. M. (1991), 'Graph drawing by force-directed placement', *Software: Practice and experience* **21**(11), 1129–1164. 89, 97

Galand, L., Perny, P. & Spanjaard, O. (2010), 'Choquet-based optimisation in multiobjective shortest path and spanning tree problems', *European Journal of Operational Research* **204**(2), 303–315. 60

Gallet, M., Massier, T. & Hamacher, T. (2018), 'Estimation of the energy demand of electric buses based on real-world data for large-scale public transport networks', *Applied energy* **230**, 344–356. 24

Gallo, G. & Pallottino, S. (1986), Shortest path methods: A unifying approach, *in* 'Netflow at Pisa', Springer, pp. 38–64. 35

Gallo, G. & Pallottino, S. (1988), 'Shortest path algorithms', *Annals of operations research* **13**(1), 1–79. 53, 60

Garaix, T., Artigues, C., Feillet, D. & Josselin, D. (2010), 'Vehicle routing problems with alternative paths: An application to on-demand transportation', *European Journal of Operational Research* **204**(1), 62–75. 24, 38, 52

Gen, M., Altiparmak, F. & Lin, L. (2006), 'A genetic algorithm for two-stage transportation problem using priority-based encoding', *OR spectrum* **28**(3), 337–354. 109

Gen, M., Cheng, R. & Lin, L. (2008), *Network models and optimization: Multiobjective genetic algorithm approach*, Springer Science & Business Media. 109

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. 46, 47, 118

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', ACM, pp. 1411–1412. 47, 117, 120

Ghoseiri, K. & Nadjari, B. (2010), 'An ant colony optimization algorithm for the bi-objective shortest path problem', *Applied soft computing* **10**(4), 1237–1246. 43, 58, 195

Gilbert, E. N. (1961), 'Random plane networks', *Journal of the Society for Industrial and Applied Mathematics* **9**(4), 533–543. 62

Glabowski, M., Musznicki, B., Nowak, P. & Zwierzykowski, P. (2012), 'Shortest path problem solving based on ant colony optimization metaheuristic', *Image Processing & Communications* **17**(1-2), 7. 43, 195

Glover, F. (1986), 'Future paths for integer programming and links to artificial intelligence', *Computers & operations research* **13**(5), 533–549. 42

Gove, R. (2019), Gragnostics: Fast, interpretable features for comparing graphs, *in* '2019 23rd International Conference Information Visualisation (IV)', IEEE, pp. 201–209. 74, 75

Guerriero, F. & Musmanno, R. (2001), 'Label correcting methods to solve multicriteria shortest path problems', *Journal of optimization theory and applications* **111**(3), 589–613. 41, 57, 60, 72

Guerriero, F., Musmanno, R., Lacagnina, V. & Pecorella, A. (2001), 'A class of label-correcting methods for the k shortest paths problem', *Operations Research* **49**(3), 423–429. 57

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). 88

Hamacher, H. W., Labbe, M., Nickel, S. & Skriver, A. J. (2002), 'Multicriteria semi-obnoxious network location problems (msnlp) with sum and center objectives', *Annals of Operations Research* **110**(1), 33–53. 58

Hansen, M. P. & Jaszkiewicz, A. (1994), *Evaluating the quality of approximations to the non-dominated set*, Technical University of Denmark. 54

Hansen, P. (1980), Bicriterion path problems, *in* 'Multiple criteria decision making theory and application', Springer, pp. 109–127. 36, 37, 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. 40

Hasan, B. S., Khamees, M. A. & Mahmoud, A. S. H. (2007), A heuristic genetic algorithm for the single source shortest path problem, *in* '2007 IEEE/ACS International Conference on Computer Systems and Applications', IEEE, pp. 187–194. 48

Hassan, M., Hamid, A. & Alkinani, M. (2020), 'Ant colony optimization for multi-objective multicast routing', *Computers, Materials & Continua* **63**(3), 1159–1173. 43, 195

Hernández, J. M. & Van Mieghem, P. (2011), 'Classification of graph metrics', *Delft University of Technology: Mekelweg, The Netherlands* pp. 1–20. 73, 75

