

Genetic Algorithms for Workforce Scheduling and Routing Problem

by Haneen Algethami, MSc

Thesis submitted to The University of Nottingham for the degree of Doctor of Philosophy

February 2017

Abstract

The Workforce Scheduling and Routing Problem (WSRP) is described as the assignment of personnel to visits across various geographical locations. Solving this problem demands tackling scheduling and routing constraints while aiming to minimise the total operational cost. With current computational capabilities, small WSRPs are solvable using exact methods. However, it is difficult to solve when they are larger. The difficulty of WSRP is further increased when processing conflicting assignments or dealing with workers unavailability at customer's areas. Genetic Algorithms (GAs) have proved their effectiveness in these regards, because of their search capability to acquire good solutions in a reasonable computational time. A GA consists of many components, which can be chosen and combined in numerous procedures. In the case of solving scheduling and routing problems separately, different GAs have been proposed. When solving WSRP problem instances, it has been quite common to use the design components, intended for scheduling or routing problems. In this thesis, 42 real-world Home Health Care (HHC) planning problem datasets are used as instances of the WSRP. Different GA components are presented in this study, tailored for the combined settings. This has made major contributions to understanding how GAs works in a challenging real-world problem. Research interests in this work are categorised into two parts. The first part aims to understand how to employ different genetic operators effectively when solving WSRPs. The work intends to design and select the best combination of components that provide good solutions. Accordingly, seven well-known crossovers, three mutation operators and eight cost-based operators are implemented. In addition, two repair heuristics to tackle infeasibility. Nevertheless, a direct chromosome representation has resulted in poor solutions. Thus, there is a need for more tailored components for this problem. Therefore, an indirect chromosome representation, designed specifically to tackle WSRPs, is presented. The aim is to ensure initial solutions feasibility. Due to the quality of solutions, the GA introduced is considered an effective baseline evolutionary algorithm for WSRP.

This work also suggested that each problem set requires different parameter settings. The second research interest intends to increase the GA efficiency. One approach is to investigate the effect of using adaptive components on the quality of WSRPs solutions. The aim is to adaptively alter parameter values instead of tuning an algorithm to a specific instance. Three aspects are adjusted during the run according to different rules: operator rates, population size, and crossover operator function. Thus, six variations of a diversity-based adaptive GA is presented. Not only the adaptive GA has improved the results, especially for large WSRP scenarios, but also it reduces the computational time. Another aspect investigated is the effect of using a group of crossover operators rather than using one operator throughout the search. Six crossover operators, well known and problem-specific are used as part of a multiple crossover GA framework. To evaluate an operator effectiveness, a reinforcement-learning model is developed with three performance measurements. The most successful variant of this algorithm finds the best-known results for the larger problem instances and matching the best-known results for some of the smaller ones. When combining this method with the adaptive GA, it provided some of the best results, as compared to established algorithms. The presented methods have contributed in reducing the operational costs for this constrained combinatorial optimisation problem.

Contents

Abstract iii

Table of Contents vi

List of Figures x

List of Tables xiii

List of Algorithms xiv

Acknowledgements 1

1 Introduction 1

1.1 Background and Motivation 1

1.2 Why Are GAs Used in This Study? 4

1.3 Thesis Objectives 6

1.4 Key Contributions 7

1.5 Publications 9

1.6 Organisation of the Thesis 10

2 Workforce Scheduling and Routing problem 12

2.1 Introduction 12

2.2 Main Features of WSRP 12

2.2.1 Problem Constraints 13

2.2.2 Problem Objectives 15

2.2.3 Weights Calculations 17

2.3 Problem instances 18

2.3.1 Why WSRPs are Difficult to Solve? 19

2.4 Solution Methods for Solving WSRPs . . . 21
2.4.1 Time-dependent Connected Tasks Constraints . . . 22
2.4.2 Non-Connected Tasks Constraints . . . 23
2.5 Summary and Final Remarks . . . 25

3 An Overview on Genetic Algorithms 26
3.1 Introduction . . . 26
3.2 Genetic Algorithms . . . 26
3.2.1 Chromosome Representation . . . 28
3.2.2 Initialisation and Evaluation . . . 30
3.2.3 Selection . . . 31
3.2.4 Search Operators . . . 32
3.2.5 GA Parameters . . . 36
3.2.6 Replacement strategies . . . 38
3.2.7 Optimising the GA performance . . . 38
3.3 Research Review on WSRP Related Problems . . . 41
3.3.1 Scheduling Problems . . . 41
3.3.2 Routing Problems . . . 43
3.3.3 The Combined Scheduling and Routing Problem . . . 48
3.4 Guidelines for designing a GA for real-world problems . . . 52
3.5 How to design a GA to solve WSRPs? . . . 53
3.6 Summary and Final Remarks . . . 56

4 Selecting Genetic Operators for WSRP 57
4.1 Introduction . . . 57
4.2 Genetic Algorithm and Operators . . . 58
4.2.1 Chromosome Representation . . . 58
4.2.2 Steady State GA vs. Generational GA . . . 59
4.2.3 Repair Operators . . . 60
4.2.4 Genetic Operators . . . 62
4.3 Experimental Study and Results . . . 64

4.3.1 Performance of Repair Operators . . . 66
4.3.2 Maximising Preferences-Constraints Satisfaction . . . 72
4.3.3 Effect of Using Different Combinations of Operators with Different Rates . . . 74
4.3.4 Computational Results for WSRPs Instances . . . 77
4.4 Summary and Final Remarks . . . 80

5 A Baseline GA for a WSRP 82
5.1 Introduction . . . 82
5.2 Tailored Genetic Algorithm for WSRP . . . 83
5.2.1 Indirect Chromosome Encoding . . . 83
5.2.2 Genetic Operators . . . 85
5.3 Experimental Study and Results . . . 91
5.3.1 Mutation Operators Performance with Different Rates . . . 92
5.3.2 Crossover Operators Performance with Different Rates . . . 95
5.3.3 Performance of Different Population Sizes . . . 96
5.3.4 Comparisons Between WSRP Solution Methods . . . 97
5.4 Summary and Final Remarks . . . 100

6 Adaptive Genetic Algorithm to Solve WSRP 102
6.1 Introduction . . . 102
6.2 Diversity-Based Adaptive GA . . . 104
6.2.1 Search Operators Rates Updates . . . 106
6.2.2 Dynamic Population Re-sizing . . . 108
6.2.3 Crossover Selection . . . 110
6.3 Experimental Study and Results . . . 111
6.3.1 Performance of the Adaptive GA in Time . . . 112
6.3.2 Effect of the Adaptive Control on the GA . . . 113
6.3.3 Baseline GA vs. Adaptive Control Methods . . . 119
6.3.4 Computational Results of Adaptive Control Methods . . . 122
6.4 Summary and Final Remarks . . . 125

7 Multiple Crossover GA to solve WSRP **127**

7.1 Introduction . . . 127

7.2 Adaptive Multiple-Crossover GA (AMCGA) . . . 129

7.2.1 Multiple-Crossover Framework (AMC) . . . 131

7.3 Experimental Study and Results . . . 134

7.3.1 Performance of AMCs in Time . . . 136

7.3.2 Performance of Different Cycle Sizes . . . 138

7.3.3 Performance Comparison of Different AMCs . . . 140

7.3.4 Using AMC as a Component of AMCGA . . . 145

7.3.5 Results of AMCGA vs.WSRPs Solutions Methods . . . 148

7.4 Summary and Final Remarks . . . 150

8 Conclusions and Future Work **152**

8.1 Introduction . . . 152

8.2 Review of Contributions . . . 153

8.2.1 Selecting genetic operators to maximise preference satisfaction in WSRP . . . 153

8.2.2 A Baseline GA for WSRP . . . 155

8.2.3 Diversity-based Adaptive GA . . . 156

8.2.4 Adaptive Multiple-Crossovers GA . . . 158

8.3 Future Work . . . 161

Bibliography **163**

Appendices **185**

A Acronyms and Abbreviations **186**

B Selecting Genetic Operators Results **187**

C Baseline GA Results **199**

D Multiple Crossover GA Results **205**

List of Figures

1.1 A WSRP example, where the number of workers (W) is 3 and the number of visit assignments (V) is 15. 2

2.1 An example of time-conflicts occurrences in a WSRP solution for one worker. Conflicting visits (grey with stripes) and non-conflicting visits (black solid). 13
2.2 Box plots comparisons of WSRP scenarios dimensions, shown as the number of visit assignments V and the number of workers W. 20

3.1 A search space illustration of a minimisation optimisation problem, shown for a contentious search-space. 30
3.2 Examples of the k-point crossover, shown for a single-point crossover 1PX (left) and two point crossover 2PX (right). 33
3.3 An example of a uniform crossover (UX). 34
3.4 Parameter setting classifications in EA's [47]. 37
3.5 Variation calculation for a population P of a size M. 40
3.6 Input-process-output model (IPO) of GA components for WSRP. 55

4.1 An illustration of a WSRP chromosome representation (a), for an instance with 9 visit assignments and 3 workers (b). 59
4.2 Effect of TCR on the initial population, shown for SSGA (blue stripes bars) and GenGA (grey solid bars) with the diversity percentage. 67

4.3 Effect of TCR on the GA dependency on WSR by using the gap to the smallest average value of μ, shown for SSGA (blue and red bars) and GenGA (grey and green bars). 69
4.4 Hard constraints violations occurrences in the best solution found by SSGA*, SSGA, GenGA* and GenGA, as reported in the last iteration. . . 71
4.5 Performance comparison between different rates. Each line represents the relative gap between the best known rate, mutation rate P_m (left plot) or a crossover rate P_c (right plot), corresponds to the problem. 75
4.6 Performance comparison between different genetic operators. Each line represents the relative gap between the best known solution, produced by a mutation operator (left plot) or a crossover operator (right plot)., corresponds to the problem. 76