Holland, J. H. (1992), 'Genetic algorithms', *Scientific american* **267**(1), 66–73. 24, 43

Holme, P. & Kim, B. J. (2002), 'Growing scale-free networks with tunable clustering', *Physical review E* **65**(2), 026107. 64, 70

Hrnčíř, J., Rovatsos, M. & Jakob, M. (2015), 'Ridesharing on timetabled transport services: A multiagent planning approach', *Journal of Intelligent Transportation Systems* **19**(1), 89–105. 52

Huarng, F., Pulat, P. & Shih, L. (1996), 'A computational comparison of some bicriterion shortest path algorithms', *Journal of the Chinese Institute of Industrial Engineers* **13**(2), 121–125. 37, 57

Humphries, M. D. & Gurney, K. (2008), 'Network 'small-world-ness': a quantitative method for determining canonical network equivalence', *PloS one* **3**(4), e0002051. 75

Inagaki, J., Haseyama, M. & Kitajima, H. (1999), A genetic algorithm for determining multiple routes and its applications, *in* 'ISCAS'99. Proceedings of the 1999 IEEE International Symposium on Circuits and Systems VLSI (Cat. No. 99CH36349)', Vol. 6, IEEE, pp. 137–140. 46, 109, 115

Ioachim, I., Gelinas, S., Soumis, F. & Desrosiers, J. (1998), 'A dynamic programming algorithm for the shortest path problem with time windows and linear node costs', *Networks: An International Journal* **31**(3), 193–204. 38

Irnich, S. & Desaulniers, G. (2005), Shortest path problems with resource constraints, *in* 'Column generation', Springer, pp. 33–65. 37

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. 45, 48, 49

Jung, W.-S., Wang, F. & Stanley, H. E. (2008), 'Gravity model in the korean highway', *EPL (Europhysics Letters)* **81**(4), 48005. 62, 65

Kavitha, T., Mehlhorn, K., Michail, D. & Paluch, K. E. (2008), 'An o (m2n) algorithm for minimum cycle basis of graphs', *Algorithmica* **52**, 333–349. 75

Khadilkar, H. & Balakrishnan, H. (2012), 'Estimation of aircraft taxi fuel burn using flight data recorder archives', *Transportation Research Part D: Transport and Environment* **17**(7), 532–537. 169

Kim, J. H. & Vu, V. H. (2003), Generating random regular graphs, *in* 'Proceedings of the thirty-fifth annual ACM symposium on Theory of computing', pp. 213–222. 70

Klingman, D., Napier, A. & Stutz, J. (1973), Netgen: A program for generating large scale (un) capacitated assignment, transportation, and minimum cost flow network problems, Technical report, TEXAS UNIV AT AUSTIN CENTER FOR CYBERNETIC STUDIES. 56, 57, 58, 65

Knowles, J. & Corne, D. (1999), The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation, *in* 'Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)', Vol. 1, IEEE, pp. 98–105. 45

Knowles, J. & Corne, D. (2002), On metrics for comparing nondominated sets, *in* 'Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)', Vol. 1, IEEE, pp. 711–716. 54

Kogotkova, S., Oehlers, M., Ermakova, T. & Fabian, B. (2018), 'Correlation analysis of local graph metrics', *Available at SSRN 3192028* . 75

Krömer, P., Uher, V. & Snášel, V. (2021), 'Novel random key encoding schemes for the differential evolution of permutation problems', *IEEE Transactions on Evolutionary Computation* . 195

Lai, D. S., Demirag, O. C. & Leung, J. M. (2016), 'A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph', *Transportation Research Part E: Logistics and Transportation Review* **86**, 32–52. 52

Latora, V. & Marchiori, M. (2001), 'Efficient behavior of small-world networks', *Physical review letters* **87**(19), 198701. 75

Lee, J. & Kim, D.-W. (2016), 'An effective initialization method for genetic algorithm-based robot path planning using a directed acyclic graph', *Information Sciences* **332**, 1–18. 48