5.1 An indirect chromosome encoding scheme illustration. 84
5.2 A PMGreedyX crossover illustration. 88
5.3 A PMFCX crossover illustration. 90
5.4 Performance comparison between mutation operators. Each line represents the relative gap between the best known solution corresponds to the problem set and a solution by a mutation operators (left plot) or a mutation rate P_m (right plot). 93
5.5 Performance comparison between crossover operators. Each line represents the relative gap between the best known solution corresponds to the problem set and a solution by a crossover operator (left plot) or a crossover rate P_c (right plot). 94
5.6 Performance comparison of different population sizes. Each line represents the relative gap between the best-known solution corresponds to the problem set and a solution by a population size M. 96
5.7 Average computation-time (in seconds) by GA, VNS and MIP DECOMP, on problem sets A to D and for GA and MIP DECOMP on problem sets E and F. 99

6.1 Graph illustrations of the dynamic population re-sizing approaches, (a) expanding and (b) shrinking. 109
6.2 Change in cost values $f(s)$ over time (in seconds) shown for methods IGA,AGA, GADP, AGADP, RCAGA, RCGADP and RCAGADP. 112
6.3 Box plots comparison of the average crossover rates P_c range, shown for methods AGA (blue), AGADP (red), RCAGA (brown) and RCAGADP (black). 114
6.4 Box plots comparison of the average mutation rates P_m range, shown for methods AGA (blue), AGADP (red), RCAGA (brown) and RCAGADP (black). 115
6.5 Box plots comparison of the average population sizes M range, shown for methods GADP (blue), AGADP (red), RCGADP (brown) and RCAGADP (black). 116
6.6 Change in diversity, shown for methods GA, AGA, GADP, AGADP, RCAGA, RCGADP and RCAGADP. 118
6.7 Average computation-time (in seconds) used in solving WSRP instances shown for methods IGA (black), AGA (blue), GADP (red), AGADP (green), RCAGA (brown), RCGADP (purple) and RCAGADP (yellow). . 124

7.1 Adaptive multiple crossovers GA (AMCGA). 130
7.2 Dissimilarity percentage calculation between two individuals. 133
7.3 An illustration of application rates calculations, in an AMC, shown for 2 crossovers and 3 cycles. 135
7.4 Change in cost values $f(s)$ over time (in seconds) shown for methods AMC_f, AMC_d and AMC_h. 137
7.5 Total number of best solutions produced by different rewarding mechanisms (AMC_f, AMC_d and AMC_h). 138
7.6 Average computation-time (in seconds) produced by AMC variations with different cycle sizes υ. 139

7.7 Average application rates values used under different rewarding mechanisms (AMC_f, AMC_d, and AMC_h). 142
7.8 Box plots comparisons of the average application rates range shown for crossovers 1PX (blue), 2PX (red), UX (black), HX (green), FCX (grey) and PMFCX (brown). 143
7.9 Total number of best solutions while using AMCGA, produced by different cycle sizes. 145
7.10 Average computational-time (in seconds) produced by AMCGA. 146

List of Tables

2.1 WSRP objectives and constraints 12
2.2 Notations for WSRP 14
2.3 Weights values for WSRP 17
2.4 Problem features for 42 home health care problem instances. 19
2.5 Cost-values $f(S)$ and computational-time Cpt (in seconds) for WSRP instances, produced by the MIP solver. 21

3.1 An example chromosome of a vector of assignments. 29
3.2 Evolutionary characteristics considered in works related to scheduling and routing of workforce. 52

4.1 Operators and parameter settings used in the experiments 64
4.2 Performance comparison of crossover operators grouped by category (routing, scheduling, cost-Based) under a mutation operator. 73
4.3 Performance comparison between the best combinations of operators (mutation - crossover) 74
4.4 Selected settings for 35 WSRP problem sets used. 77
4.5 Cost-values $f(S)$ and computational-time Cpt (in seconds) for WSRP instances, produced by SSGA and GenGA selected operators, crossover probabilities P_c and mutation probabilities P_m. 78

5.1 Preliminary parameter settings for the baseline GA 91
5.2 Recommended settings for WSRP problem sets. 97

5.3 Cost-values $f(S)$ and computational-time Cpt (in seconds) for WSRP instances, produced by the GA, VNS and MIP DECOMP. on all 42 WSRP problem instances. 97
5.4 Average relative gap to the best-known solution obtained by GA, VNS and MIP DECOMP. 98

6.1 Initial parameter settings for WSRP problem sets (before the run). 111
6.2 Statistical results comparison generated by a non-parametric Friedman's test combined with performances metrics 120
6.3 Cost-values grouped by GA methods generated by a non-parametric Wilcoxon sign rank test, where S+ (S -) means that method i is significantly better (worse) than method j within a confidence interval of 95%. 121
6.4 Cost-values $f(S)$ and computational-time Cpt (in seconds) for WSRP instances, produced by the baseline GA (IGA) and six diversity-based adaptive GAs on all problem instances. 123

7.1 Parameter settings for AMC and AMCGA. 135
7.2 Statistical results of the AMCs methods generated by a non-parametric Friedman's test combined with performances metrics 140
7.3 Cost-values $f(S)$ and computational-time Cpt (in seconds) for WSRP instances, produced by IGA, AMC_r, AMC_f, AMC_d and AMC_h. 144
7.4 Cost-values $f(S)$ and computational-time Cpt (in seconds) for WSRP instances, produced by AGA, AMC_d, $AMCGA_{10}$ and $AMCGA_{50}$ method. 147
7.5 Cost-values $f(S)$ and computational-time Cpt (in seconds) for WSRP instances, produced by all WSRP solution methods on 42 instances. . . . 149

B.1 Results of performance comparisons of different combinations of operators SSGA and GenGA 188
B.2 The normalised results for problem set A generated by SSGA* and SSGA 189
B.3 The normalised results for problem set B generated by SSGA* and SSGA 190
B.4 The normalised results for problem set C generated by SSGA* and SSGA 191

B.5 The normalised results for problem set D generated by SSGA* and SSGA 192
B.6 The normalised results for problem set E generated by SSGA* and SSGA 193
B.7 The normalised results for problem set A generated by GenGA* and GenGA194
B.8 The normalised results for problem set B generated by GenGA* and GenGA195
B.9 The normalised results for problem set C generated by GenGA* and GenGA196
B.10 The normalised results for problem set D generated by GenGA* and GenGA197
B.11 The normalised results for problem set E generated by GenGA* and GenGA198

C.1 The normalised results for problem set A 200
C.2 The normalised results for problem set B 201
C.3 The normalised results for problem set C 202
C.4 The normalised results for problem set D 203
C.5 The normalised results for problem set E 204

D.1 Results of AMC_f with different cycle sizes 206
D.2 Results of AMC_d with different cycle sizes 207
D.3 Results of AMC_h with different cycle sizes 208
D.4 Results of $AMCGA$ with different cycle sizes 209

List of Algorithms

1 Basic Genetic Algorithm (GA) 28
2 Time-conflict reduction repair (TCR) 61
3 Worker suitability repair (WSR) 62
4 Cost-Based Uniform Crossover (CBUX) 63
5 Cost-Based Mutation (CBM) 64
6 Cost-Based Greedy Crossover (CGX) 86
7 Partially Matched Greedy Crossover (PMGreedyX) 87
8 Flat-Costs Crossover (FCX) 88
9 Partially Matched Flat-Costs Crossover (PMFCX) 89
10 Flat-Costs-Flip Mutation (FCF) 91
11 Zero-Flip Mutation 91
12 Adaptive Parameter Control 106

Acknowledgements

I am grateful to Allah for the greatest gift of my life, self-belief, that was necessary to complete this thesis. I am also grateful for the help of many who have believed in me as an individual. There are many more than I can list here, but I have done my best to name a few. Family first, my Mother Nawal and my father Menahi who told me I was great before there was any legitimate reason to do so. My uncle, Dr. Ahmad Adas who helped me through the process of achieving my goals. My brother Sami for being the lighthouse that directed me through life. My sisters: Samar, Samiah, and Tomador, for providing me with unfailing support and continuous encouragement throughout my years of study. My fiancé Mujri for his love, care and dedicated efforts which contributed a lot throughout writing this thesis. Special thanks to my dear friends Jenna, Mahetab, Nesmah, and Sahar, with whom I shared this time at Nottingham, your kind words of encouragement brightened up the hard times for which I am grateful every single day.

I would like to express my sincere gratitude to my supervisor Dr. Dario Landa-Silva for all his support and guidance throughout my doctoral study. Also, for his patience, motivation, and immense knowledge. I would like to thank my second supervisor Dr. Per Kristian Lehre and the following academic staff: Dr. Andrew J. Parkes, Dr. Rong Qu, and Dr. Robert John for their comments, which helped me in improving my work. A very special gratitude goes out to the support provided by the administrative staff at the school of computer science, in particular, Deborah Pitchfork and Christine Fletcher.

Thanks to the many fellow researchers at ASAP group: Arturo Castillo-Salazar, Wasakorn Laesanklang, and Rodrigo Lankaites Pinheiro. My thanks and appreciation also go to Dr. Anna Martínez-Gavara, Dr. José Carlos Ortiz-Bayliss, Dr. Raquel Martínez-España and Dr. Shahriar Asta who encouraged me to be a confident researcher.

This research would not have been possible without the financial support provided by Taif University and the Royal Embassy of Saudi Arabia Cultural Bureau in the UK. Finally, I would like to thank the readers, and I hope they find this thesis educational.

الإنسانة التي ربّتني في صغري، وعلمتني وأحاطتني بحنانها والتي دائماً وأبداً أجدها بجانبي في أزماتي،

إلى أغلى من عرفها قلبي، بكل الحب أهديها كلمة شكر .

كلمة شكر وامتنان، إلى صاحب القلب الطيّب والنفس الأبيّة، إلى صاحب الابتسامة الفريدة ،

إلى من حارب وساهم الكثير مِن أجلي .

أتمنى من الله عز وجل أن يعطيكم الصحة والعافية،

شكراً لكم على ما قدمتموه لي من أحاسيس نابعة من قلوبكم ودام الله عزكم ودام عطاؤكم.

Chapter 1

Introduction

This thesis is the product of four years of research on genetic components, applied to the scheduling and routing of the workforce. This first chapter presents a brief introduction of the background, the motivation for carrying out this research and the specific objectives of this thesis. In addition, it discusses the key contributions made to the field and provides an overview of the remaining chapters.

1.1 Background and Motivation

The decision to pursue this area of research emerged after my master dissertation, which involved genetic algorithm (GA) for nurse rostering problem at Riyadh hospital. Solving real-world problems can be a useful approach in facilitating daily functions, especially in the health care context. Equally important is the efficiency of optimisation technologies as solution methods in comparison to the manual approach, cost-wise and quality-wise.

Recently, there has been a renewed interest in the growing population of elderly, or patients with chronic diseases who need care at their homes [14]. Hence, home care related optimization problems are introduced.

One example is the workforce scheduling and routing problem (WSRP) [18, 21]. This problem involves scheduling and routing workers to travel to different locations to visit customers in order to complete a set of tasks or activities. In other words, WSRP is

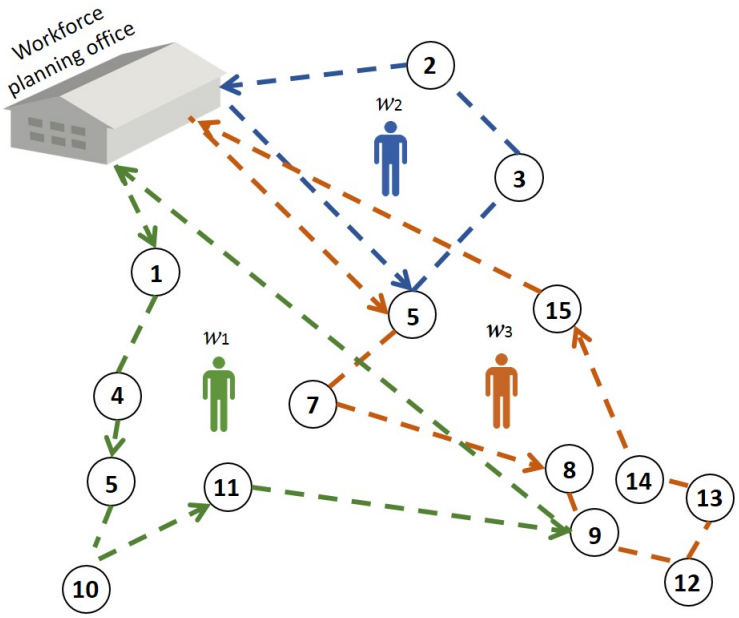


Figure 1.1: A WSRP example, where the number of workers (W) is 3 and the number of visit assignments (V) is 15.

a combination of two combinatorial optimisation problems, personnel scheduling, and routing, which are known to be NP-hard [88]. The scheduling aspect allocates workers to customers to fulfil work demands as well as satisfying *customers' and workers' preferences*. The routing aspect requires generating routes for workers to visit customers across various locations and within given time-windows.

The goal of WSRP considered in this thesis is to generate a daily plan of assignments, i.e. to assign a set of *workers* $W = \{w_1, w_2, \ldots, w_{|W|}\}$ to perform a set of *visits* $T = \{i, j, \ldots, k_{|V|}\}$ for customers at different locations, where V is the number of visit assignments. Each visit requires a service for one or multiple *tasks*. The terms assignment and visit are interchangeably used throughout the thesis.

The WSRP objective is to minimise operational costs and travel distances while attending the additional requirements expressed by customers, workers, and the business corporation. For example, an illustration of a possible plan for a WSRP instance is presented in Figure 1.1. In this example, 3 workers (w_1, w_2, w_3) are assigned to 15 different visits, located in different geographical areas. A path is plotted as a dotted line with a different colour for each worker, i.e., worker 1 is green, worker 2 is blue and

worker 3 is orange. This indicates that worker w_1 is assigned to the set of visits (1, 4, 6, 10, 11, 9). Worker w_2 is assigned to the set of visits (5, 7, 8, 9, 12, 13, 14, 15). Worker w_3 is assigned to the set of visits (5, 2, 3). Note that two workers are assigned to visits 5 and 9, in order to perform different tasks for the same visit.

A key challenge for this combined settings of scheduling and routing is providing a solution that can improve the workforce performances while minimising the operational costs. For this reason, the author's experience on personnel rostering has motivated undertaking WSRPs instances.

A type of WSRP arises in home health care problems (HHC), where *workers* are carers, nurses, doctors, etc., who provide services to one or multiple tasks. Examples include grocery shopping, cooking, helping with bathing, administering medication, monitoring serious illness and unstable health status and injections. *Customers* on the other hand are the patients receiving the health care at their homes. A customer books a time slot, depending on their availability and expects the workers to arrive within that time slot in order to complete a set of tasks [18].

The type of the service required by a customer affects the decision of a worker assignment. For example, if a career is replaced by a higher skilled worker, in this case a nurse; then the range of tasks changes with regards to the worker's skills and the help required. For example, a customer might require a task such as recovering from a surgery [79]. Similarly, if the nurse is replaced by a doctor; different tasks are required. The more specialised the qualifications required, the higher the cost becomes for an assignment. Therefore, pairing a visit to the specified skilled worker can benefit the overall cost of the assignment.

This thesis considers 42 problem instances from six different real-world HHC planning problem, provided by previous research, as WSRP. There are some real-world problems that have similar characteristics to HHC, however, they are not tackled in this thesis, as they are beyond the scope of the study. Such as security guard routing and rostering, worker allocation and maintenance personnel scheduling [18].

1.2 Why Are GAs Used in This Study?

In order to answer the question arise, the nature of WSRP is analysed to identify any challenges that may occur when tackled. It can be claimed that the WSRP is categorised as a logistical combinatorial problem, which involves logistic planning based on formulating, solving and analysing. Other examples are the shortest path problem, the traveling salesman problem (TSP), the vehicle routing problem (VRP) and the job shop scheduling problem (JSSP) [59].

Many logistic problems are considered as NP-hard problems that can arise in real-world settings. If the size of the problem is small, these types of problems are solvable by exact algorithms. However, it is difficult to solve when they are larger in size. This same remark is also applicable to WSRP as presented by [78]. The research has tackled WSRPs instances by using a mixed integer programming (MIP) with decomposition method. Researchers have reported that real-world instances of the WSRP are difficult to solve, especially for large-scale problems that required a considerable amount of computational-time (up to several hours), indicating the need for faster solution methods for this problem.

Another attempt has been made to tackle the same set of instances with a variable neighbourhood search (VNS) [127]. The research has presented problem-specific heuristics that provided good results for WSRPs in less time, indicating the need of using tailored elements in the development of an efficient algorithm for WSRP. Similarly, the hyper heuristics study by [108] which has tackled three types of WSRP problem sets, using a problem-specific adaptive selection of low-level heuristics. This study has proven further the importance of using problem-specific elements when solving the WSRP.

To summarise, the current solutions methods are either time-consuming [79] or require domain-knowledge [108, 127]. Previous research has highlighted the role of using tailored heuristics in reducing issues in WSRP solutions, such as conflicting assignments, workers unavailability at a customer's area and worker/customer preferences. Therefore, it can be argued that the use of a metaheuristic method composed of tailored heuristics can provide good solutions within a reasonable computational-time.

This thesis focuses on Genetic Algorithms (GA), which are considered as a class of the widely known Evolutionary Algorithms (EAs), that are influenced by Darwin's theory of evolution.

According to the natural selection principle "survival of the fittest", living beings compete with each other for resources in every generation and the best have a better chance of survival [109]. With time, after many generations, the individuals become better and much more adapted to the environment.

The GA consists of many components, which can be chosen and combined in various methods, e.g., the data structure used to represent a solution and the evolutionary operators that produce new chromosomes from old ones. According to [175], GAs require an encoded solution of the problem (denoted as a chromosome), that goes through different operators to preserve preferable features from its parents (the concept of natural selection). Genetic information is passed to the next generation with more improvement.

GAs and other EAs are population-based search techniques that discretely explore the solution space. These algorithms are capable of providing multiple solutions in a single run, instead of aiming to improve one solution throughout the run [109].

Different GAs have provided solutions where exact methods are less effective. For example, scheduling problems [4, 143, 180], timetabling [160], JSSP [52] and VRP [133].

Existing research on WSRP has included applying EAs to real-world problems, where scheduling and routing are combined. Examples include the application of GAs [30, 76, 117, 118] and particle swarm optimisation (PSO) [5, 67].

In those works, it is a common practice to attempt to hybridize the algorithm with repair heuristics to overcome infeasibility issues. Still, none of these techniques attempted to identify effective combinations of GA components. It is expected that the GA design for a WSRP might reuse the same components that have been used on the GA applications for scheduling and routing separately [30, 118]. These well-known components might perform well under the WSRP. However, this problem is a highly constrained and is harder to satisfy if a straightforward procedure is used. Thus, the combined nature of WSRP requires more insights with respect to the design elements.

To the best of my knowledge, GA applications of problems involving scheduling and

routing combined [30, 118] have not explicitly investigated problem-specific evolutionary components on a design level. In this thesis, the interest of obtaining solutions for WSRP through GAs motivated the research in understanding the importance of using suitable genetic elements for WSRPs.

An initial review of the literature is conducted on the GAs application on similar problems, in order to classify their behavioural properties when tackling the WSRP. Preliminary results revealed that solving WSRP without any domain knowledge showed poor solutions. Hence, it is suggested by the author of this thesis to use problem-specific components, tailored for WSRP, to minimise the number of constraint violations along with the traveling-costs.

This research has focused on studying GAs to further understand the evolutionary process under a combined setting, particularly for WSRP instances. Additionally, it has investigated different elements that may improve the algorithm effectiveness.