Lesire, C. (2010), An iterative a* algorithm for planning of airport ground movements., *in* 'ECAI', Vol. 2010, pp. 413–418. 50, 51

Li, C., Wang, H., De Haan, W., Stam, C. & Van Mieghem, P. (2011), 'The correlation of metrics in complex networks with applications in functional brain networks', *Journal of Statistical Mechanics: Theory and Experiment* **2011**(11), P11018. 75

Li, D., Yang, M., Jin, C.-J., Ren, G., Liu, X. & Liu, H. (2020), 'Multi-modal combined route choice modeling in the maas age considering generalized path overlapping problem', *IEEE Transactions on Intelligent Transportation Systems* . 52, 87

Li, R., Leung, Y., Huang, B. & Lin, H. (2013), 'A genetic algorithm for multiobjective dangerous goods route planning', *International Journal of Geographical Information Science* **27**(6), 1073–1089. 35, 42, 45, 48, 109, 122, 124

Liefooghe, A. & Derbel, B. (2016), A correlation analysis of set quality indicator values in multiobjective optimization, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference 2016', pp. 581–588. 54, 180

Lin, J. & Ban, Y. (2013), 'Complex network topology of transportation systems', *Transport reviews* **33**(6), 658–685. 61, 62, 75

Lin, L. & Gen, M. (2008), 'An effective evolutionary approach for bicriteria shortest path routing problems', *IEEJ Transactions on Electronics, Information and Systems* **128**(3), 416–423. 24, 47, 109, 116, 118, 120

Lin, L. & Gen, M. (2009), Priority-based genetic algorithm for shortest path routing problem in ospf, *in* 'Intelligent and Evolutionary Systems', Springer, pp. 91–103. 46

Liu, Q., Li, X., Liu, H. & Guo, Z. (2020), 'Multi-objective metaheuristics for discrete optimization problems: A review of the state-of-the-art', *Applied Soft Computing* **93**, 106382. 42

Loui, R. P. (1983), 'Optimal paths in graphs with stochastic or multidimensional weights', *Communications of the ACM* **26**(9), 670–676. 37

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T. & Birattari, M. (2016), 'The irace package: Iterated racing for automatic algorithm configuration', *Operations Research Perspectives* **3**, 43–58. 125, 177, 178

Machuca, E., Mandow, L., de la Cruz, J. L. P. & Ruiz-Sepulveda, A. (2010), An empirical comparison of some multiobjective graph search algorithms, *in* 'Annual Conference on Artificial Intelligence', Springer, pp. 238–245. 41, 58, 59, 69, 72, 127

Machuca Sánchez, E. et al. (2012), 'An analysis of some algorithms and heuristics for multiobjective graph search'. 25, 57, 72, 76, 155

Mandow, L. & De La Cruz, J. L. P. (2010), 'Multiobjective a* search with consistent heuristics', *Journal of the ACM (JACM)* **57**(5), 27. 41, 55

Mandow, L., De la Cruz, J. P. et al. (2005), A new approach to multiobjective a* search., *in* 'IJCAI', Vol. 8, Citeseer. 41, 51, 63, 75, 128

Marinakis, Y., Migdalas, A. & Sifaleras, A. (2017), 'A hybrid particle swarm optimization–variable neighborhood search algorithm for constrained shortest path problems', *European Journal of Operational Research* **261**(3), 819–834. 43

Maristany de las Casas, P., Borndörfer, R., Kraus, L. & Sedeño-Noda, A. (2021), 'An fptas for dynamic multiobjective shortest path problems', *Algorithms* **14**(2), 43. 71, 72

Martins, E. Q., Paixão, J. M., Rosa, M. S. & Santos, J. L. (1999), 'Ranking multiobjective shortest paths', *International Journal of Foundations of Computer Science* **10**(3), 247–261. 60

Martins, E. Q., Paixão, J. M., Rosa, M. S. & Santos, J. L. (2007), 'Ranking multiobjective shortest paths'. 58

Meng, Q., Lee, D.-H. & Cheu, R. L. (2005), 'Multiobjective vehicle routing and scheduling problem with time window constraints in hazardous material transportation', *Journal of transportation engineering* **131**(9), 699–707. 25, 35

Mezura-Montes, E. & Coello, C. A. C. (2011), 'Constraint-handling in nature-inspired numerical optimization: past, present and future', *Swarm and Evolutionary Computation* **1**(4), 173–194. 49

Michalewicz, Z. & Schoenauer, M. (1996), 'Evolutionary algorithms for constrained parameter optimization problems', *Evolutionary computation* **4**(1), 1–32. 49

Miranda, D. M., Branke, J. & Conceição, S. V. (2018), 'Algorithms for the multi-objective vehicle routing problem with hard time windows and stochastic travel time and service time', *Applied Soft Computing* **70**, 66–79. 49

Mohemmed, A. W., Sahoo, N. C. & Geok, T. K. (2008), 'Solving shortest path problem using particle swarm optimization', *Applied Soft Computing* **8**(4), 1643–1653. 43, 46, 49, 120

Mohri, M. (1997), 'Finite-state transducers in language and speech processing', *Computational linguistics* **23**(2), 269–311. 21

Moscato, P. & Norman, M. G. (1992), 'A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems', *Parallel computing and transputer applications* **1**, 177–186. 142

Mote, J., Murthy, I. & Olson, D. L. (1991), 'A parametric approach to solving bicriterion shortest path problems', *European Journal of Operational Research* **53**(1), 81–92. 37, 59, 60, 68

Müller-Hannemann, M. & Weihe, K. (2001), Pareto shortest paths is often feasible in practice, *in* 'International Workshop on Algorithm Engineering', Springer, pp. 185–197. 39, 53, 72, 74, 78

Munetomo, M., Takai, Y. & Sato, Y. (1997), An adaptive network routing algorithm employing path genetic operators, *in* 'ICGA'. 45

Newman, M. E. & Watts, D. J. (1999), 'Renormalization group analysis of the small-world network model', *Physics Letters A* **263**(4-6), 341–346. 65, 70

Nielsen, L. R., Andersen, K. A. & Pretolani, D. (2003), 'Bicriterion shortest hyperpaths in random time-dependent networks', *IMA Journal of Management Mathematics* **14**(3), 271–303. 59

Osaragi, T. & Hiraga, Y. (2014), Street network created by proximity graphs: its topological structure and travel efficiency, *in* 'Connecting a Digital Europe through Location and Place. Proceedings of the AGILE'2014 International Conference on Geographic Information Science, Castellón, June, 3-6, 2014. ISBN: 978-90-816960-4-3', Springer. 62

Paixão, J. M. & Santos, J. L. (2013), Labeling methods for the general case of the multi-objective shortest path problem–a computational study, *in* 'Computational Intelligence and Decision Making', Springer, pp. 489–502. 60

Pajor, T. (2009), 'Multi-modal route planning'. 53

Pangilinan, J. M. A. & Janssens, G. K. (2007), 'Evolutionary algorithms for the multiobjective shortest path problem', *International Journal of Mathematical and Computational Sciences* **1**(1), 7–12. 42

Penrose, M. et al. (2003), *Random geometric graphs*, Vol. 5, Oxford university press. 67

Powell, W. B. & Chen, Z.-L. (1997), A generalized threshold algorithm for the shortest path problem with time windows., *in* 'Network Design: Connectivity and Facilities Location', pp. 303–318. 38

Psaraftis, H. N. & Kontovas, C. A. (2013), 'Speed models for energy-efficient maritime transportation: A taxonomy and survey', *Transportation Research Part C: Emerging Technologies* **26**, 331–351. 52

Psaraftis, H. N. & Kontovas, C. A. (2014), 'Ship speed optimization: Concepts, models and combined speed-routing scenarios', *Transportation Research Part C: Emerging Technologies* **44**, 52–69. 52, 87

Pugliese, L. D. P., Granat, J. & Guerriero, F. (2020), 'Two-phase algorithm for solving the preference-based multicriteria optimal path problem with reference points', *Computers & Operations Research* **121**, 104977. 60