1.3 Thesis Objectives

The main objective of this research is to develop a working GA that solves the WSRP as a real-world problem. With this in mind, the GA must produce good results when compared with established algorithms available in the literature.

To achieve the objectives, the shortcomings of the existing research that tackled WSRPs are addressed as two main questions: 1) How a GA works as a problem solver for WSRP and 2) What are the design elements to be considered to optimise the GA performance for this problem. Most research on GAs have targeted similar question, in some way or another, but in this thesis, the focus is on the combined settings of WSRP.

To answer these questions, the research objectives are divided into two twofold, which are mentioned below along with the steps taken to tackle them as this thesis contributions.

Objective 1 : *Investigating the performance of problem-specific components to solve WSRPs effectively.*

To understand how to effectively employ different combinations of operators when tackling WSRPs, the following steps are undertaken throughout the search:

1. Identify WSRP features, that can be challenging when a simple EA is used (Chapter2).
2. Develop repair heuristics to tackle infeasibility (Chapter 4).
3. Study different combinations of crossover/mutation operators (Chapter 4).
4. Design an indirect chromosome encoding scheme that ensures the feasibility of solutions (Chapter 5).
5. Develop a set of cost-based genetic operators tailored for the indirect chromosome encoding scheme (Chapters 5).

Objective 2 : *Developing an adaptive genetic algorithm customised for WSRPs.*
To increase the GA efficiency and reduce the number of empirical tests, adaptive parameter control methods are explored in this thesis. Thus, the following steps are undertaken throughout the search:

1. Identify WSRP features that contribute to the degradation of the GA performance in later generations (Chapter 6).
2. Study a possible synergy among different crossover operators (Chapter 7).
3. Develop a multiple crossover method that works as one operator (Chapter 7).
4. Develop an online learning model to evaluate operator effectiveness (Chapter 7).
5. Develop a diversity-based adaptive multiple crossovers genetic algorithm (*AMCGA*) that integrates the adaptive rates adjustments approach into the best performing approach *AMC* 7.

1.4 Key Contributions

The thesis contributes to the research in the area of GAs for solving real-world instances of workforce scheduling and routing problems (WSRPs). The contribution also contains the development of an adaptive genetic algorithm to tackle WSRPs. The research ideas explored during the PhD period are outlined below.

1. **Tailored repair heuristics**

   A *time conflict reduction* (TCR) operator is applied to each individual in the initial population in order to reassign visits and reduce the number of time conflicts. The *worker suitability repair* (WSR) operator seeks to improve the suitability of workers for each visit.

2. **Problem specific crossover operators**

   Five different crossover operators were proposed to improve the results by restricting mating to the feasible region only. All operators were evaluated by using an offline tuning process for their parameters.

3. **Problem specific mutation operators**

   Three different mutation operators were proposed to ensure workers' suitability to be assigned to a specified visit. All operators were evaluated by using an offline tuning process for their parameters.

4. **Indirect chromosome representation**

   The encoding is proposed to maintain a flow of feasible solutions. The process assigns suitable workers only and avoids solutions with time-conflicts.

5. **Population re-sizing in an adaptive parameter control method**

   To reduce the baseline GA computation-time, adaptive parameter control is proposed. The dynamic population re-sizing is considered to allow the GA to restrict/expand the search to the required individuals. Also, a random crossover exchange is used to explore the influence of adjusting the crossover function.

6. **On-line learning method for operators rewards**

   Crossovers performances are evaluated during the learning process to alter their application rates during the run. The score is rewarded based on the feedback from the problem-specific application of an online learning mechanism.

7. **Multiple crossover framework**

   Many operators have been proven to perform well under the baseline GA. Thus, an adaptive multiple crossover scheme is proposed to investigate the performance

of a group of operators, that obtained better results than the baseline GA in less computation-time.

8. **Diversity-based adaptive multiple crossovers genetic algorithm**
   To help improve the efficiency of the baseline GA, this method is proposed as a composition of two adaptive aspects, rate adjustment approach and multiple crossover framework. This method allows a deeper insight into the significance of using adaptive components for tackling WSRP.

Overall, the tailored chromosome encoding, repair heuristics, the problem-specific operators, adaptive parameter control and multiple crossover method can provide a starting point for methods comparisons of a new solution for WSRP.

1.5 Publications

The following published articles have resulted from the research work presented in this thesis:

1. A Study of Genetic Operators for the Workforce Scheduling and Routing Problem. Haneen Algethami, Dario Landa-Silva. Proceedings of the XI Metaheuristics International Conference (MIC 2015), pp. 75.1-75.11, Agadir, Morocco, June 2015.

2. Extended Decomposition for Mixed Integer Programming to Solve a Workforce Scheduling and Routing Problem. Wasakorn Laesanklang, Rodrigo Lankaites Pinheiro, Haneen Algethami, Dario Landa-Silva. Operations Research and Enterprise Systems, Series Communications in Computer and Information Science, Vol. 577, pp. 191-211, Springer, 2015.Extended Version of Paper Presented at the 4th International Conference on Operations Research and Enterprise Systems (ICORES 2015), Lisbon, Portugal, January 2015. (uses the method described in Chapter 4).

3. A Genetic Algorithm for a Workforce Scheduling and Routing Problem. Haneen Algethami, Rodrigo Lankaites Pinheiro, Dario Landa-Silva. Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC 2016), pp. 927-934, IEEE Press, Vancouver, Canada, July 2016. (uses the method described in Chapter 5).

4. Selecting Genetic Operators to Maximise Preference Satisfaction in a Workforce Scheduling and Routing Problem. Haneen Algethami, Dario Landa-Silva and Anna Mart'inez-Gavara. Proceedings of 6th International Conference on Operations Research and Enterprise Systems (ICORES 2017), Porto, Portugal, February 2017. (uses the method described in Chapter 4).

5. Diversity-Based Adaptive Genetic Algorithm for a Workforce Scheduling and Routing Problem. 2017 IEEE Congress on Evolutionary Computation (CEC) Special Session on Diversity Preservation Mechanisms for Population-based Metaheuristics, Donostia - San SebastiÃąn, Spain, 2017. (uses the method described in Chapter 6).

Additional Planned Publications

Two additional publications are being considered based on the results presented in this thesis.

1. An Adaptive Multiple Crossover Genetic Algorithm for a Workforce Scheduling and Routing Problem. Submitted as a journal article. The planned publication present three variations of the baseline GA that (uses the method described in Chapter 7).

2. Designing Components of a Genetic Algorithm for a Workforce Scheduling and Routing Problem. This publication presents a gradual development of problem-specific genetic components that are designed to tackle WSRP instances.

1.6 Organisation of the Thesis

Chapter 2 describes the research problem tackled in this thesis, including WSRP's main characteristics, constraints, and objectives. Problem instances used in this thesis are also discussed in this chapter with a review of WSRP solution methods.

Chapter 3 provides a general background on GAs, including design elements, efficiency enhancement techniques, and parameter control. Also, this chapter reviews the literature on WSRP related problems, with respect to the GA design.

Chapter 4 proposes an in-depth study on selecting genetic operators to maximise customers' and workers' preference constraints. Performance matrices are used to assess the quality of each mutation and crossover combination in two variations of well-known GAs.

Chapter 5 proposes an indirect chromosome encoding scheme of WSRP problems with six novel cost-based genetic operators tailored for the indirect chromosome encoding scheme.

Chapter 6 proposes a diversity-based adaptive parameter control method as an extension of the baseline GA. This chapter uses a qualitative approach to investigate if there exists a significant difference between all proposed methods.

Chapter 7 proposes a multiple crossover framework that investigates the effect of crossovers corporations on the GA performances. Performance measurements are used to evaluate each crossover. This study helps to develop a new diversity-based GA customised for WSRP.

Chapter 8 presents a summary of the most important aspects of the research contributions. It also compares the results obtained by the solution methods presented in this thesis with those in the literature. Finally, the potential future research directions are indicated this chapter.

Chapter 2

Workforce Scheduling and Routing problem

2.1 Introduction

This chapter reviews the problem tackled in this research over three sections. The first section portrays the home health care planning problem (HHC) as an instance of WSRP by identifying the hard constraints, soft constraints, objectives, and weights. Then, in the second section, the WSRP instances used in this thesis are explained and discussed. This chapter also reviews different solution methods customised for the WSRP. Finally, a brief summary of this chapter is reported.

2.2 Main Features of WSRP

Table 2.1: WSRP objectives and constraints

Objectives	**Hard constraints**	**Soft constraints**
Minimise travelling cost	Assign all visits	Respect workers area availability
Minimise payment cost	Respect visit time (No time-conflicts)	Respect workers time availability
Minimise penalty cost	Respect max working time per week	Assign preferred workers to visits
	Respect min working time per week	Assign workers to preferred areas
	Assign a qualified workforce	Assign workers with a preferred skill

A list of WSRP objectives and constraints considered in this thesis are shown in Table 2.1, as presented by [79]. An assignment of a worker to a visit must consider worker

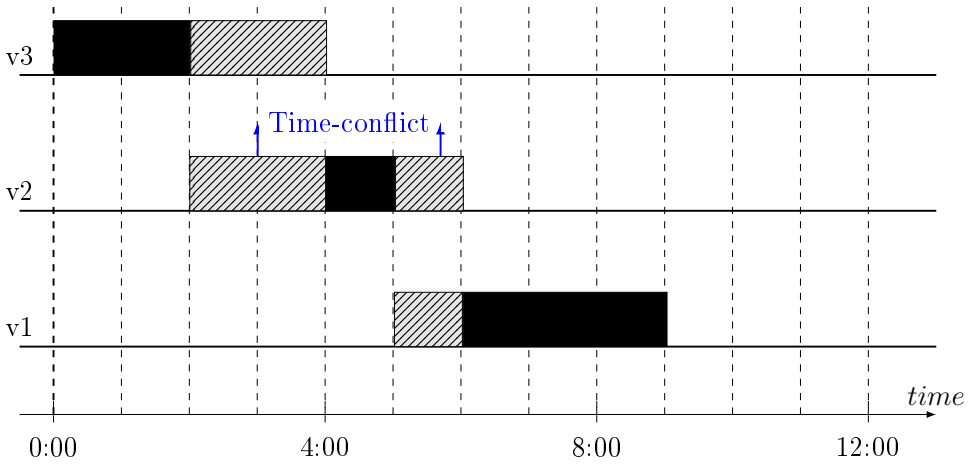


Figure 2.1: An example of time-conflicts occurrences in a WSRP solution for one worker. Conflicting visits (grey with stripes) and non-conflicting visits (black solid).

availability, travelling times, visit duration, skills and qualifications, visit requirements, preferences and costs [101].

A good quality plan for an HHC problem should have a low operational cost as well as assigning workforce. Thus, a solution requires all visits to be assigned according to workers' skills and avoiding time-conflicts with respect to workers' time and area availability. Note that the planning horizon is one day for all instances.

A *time-conflict* occurs when a worker is assigned to visits overlapping in time. Figure 2.1 shows an illustration of time-conflict occurrences among visits for one worker. The conflicting visits are shown as grey rectangles with stripes, while the non-conflicting visits are shown as black solid rectangles. Visits v_1 and v_2 are conflicting from 5:00 $\rightarrow$ 6:00. Visits v_2 and v_3 are also conflicting from 2:00 $\rightarrow$ 4:00. Having time conflicts occurrences result in obtaining infeasible solutions.

Table 2.2 shows the notation used in the following sections with respect to the WSRP objectives and constraints. The details of WSRP features are explained next.

2.2.1 Problem Constraints

To obtain a WSRP plan, the main characteristics are expanded for the problem aspects and are grouped into three main parts: *workers*, *visits* and *locations*. Each group is associated with different constraints that defines the required solution.

Table 2.2: Notations for WSRP

Symbol	Description
Sets	
W	Set of workers
T	Set of visits
Q	Set of skills
Parameters	
V	Number of assignments
$d_{i,j}$	Distance between visit i to a visit j
p_j^w	Cost to assign worker w to a visit j
r_j	The preference satisfaction score for a visit j
$\lambda_1, \ldots, \lambda_5$	Weights of the objective function
Variables	
$x_{i,j}^w$	Binary decision variable to assign worker w into path between visit i and visit j
y_j	Integer decision variable to determine the number of unsatisfied assignments for visit j (i.e. may require more than one worker)
τ_j^w	Binary decision variable indicating assignment of worker w to visit j conflicting with the other assignments
ψ_j^w	Binary decision variable indicating assignment of worker w to visit j to violate area availability
θ_j^w	Binary decision variable indicating assignment of worker w to visit j to violate time availability
ρ_j^w	Binary decision variable indicating assignment of worker w to visit j with preference penalty

Workers are the set of employees $W = \{w_1, w_2, \ldots, w_{|W|}\}$ that provide services for customers. Workforce-related features are as follows:

1. Each workers w has one skill or more from a set of skills $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$.
2. Each worker has to be available to work on one visit only. Returning to the completed visit is not allowed.
3. Each worker is required to work a fixed number of hours, on a daily basis.
4. One worker might be assigned to more than one visit, henceforth not all workers are assigned. Where $x_{i,j}^w$ is the binary decision variable used to assign a worker w into a path between visit i and visit j.
5. A worker w arrives at the location of a visit j at the requested arrival time, denoted as a_j^w.
6. A worker must be available to work at the time and the geographical area of a visit, i.e. not assigned to any other visits at the same time.
7. A worker may have allocation preferences for a specific area.

Visits are the set of visit assignments $V = \{i, j, \ldots, k_{|V|}\}$, requested by customers, that are performed by workers who hold specified qualifications. Visit-related features are as follows:

1. All visits must be assigned and completed by a set of workers, i.e. all visits must be part of a daily plan.

2. A worker skill is either compatible with a visit requirements or not, thus the relationship is 0 or 1.

3. Each visit might have one or more task, thus a visit must be assigned to a worker or a team of workers.

4. All workers must have the required skills to meet the visit requirement.
   For a visit j, an assignment $x_{i,j}^{w} = 1$ will force $y_j \leq r_j$. Where r_j is the *number of workers* required to make visit j and y_j is an integer decision variable, indicating the number of workers is less than the visit j requirement, where $\sum x_{i,j}^{w} + y_j = r_j$.

5. Each visit must be serviced within a time-window.

6. A customer requesting a visit may have a preference on a specified skill.

7. A customer requesting a visit may have a preference for specified workers.

locations: are the visits' geographical areas, where areas are grouped locations. Location-related features are as follows:

1. A worker must travel from a visit i to a visit j until completing the work of the day. For example, $x_{i,j}^{w} = 1$ means a worker w is assigned to travel from visit i to visit j thus, worker w makes both visits.

2. A worker is unavailable during the *travelling time*, where the travelling time is the time taken to go from a visit i to a visit j.

3. Each worker has specified start and end locations (usually they are the same), for one day plan.

2.2.2 Problem Objectives

The WSRP objective is to minimise operational costs and travel distances while attending the additional requirements expressed by customers, workers, and the business

corporation. A key challenge for this combined setting of scheduling and routing is providing a solution that can improve the workforce performances while minimising the operational costs. Thus, a solution S is a set of assignments to workers in order to make visits. The objective function, on the other hand, includes the *operational cost* as well as the *penalty cost*.

The operational cost is the accumulated cost $d_{i,j} + p_j^w$, where $d_{i,j}$ is the distance travelled between visit i to visit j and p_j^w is the cost of assigning worker w to visit j, i.e. wages, equipment used to deliver a visit plus the journey costs for all workers, as calculated by the service provider in the HHC scenarios.

Since feasibility is not guaranteed in a WSRP solution, the penalty costs are tackled as the accumulated penalty for violations of the constraints presented in Table 2.1.

An assignment of worker w to visit j is made to a path connecting between two nodes and can be written as a tuple $\left(x_{i,j}^w, y_j, a_j^w, \psi_j^w, \theta_j^w, \tau_j^w\right)$. This assignment is composed of binary decision variables indicating violations occurrences on area availability (ψ_j^w), time availability (θ_j^w) and conflicting assignments (τ_j^w). If a visit j violates time availability, then $\theta_j^w = 1$. The same applies to area availability violation where $\psi_j^w = 1$, when violation occurs. Likewise, $\tau_j^w = 1$, if the assignment has time conflicts with the other assignment.

Additional preferences include workers preferring to work in certain geographical areas, customers requiring workers with special skills or preferring certain workers to perform a task. For instance, a customer might prefer a female carer to perform the task. The non-satisfaction of preferences is also included in the penalty cost. Therefore, there are three types of preferences including preferred worker-customer pairing, worker's preferred region, and customer's preferred skills.

There is a degree of satisfaction for these preferences when assigning a worker w to a task j and is given by ρ_j^w which has a value that ranges between $\left[0, 3\right]$. For each assignment, the satisfaction value for each preference ranges between $\left[0, 1\right]$, from not satisfied to satisfied. The satisfaction level is reverted to a penalty by subtracting it from the full satisfaction score, which is $3r_j$ for a visit j.

A weighted sum is proposed to combine the objectives into a single scalar value. The

objective function is written as in equation (2.1), *where* weights $\lambda_1, \ldots, \lambda_5$ are defined in Table 2.3 to establish priority between objectives. Lower priority objectives should have less weight value. The objective function can be written as:

$$f(S) = \lambda_1 \sum_{w \in W} \sum_{i \in T} \sum_{j \in T} (d_{i,j} + p_j^w) x_{i,j}^w + \lambda_2 \sum_{w \in W} \sum_{i \in T} \sum_{j \in T} (3r_j - \rho_j^w) x_{i,j}^w + \lambda_3 \sum_{w \in W} \sum_{j \in T} (\psi_j^w + \theta_j^w) + \lambda_4 \sum_{j \in T} y_j + \lambda_5 \sum_{w \in W} \sum_{j \in T} \tau_j^w \quad (2.1)$$

The objective function, shown in equation (2.1), has two of the four priority levels stated in Table 2.3, that involve a cost related to the geographical regions.

In Table 2.1, assignments outside a worker's available regions are penalised. This is considered as *soft constraints penalty*. However, assignments made in a less-preferred region (but still within the worker's available regions) are penalised. This is considered as *preferences penalty*.

2.2.3 Weights Calculations

Table 2.3: Weights values for WSRP

Objective	Weight	Value
Operational costs	λ_1	$mileage/k$
Preferences penalty costs	λ_2	1
Soft constraints penalty	λ_3	$\lvert V \rvert / 2$
Unassigned visits	λ_4	${\lambda_3}^2$
Time-conflicts	λ_5	${\lambda_3}^3$

The weights associated with each objective are set to values that reflect the difference between the priority levels, as suggested by [18, 136]. The main goal is to maintain a balance of priorities for each instance based on each problem specifications. Thus, the

weights calculations presented in Table 2.3 differ from one instance to another.

Hard constraint violations are not allowed in the WSRP solution. However, time conflicts constraint is more difficult to satisfy. Thus, the highest priority level is given to minimise the conflicting assignments τ_j^w, where the associated weight λ_5 is set to the highest value. In this work, a solution with conflicting assignments is considered to be an infeasible solution.

The second highest priority to be minimised is the unassigned visits y_j where λ_4 is set to be very high, but still less than the value of λ_5. This is due to the service provider requirements for completing as many visits as possible. However, in this thesis, a chromosome representation ensures all visits to be assigned to workers, hence no unassigned visits violations. This is explained later in the next chapter.

In practice, the service provider may ask workers to undertake visits that are outside their time availability and/or geographical region. Thus, the next objective priority is λ_3, given to minimise the soft constraints penalty, i.e., the number of workers with time-availability violations ψ_j^w and the number of workers with area-availability violations θ_j^w. As presented by Table 2.3, the weight values for λ_3, λ_4 and λ_5 are relative to the number of the assigned visits, therefore only assigned visits are violated.