Pugliese, L. D. P. & Guerriero, F. (2013), 'A survey of resource constrained shortest path problems: Exact solution approaches', *Networks* **62**(3), 183–200. 38

Pulido, F.-J., Mandow, L. & Pérez-de-la Cruz, J.-L. (2015), 'Dimensionality reduction in multiobjective shortest path search', *Computers & Operations Research* **64**, 60–70. 60

Raith, A. & Ehrgott, M. (2009), 'A comparison of solution strategies for biobjective shortest path problems', *Computers & Operations Research* **36**(4), 1299–1331. 25, 37, 56, 58, 60, 71, 72

Ravizza, S., Atkin, J. A. & Burke, E. K. (2014), 'A more realistic approach for airport ground movement optimisation with stand holding', *Journal of Scheduling* **17**(5), 507–520. 50, 51

Ravizza, S., Chen, J., Atkin, J. A., Burke, E. K. & Stewart, P. (2013), 'The trade-off between taxi time and fuel consumption in airport ground movement', *Public Transport* **5**(1-2), 25–40. 24, 50, 163, 169

Santoro, A., Latora, V., Nicosia, G. & Nicosia, V. (2018), 'Pareto optimality in multilayer network growth', *Physical review letters* **121**(12), 128302. 62, 65, 67, 70

Sasaki, D. & Obayashi, S. (2005), 'Efficient search for trade-offs by adaptive range multi-objective genetic algorithms', *Journal of Aerospace Computing, Information, and Communication* **2**(1), 44–64. 45

Seada, H. & Deb, K. (2014), 'U-nsga-iii: A unified evolutionary algorithm for single, multiple, and many-objective optimization', *COIN report* **2014022**. 125

Sedeno-Noda, A. & Colebrook, M. (2019), 'A biobjective dijkstra algorithm', *European Journal of Operational Research* **276**(1), 106–118. 60

Serafini, P. (1987), Some considerations about computational complexity for multi objective combinatorial problems, *in* 'Recent advances and historical development of vector optimization', Springer, pp. 222–232. 39

Shi, N., Zhou, S., Wang, F., Tao, Y. & Liu, L. (2017), 'The multi-criteria constrained shortest path problem', *Transportation Research Part E: Logistics and Transportation Review* **101**, 13–29. 38

Skriver, A. J. & Andersen, K. A. (2000), 'A label correcting approach for solving bicriterion shortest-path problems', *Computers & Operations Research* **27**(6), 507–524. 25, 56, 57, 58, 60, 68, 70, 72

Smith-Miles, K. & Bowly, S. (2015), 'Generating new test instances by evolving in instance space', *Computers & Operations Research* **63**, 102–113. 194

Soroush, H. (2008), 'Optimal paths in bi-attribute networks with fractional cost functions', *European journal of operational research* **190**(3), 633–658. 58

Srinivas, N. & Deb, K. (1994), 'Muiltiobjective optimization using nondominated sorting in genetic algorithms', *Evolutionary computation* **2**(3), 221–248. 44

Strobl, C., Malley, J. & Tutz, G. (2009), 'An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests.', *Psychological methods* **14**(4), 323. 81

Telesford, Q. K., Joyce, K. E., Hayasaka, S., Burdette, J. H. & Laurienti, P. J. (2011), 'The ubiquity of small-world networks', *Brain connectivity* **1**(5), 367–375. 75

Ticha, H. B., Absi, N., Feillet, D. & Quilliot, A. (2017), 'Empirical analysis for the vrptw with a multigraph representation for the road network', *Computers & Operations Research* **88**, 103–116. 52

Tonda, A. (2020), 'Inspyred: Bio-inspired algorithms in python', *Genetic Programming and Evolvable Machines* **21**(1-2), 269–272. 125, 147, 180

Toussaint, G. T. (1980), 'The relative neighbourhood graph of a finite planar set', *Pattern recognition* **12**(4), 261–268. 67

Tu, Q., Cheng, L., Yuan, T., Cheng, Y. & Li, M. (2020), 'The constrained reliable shortest path problem for electric vehicles in the urban transportation network', *Journal of Cleaner Production* **261**, 121130. 35