On the other hand, if the service provider could not fulfil the highest preference level; the fourth priority is given to minimise the preferences penalty through λ_2.

Finally, the lowest priority is given to minimise the operational cost through λ_1. This weight value is presented in Table 2.3, where k is the operational cost. Hence, mileage is calculated for the assigned workers only.

Hard constraint violations are always penalised with a larger weight than soft constraints and therefore the total preference penalty cost. Accordingly, $\lambda_3 + \lambda_4 + \lambda_5 > 1$ while $\lambda_1 < 1$. Under those conditions, the objective is shifted to minimise the constraint violations, by moving the $f(S)$ cost-value closer to the feasible region.

2.3 Problem instances

Problem instances from six UK real-world home health care scenarios are used as examples of WSRP. The instances data and weights are available at `https://drive.`

google.com/open?id=0B2OtHr1VocuSNGVOT2VSYmp6a2M (blue setting). This tool also provides alternative weights parameters based on the user specifications. The green setup avoids preferences calculations and the red setup prioritises operational costs over the penalty cost.

In this study, six scenarios were used with seven problem instances each, for 42 instances. Table 2.4 shows the main features of each problem instance: the number of visits, the number of geographical areas (visit locations are grouped into areas), and the number of workers.

Scenario A is considered the smallest, while instances in scenario F are the largest. Problem instances in scenario C are of disproportional nature as the number of workers is much larger than the number of visits.

Table 2.4: Problem features for 42 home health care problem instances.

	A								**B**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	***mean***	**1**	**2**	**3**	**4**	**5**	**6**	**7**	***mean***
Number of Visits	31	31	38	28	13	28	13	26	36	12	69	30	61	57	61	47
Number of assignment	32	31	38	28	13	28	13	26	36	12	71	30	62	57	61	47
Number of Workers	23	22	22	19	19	21	21	21	25	25	34	34	32	32	32	31
Number of Areas	6	4	5	4	4	8	4	5	6	5	7	5	8	8	7	5
	C								**D**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	***mean***	**1**	**2**	**3**	**4**	**5**	**6**	**7**	***mean***
Number of Visits	177	7	150	32	29	158	6	80	483	454	585	520	538	610	611	543
Number of assignment	177	7	150	32	29	158	6	80	516	482	623	558	568	655	657	580
Number of Workers	1037	618	1077	979	821	816	349	814	164	166	174	174	173	174	173	171
Number of Areas	8	4	7	8	6	11	6	7	13	12	15	15	15	15	15	14
	E								**F**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	***mean***	**1**	**2**	**3**	**4**	**5**	**6**	**7**	***mean***
Number of Visits	418	425	462	351	461	301	498	417	1211	1243	1479	1448	1599	1582	1726	1470
Number of assignment	440	440	475	363	484	321	521	435	1238	1267	1564	1510	1706	1690	1847	1546
Number of Workers	243	244	267	266	278	278	302	268	805	769	898	789	889	783	1011	849
Number of Areas	13	14	15	13	15	13	16	14	45	46	54	47	59	44	64	51

2.3.1 Why WSRPs are Difficult to Solve?

Figure 2.2 illustrates a group of box plots for each WSRP problem set, that present the overall total of the number of visits assignments $|V|$ and the total number of workers $|W|$. These plots provide a useful way to visualise the problem size by the vertical distance between the smallest value and the largest value, including any outliers.

As it can be seen from Figure 2.2, the gap between the number of visits and the number of workers increases through the problem sets. Distributing assignments of visits over a set of workers is very significant in WSRP solutions, therefore any surplus between

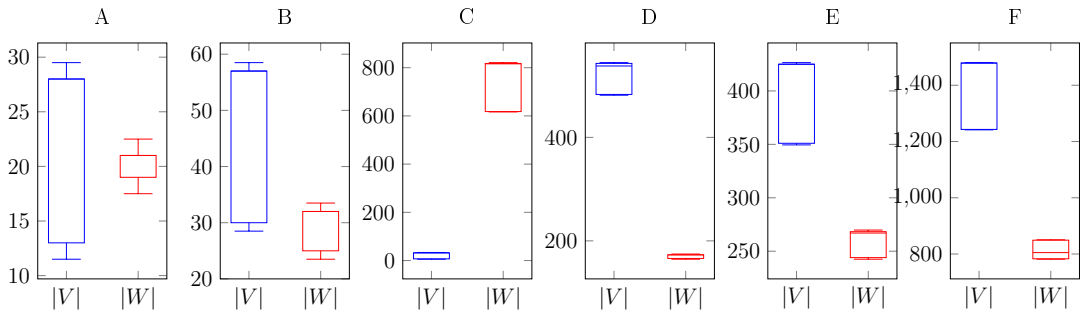


Figure 2.2: Box plots comparisons of WSRP scenarios dimensions, shown as the number of visit assignments V and the number of workers W.

the number of visits assignments and the number of workers results in prioritising one over the other.

In some of the problem sets, the number of visits is larger than the number of workers available. Accordingly, any violations of visitations requirements constraints, such as conflicted visits, increases the value of the penalty function. On the other hand, in problem set C, the number of visits is less than the number of workers; hence, the opposite occurs with violations on workforce-related constraints.

The number of visits assignments and the number of available workforce determine the problem size of a WSRP. A small size WSRP in this thesis has around 50 tasks and 30 workers, as in the case of A and B. Whereas a mix size problem has around 80 visits and 800 workers or more, as in the case of C. A large problem size has more than 400 visits and around 270 workers, as the case of D and E. Problem set F has an average of 1470 visits and 900 workers, which characterise it as even larger than large instances.

Table 2.5 presents an overview of optimal solutions provided by a mixed integer programming (MIP) solver in [78]. The cost-values, i.e. objective function values, is denoted as $f(S)$. The computational-times is denoted as Cpt.

Researchers have reported that real-world instances of the WSRP are difficult to solve [18, 107]. This was proven further by the MIP, where it was unable to provide solutions for 24 instances denoted as n/a.

The problem size affects the algorithm performance, where small instances are less challenging computationally. Hence, MIP has provided optimal solutions for A, B and some of the C problem set. Nevertheless, the MIP ran out of memory for D, E and

F problem sets. Thus, larger problems require more efficient algorithms with problem specific-information.

Note that in [78], unassigned visits constraint is considered as a soft constraint. However, in this work, it is included as a hard constraint; hence, all visits must be assigned. Also, the time-conflict constraint is introduced in this thesis, while the study by [78] avoided conflicts by assigning only available workers with no conflicting assignments.

Table 2.5: Cost-values $f(S)$ and computational-time Cpt (in seconds) for WSRP instances, produced by the MIP solver.

	Problem	$f(S)$	Cpt		**Problem**	$f(S)$	Cpt		**Problem**	$f(S)$	Cpt
	1	3.49	7		**1**	n/a	n/a		**1**	n/a	n/a
	2	2.49	8		**2**	3.15	6		**2**	n/a	n/a
	3	3	14		**3**	n/a	n/a		**3**	n/a	n/a
A	**4**	1.42	5	**C**	**4**	11.15	90	**E**	**4**	n/a	n/a
	5	2.42	1		**5**	12.34	55		**5**	n/a	n/a
	6	3.55	5		**6**	n/a	n/a		**6**	n/a	n/a
	7	3.71	1		**7**	4.3	1		**7**	n/a	n/a
	1	1.7	21		**1**	n/a	n/a		**1**	n/a	n/a
	2	1.75	2		**2**	n/a	n/a		**2**	n/a	n/a
	3	1.72	6003		**3**	n/a	n/a		**3**	n/a	n/a
B	**4**	2.07	25	**D**	**4**	n/a	n/a	**F**	**4**	n/a	n/a
	5	1.82	585		**5**	n/a	n/a		**5**	n/a	n/a
	6	1.62	184		**6**	n/a	n/a		**6**	n/a	n/a
	7	1.79	300		**7**	n/a	n/a		**7**	n/a	n/a

2.4 Solution Methods for Solving WSRPs

Different methods have been applied to different variants of WSRP, such as home care [5], home health care [13], scheduling technicians [67], security personnel routing and rostering [107] and manpower allocation [40]. Solution methods used for WSRPs included: exact, heuristic and metaheuristics.

A number of studies have applied GAs to real-world problems where scheduling and routing are combined. Examples include [30, 118]. In those works, the focus has been on algorithm design in order to obtain good solutions. However, well-known operators and repair heuristics were used to deal with infeasibility issues. Far too little attention has been given to introducing new a tailored GA for WSRP. To date, the impact of using

suitable operators for tackling WSRP instances has not yet been investigated.

Within a WSRP scenario, the visits tasks to be undertaken by workers can be connected or non-connected. Thus, solution methods can be classified based on the constraints used into two main categories: time-dependent tasks constraints and non-connected tasks constraints. Solution methods for these classifications are described next.

2.4.1 Time-dependent Connected Tasks Constraints

Tasks are described to be connected when there is a relation precedence between tasks or when two or more workers are required. This concept is relatively similar to the vehicle routing problem with time windows (VRPTW) with synchronised visits [41].

Connected tasks are tackled in WSRPs by a number of studies. For example, the study by [14] has tackled a daily planning of home care staff scheduling and routing problem, an airline scheduling problem and a forest operation with temporal precedence and synchronisation constraints. The applied CPLEX modelling environment has failed in obtaining solutions for larger instances. Thus, the optimisation based heuristic approach was proposed. The authors found that in the home care application, more workers were used if there was no feasible solution found among the regular workers. Thus, they suggested the use flexible time-windows for the synchronised visits. The heuristic proposed was able to find high-quality solutions quickly.

The use of constructive heuristics to generate a feasible initial solution was widely utilised in the literature. Such as the study by [101], that has tackled a HHC routing and scheduling problem. The CPLEX has failed in finding feasible solutions, thus, an adaptive variable neighbourhood search (AVNS) was proposed with a construction heuristic to generate a feasible initial solution. The AVNS was able to solve eight out of ten instances to optimality and provided near-optimal solutions for the other two within seconds. The study by [101] has shown the critical role of using adaptive methods in enhancing the algorithm performance for this type of problems.

A study by [18] applied a mixed integer programming (MIP) solver to solve WSRPs with connected tasks. Five different datasets, with 375 instances, were taken from the literature that included connected tasks constraint. Out of the 375 instances, the solver

found feasible solutions for 356 of them, with a time limit up to two hours. It was not possible to find solutions, with the same settings, for eleven of the remaining nineteen instances. The study has revealed that tackling instances with 50 activities or more required considerable computational time. Hence, a greedy heuristic tailored for such scenarios was proposed by [17] on only four groups of instances. The greedy heuristic obtained better results in less time when time-dependent tasks constraints were tackled.

A repeated decomposition method with conflict repair heuristic was combined with a MIP solver method was presented by [80], using the same set of instances utilised by [18]. The conflict repair used a heuristic that identified the conflicting paths in the solutions in the sub-problems after the problem decomposition and then repaired them. The solver provided 84 better results than the greedy heuristic mentioned earlier. However, more computational time was required.

2.4.2 Non-Connected Tasks Constraints

Some WSRP scenarios have no regards to time-dependent tasks constraints. For example, a study by [108] presented three types of WSRP problem sets, as a variation of vehicle routing and personnel rostering problems. Including home care scheduling problem (HCSP), routing and rostering of security guards (SPRR) and the maintenance personnel-scheduling problem (MPSP). The only problem that has time-dependent tasks was SPRR. All these instances aim to construct the shortest routes for a workforce, in which they visit customers at different geographic locations while respecting a set of constraints.

These problems were tackled by [108], using an adaptive selection of hyper-heuristics, which aim to determine the requirements for solving particular instances of the aforementioned problems. The heuristics were chosen based on selection probabilities, dependent on a value of the number of new solutions obtained by a heuristic proportional to the time spent. These heuristics change the order of visits within a route or between routes. This method has benefited from using the adaptive selection of heuristics by obtaining solutions for problems that are in-line with WSRP. Therefore, adaptive components are included in this thesis. However, this method is not included in the experimental

comparisons due to major differences in the objectives and the constraints.

Three decomposition methods combined with a MIP solver was presented by [79]. The first method proposed was a geographical decomposition with conflict avoidance that identified the unavailability time for every worker. The second method proposed was a geographical decomposition with conflict repair that used conflicting assignments repair and heuristic assignment. The third method proposed was a repeated decomposition with conflict repair that sets the sub-problems size to 20 tasks instead of 20 locations. The latter has proven to be the best method in terms of solutions' quality and computation time. Nevertheless, larger problem instances with hundreds of visits required considerable computation time (up to several hours), indicating the need for faster solution methods.

The same set of instances were tackled by a variable neighbourhood search (VNS) algorithm in [127], using seven shaking operators as problem-specific neighbourhood heuristics: random flip, area availability flip, time availability flip, preferred worker flip, preferred skills flip, preferred area flip and priority-based flip. In addition, three neighbourhood searches were used: randomised hill climbing, priority-based search and availability violations search. Due to the overusing of problem domain heuristics, the VNS obtained high-quality solutions in less computation time.

In summary, the current solutions methods for WSRPs are either time-consuming or require domain-knowledge incorporation in greedy heuristics in reducing issues in WSRP solutions, such as conflicting assignments, workers unavailability at customers areas and worker/customer preferences. As a result, this thesis proposes a GA, as a population-based metaheuristic, composed of problem-specific operators, to ensure the search ability in providing good solutions within a reasonable computational-time.

2.5 Summary and Final Remarks

As this chapter has shown, WSRP is a combination of two combinatorial optimisation problems, personnel scheduling, and routing, which are known to be NP-hard [88]. Therefore, the WSRP can be considered as a combinatorial optimisation problem categorised as an NP-hard problem.

Assignment of routes to the workforce in order to complete all visits while satisfying all restrictions is a challenging and difficult task. The aim is to minimise the total operational costs along with the penalty-costs, which are the weighted sum of three types: the cost of hard constraints violations (time-conflicts), the cost of soft constraint violations and cost of preference satisfaction penalties.

Applying exact methods was unable to provide solutions for instances with a number of visits ≥ 150. These sets of instances require efficient algorithms with high computational power. Thus, efficient methods, equipped with tailored heuristics, were proposed in order to reduce issues in WSRP solutions.

In the following chapters, a study of the genetic components will be outlined as the methodology applied for this thesis. Then, a review of the current research on genetic components used in the related settings is explored. From these observations, suggested guidelines of how to implement a competent GA to handle these sets of constraints and objectives successfully will be concluded.

Chapter 3

An Overview on Genetic Algorithms

3.1 Introduction

This chapter introduces genetic algorithms (GAs), used in this thesis as problem solvers to tackle the workforce scheduling and routing problem (WSRP). First, a detailed description of the GA elements is stated. Then, limitations of traditional GA approaches are given when tackling real-world problems, and suggestions are made for WSRP-related design enhancements to construct a robust GA for the combined settings.

This chapter also provides an overview of previous work that involves the design of EA components. The review is divided into four parts. The first part outlines different EA concepts known when tackling real-world applications. The second part reviews the EA applications in scheduling problems, followed by applications in routing problems. The fourth part examines the literature related to the WSRP, by reviewing the combined settings of scheduling and routing problems together.

This chapter aims to help the reader to understand the practical and theoretical parts of GAs, and to be acquainted with the terminology that will be used later in the following chapters.

3.2 Genetic Algorithms

Evolutionary algorithms (EAs) are methods that use ideas inspired by biological evolution to obtain solutions to a particular problem [57]. A sample of potential points

(solutions), usually randomly generated, are evolved by using different specific operators, which are biologically inspired. Since the EA makes no assumptions about the underlying landscape of the search space, it often provides a set of appropriate approximating solutions to all types of problems.

The mainstream instances of EAs are genetic algorithms (GA) [33], evolutionary strategy (ES) [33], evolutionary programming (EP) [33]. Example EAs include differential evolution (DE) [154], ant colony optimisation (ACO) [43], memetic algorithms (MA) [112] and particle swarm optimisation (PSO) [75].

Charles Darwin's evolutionary theory inspired the GA concepts and was first introduced by [71]. According to Darwin's theory, individuals in one population battle each other over limited resources. This method is named as the *natural selection* or survival of the fittest. The best individuals, also known by the fittest, have a better chance of survival. Numerous generations pass and individuals show better characteristics over evolution in respect to their environment [57].

A GA is a search heuristic that simulates natural selection and represents an important area in artificial intelligence (AI) [109]. As such, any GA performance is affected by the design elements. Such as the genetic representation of a solution and the fitness function outline to be evaluated [72].

To be considered as problem solvers for combinatorial problems, a GA must be capable of finding solutions *quickly*, *reliably*, and *accurately*. The GA applications have proved their effectiveness, when used for discrete and real-world problems, since GAs use population points and required only minor adjustments [152]. This will be discussed in detail in Section 3.3.

A pseudo-code of a general GA [50] is given below in Algorithm 1. A simple genetic evolution starts by randomly generated individuals (solutions), collectively named as a population (step 1). The process of evolving the population, as an iterative procedure, is called a generation. That is, in each generation, also known as iteration, several steps of a GA are executed, where each generation takes the information passed on from the preceding generation. At the generation setup, each individual represents a solution with respect to the problem. All individuals, in one population, are evaluated and

their fitness is calculated according to the objective value of the problem being tackled (step 3). A new population is formed by manipulating the individuals from the current population, through search operators (steps 4). The algorithm is terminated when either a maximum number of generations has been produced, computation-time limitations or suitable fitness level has been accepted for that particular population [109].

Basic components are implemented to design a GA as a solver: chromosome representation, population initialisation, fitness evaluation, parents selection and search operators [51]. The following sections present a detailed description of GA elements that are used in this thesis.

Algorithm 1 Basic Genetic Algorithm (GA)

1: Randomly create a population P of M individuals
2: **repeat**
3: Evaluate each individual $k \in P$
4: Generate new population P' by recombination and mutation
5: Replace P with P'
6: **until** termination condition is met

3.2.1 Chromosome Representation

A *solution* k is encoded as a vector of variables $(C_a, C_b, \ldots, C_n)$, of a length n. A solution is also known as a *chromosome* or an *individual*, where an element C_i at the i_{th} position in the chromosome is noted as a *gene*. The complete set of genes in the chromosome is denoted as *genotype*. The *phenotype* is the solution characteristics after decoding the genotype [142].

The specifications of a given problem must be understood first to be able to identify the type of the chromosome representation it requires. The solution representation defines how a GA solves a particular problem [142]. This simple data structure is used in every step of a GA, thus selecting a suitable representation scheme affects the efficiency of the search in a GA [109].

Representation schemes are categorised based on the *type of values* or by the *interpretation of values*. The variation of these values depends on the problem tackled, such as binary, octal, hexadecimal, tree or as a permutation [109]. The interpretation of the

values is also dependent on the problem tackled, such as the sequence or the order of these variables. To understand the represented solution, a decoding mechanism is required to transform a chromosome to a candidate solution by interpreting each gene-specific value.

Other representation classifications have emerged based on this process. For example, *direct* and *indirect* representation schemes. The *direct representation* is a vector that directly encodes the solution, e.g. a simple permutation of visits, jobs, operations, etc. [142], in which the relative order of elements is important in decoding a solution.

Table 3.1 illustrates a vector of visit assignments, *where* indexes are ordered visits indicating gene positions and contents of genes are the indexes of workers selected from a set $W = \{w_1, w_2, \ldots, w_{|W|}\}$. In this example, all visits are assigned, but not all workers are used in assignments.