Tung, C. T. & Chew, K. L. (1992), 'A multicriteria pareto-optimal path algorithm', *European Journal of Operational Research* **62**(2), 203–209. 41, 128

Watts, D. J. & Strogatz, S. H. (1998), 'Collective dynamics of 'small-world'networks', *nature* **393**(6684), 440. 61, 62

Waxman, B. M. (1988), 'Routing of multipoint connections', *IEEE journal on selected areas in communications* **6**(9), 1617–1622. 127

Weise, J. & Mostaghim, S. (2021), 'A scalable many-objective pathfinding benchmark suite', *IEEE Transactions on Evolutionary Computation* . 44, 59, 60, 194

Weiszer, M., Burke, E. K. & Chen, J. (2020), 'Multi-objective routing and scheduling for airport ground movement', *Transportation Research Part C: Emerging Technologies* **119**, 102734. 24, 51, 87, 96, 138, 164, 166, 168, 169, 184, 190

Weiszer, M., Burke, E. K. & Chen, J. (2021), 'A note on multi-graph structure for constrained routing and scheduling problems', *Unpublished manuscript* **-**, –. 38, 190

Weiszer, M., Chen, J. & Stewart, P. (2015), 'A real-time active routing approach via a database for airport surface movement', *Transportation Research Part C: Emerging Technologies* **58**, 127–145. 165, 166, 169

Wen, M., Pacino, D., Kontovas, C. & Psaraftis, H. (2017), 'A multiple ship routing and speed optimization problem under time, cost and environmental objectives', *Transportation Research Part D: Transport and Environment* **52**, 303–321. 25, 52

Wu, X., He, X., Yu, G., Harmandayan, A. & Wang, Y. (2015), 'Energy-optimal speed control for electric vehicles on signalized arterials', *IEEE Transactions on Intelligent Transportation Systems* **16**(5), 2786–2796. 52, 87

Xiong, G. & Wang, Y. (2014), 'Best routes selection in multimodal networks using multi-objective genetic algorithm', *Journal of Combinatorial Optimization* **28**(3), 655–673. 24, 35, 42, 52

Yang, X., Li, X., Ning, B. & Tang, T. (2016), 'A survey on energy-efficient train operation for urban rail transit', *IEEE Transactions on Intelligent Transportation Systems* **17**(1), 2–13. 25, 52, 87

Yu, H. & Lu, F. (2012), 'A multi-modal route planning approach with an improved genetic algorithm', *Advances in Geo-Spatial Information Science* **193**. 35, 48, 110, 111

Z (n.d.*a*), 'Icao, 2004. advanced surface movement guidance and control systems (a-smgcs) manual. international civil aviation organization', `http://www.icao.int/Meetings/anconf12/Document`. 181, 188

Z (n.d.*b*), 'This research utilised queen mary's apocrita hpc facility, supported by qmul research-it.', `http://doi.org/10.5281/zenodo.438045`. 28, 77, 96, 125, 147, 180

Zeng, W. & Church, R. L. (2009), 'Finding shortest paths on real road networks: the case for a', *International journal of geographical information science* **23**(4), 531–543. 60

Zhan, F. B. & Noon, C. E. (1998), 'Shortest path algorithms: an evaluation using real road networks', *Transportation science* **32**(1), 65–73. 60

Zhang, P., Wang, J., Li, X., Li, M., Di, Z. & Fan, Y. (2008), 'Clustering coefficient and community structure of bipartite networks', *Physica A: Statistical Mechanics and its Applications* **387**(27), 6869–6875. 75

Zhang, Q. & Li, H. (2007), 'Moea/d: A multiobjective evolutionary algorithm based on decomposition', *IEEE Transactions on evolutionary computation* **11**(6), 712–731. 125

Zitzler, E. & Thiele, L. (1998), 'An evolutionary algorithm for multiobjective optimization: The strength pareto approach', *TIK report* **43**. 45

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M. & Da Fonseca, V. G. (2003), 'Performance assessment of multiobjective optimizers: An analysis and review', *IEEE Transactions on evolutionary computation* **7**(2), 117–132. 54