For example, let $k =< 3, 1, 2, 1 >$ be an an arbitrary solution of visit assignments. The value at the i_{th} position holds the assignment information for the i_{th} visit, i.e. the first visit is assigned to worker 3, the second is assigned to worker 1, the third is assigned to 2, and so on. The order of the chromosome determines which visit is served first.

Table 3.1: An example chromosome of a vector of assignments.

Index	1	2	3	4	5	...	n
Content	w_d	w_b	w_d	w_d	w_c	...	w_g

Direct representations are easy to implement and can produce a large number of encoded solutions in the search space. However, constraints are usually not handled by the representation, thus it can produce poor solutions that can complicate the search by getting stuck in an infeasible region[142]. As a result, special encoding schemes are designed to tackle real-world problems.

Using problem-specific knowledge into the design of the encoding scheme enable tackling the constraints at the individual level [4]. This type of representation is known as an *indirect representation*, in which the generated solution is an abstraction, that requires an encoding mechanism prior to the chromosome decoding.

A straightforward procedure is often used to transform a chromosome into a candidate solution [142]. As a result, smaller and simpler search spaces are produced. The level of abstraction must balance between genotype simplicity and fast encoding/decoding pro-

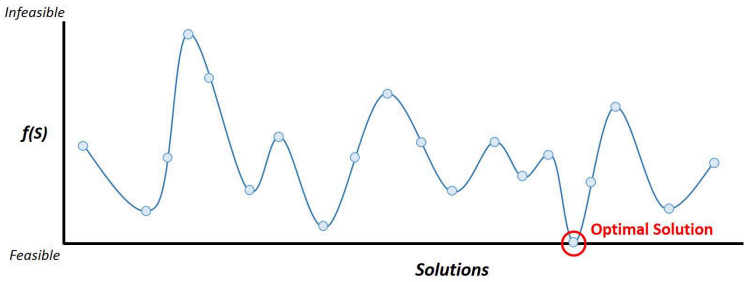


Figure 3.1: A search space illustration of a minimisation optimisation problem, shown for a contentious search-space.

cess. An *indirect representation scheme*, tailored for WSRP solutions, is presented in Chapter 5.

3.2.2 Initialisation and Evaluation

At each iteration, solutions are evaluated by using the problem's objective function $f(S)$, in order to rank all individuals in the fitness landscape based on their quality [109]. Each individual has a *fitness-value* that influence the natural selection process. Based on the problem criteria, the goal is to find the fittest individual, with a smaller or larger fitness-value, than the best so far solution.

A group of M individuals, usually generated at random, to create a population $P = \{k_1, k_2 \ldots, k_M\}$. The aim is to cover the entire search space of possible solutions to the problem tackled. Figure 3.1 illustrates an example of the search space, for an optimisation problem [135]. Red circled points are the best candidate solutions. The fitness value $f(S)$, in this case, is minimised, thus the superiority of one solution over another is having a lower $f(S)$.

The *search space* G is divided into a feasible region and an infeasible region. A *feasible region* G_F includes values that satisfy the problem's constraints, where $\forall x \in G_F$ satisfy all hard constraints. A search space may contain more than one feasible region that has many *local minimum*, where the best solution might exist in that region. A *global minimum*, on the other hand, has the best solution in the entire search space [144]. An *infeasible region*, includes infeasible solutions, where $\forall x \in G_I$ violate at least one hard constraint. Thus, the solution space can be denoted as $G_F \cup G_I = G$, where $G_F \cap G_I = \phi$ [25].

The size of the solution space is the reflection of the problem size, where a solution space with many local optima may require a large population size, in order to have more chance of finding globally optimum solutions [50]. However, more time is required to find the fittest individual. Therefore, heuristics are usually incorporated in the initialisation step to control the algorithm speed [144].

When all the individuals in a population have the same fitness-value; no more improvements are provided (known also as *convergence*). Premature convergence occurs when a population has converged to the same solution, over many iterations, early in the search. Hence, It is important to maintain a diverse set of individuals throughout the evolutionary process [42].

3.2.3 Selection

After evaluating each individual, the fittest subset of individuals, are selected with respect to a selection pressure in order to generate the next generation. The *selection pressure* is identified as the selection degree of the number of individuals [61]. The higher the selection pressure, the more fit individuals are generated. Over the generations, the selection pressure enforces improvements over the quality of individuals selected. After creating a pool of offspring, the new individuals are combined with the original population to maintain the size of the population. This method is repeated until the stopping criterion is met.

Many selection methods are used in different variations of GAs, such as, fitness proportionate selection (roulette-wheel selection) [72], stochastic universal sampling [9], ranking selection [9], and tournament selection [61]. The roulette-wheel selection method consists of selecting individuals stochastically from one generation to create the next generation. The fittest individuals have a greater chance of survival than weaker ones. Hence, fitter individuals have a better probability of survival and weaker individuals are not without a chance.

A tournament selection is used in this thesis because it allows a balance between "good" and "worse" individuals into a population [50]. In a population of a size M, two parents k_i, k_j are selected from parent lists L_1 and L_2, each of size $M/2$, according

to their fitness. The two parent lists are created to maintain a biased selection with different individuals, selected at random from population P.

The best individual of one group is added to L_1 and the best individual of the other group is added to L_2. This process is repeated λ times, where $\lambda = M/2$.

3.2.4 Search Operators

Choosing a suitable operator effects the quality of the generated offspring [162]. Thus, different types of operators used in this thesis are reviewed in the following sections.

Recombination/Crossover operators

After selecting the parents, from $k_1, \ldots, M/2$, $k_i \in L_1$ and $k_j \in L_2$ are combined to produce two offspring, using a *recombination* operator, to maintain good features from the selected parents [69]. The aim is to encourage the search by *exploiting* the solution space [51].

Recombination operators in GAs are noted as *crossovers* with a probability P_c to be implemented. This operator is also known as a *mixer*, where genes from both parents are mixed/exchanged to generate an offspring [162].

Many crossovers are tackled in different variations of GAs, which use different methods to preserve features from parents as explained next.

- k-point crossovers. First implemented by [34] that aims to disrupt the order of a parent chromosome. A number of k-points are selected, such that genes from parents are copied to an offspring before and after a crossing point.

  Figure 3.2 presents an example of k-point crossovers for a permutation of visit assignments, where one/single point crossover (1PX) is on the left-hand side, and two-point crossover (2PX) is on the right-hand side. These are the most common variations of this crossover, and they work as follows:

  - One/single point crossover (1PX). A point is selected at random between 1 and the chromosome length, as shown in the left-hand side of Figure 3.2 with a red line. The genes before this point are copied from one parent and the genes after are copied from the other parent.

– Two-point crossover (2PX). Two points are selected at random, between one and the chromosome length, as shown in the right-hand side of Figure 3.2 with a red line. Alternating segments are swapped to get new off-springs.

The only drawback for this group of crossovers is that they maintain a large number of alleles from the parents, at the same position. Thus, there is a slow drift in the offspring genotype [114].

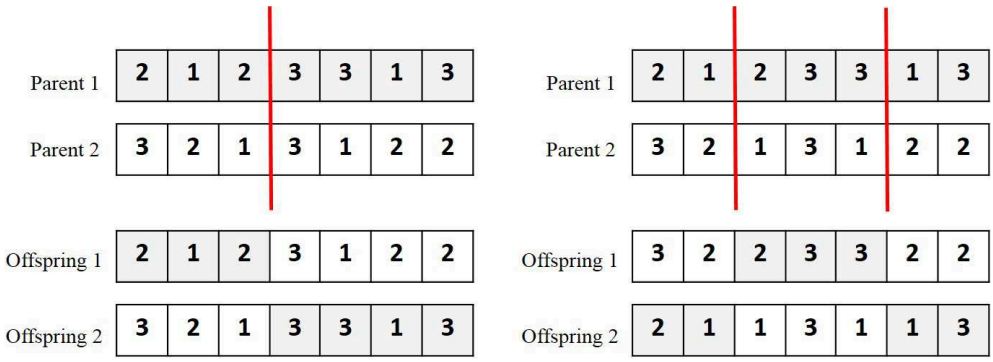


Figure 3.2: Examples of the k-point crossover, shown for a single-point crossover 1PX (left) and two point crossover 2PX (right).

- Uniform crossover (UX). This crossover is first introduced by [2], but most often is credited to [156]. Figure 3.3 shows an example of the UX behaviour for a permutation of visit assignments. Different from k-point crossovers, the number of crossing points are not fixed. Instead, a mixing ratio, usually 50%, is used to choose a uniform random real number u from interval <0,1>. If $u \leq 0.5$; then genes are swapped from parent's individuals, to be copied into offspring. Otherwise, copy the same genes with their current positions from the parent's individuals [22].

  The UX is used independently at each string position, hence this crossover is utilised on a gene level rather than a segment level, as the case in k-point crossovers.

Mutation

Once an offspring is generated, a *mutation* operator is applied with a probability P_m, in order to enable global search and explore new possible solutions (*exploration*) [51]. The mutation operator aims to increase the population diversity by escaping local

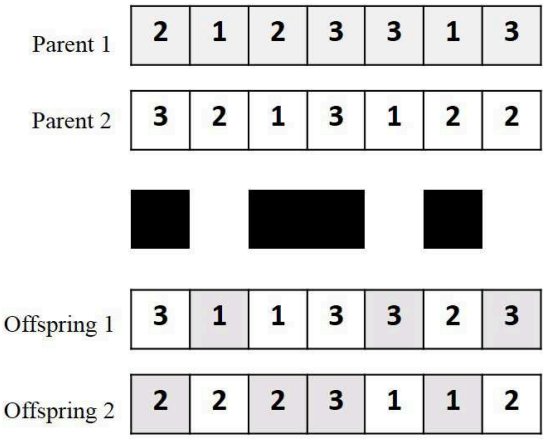


Figure 3.3: An example of a uniform crossover (UX).

optima [144]. The mutation rate P_m is the probability of a mutation function being applied on a chromosome [51]. Different examples of traditional mutation operators, used in this thesis, are as follows.

1. Random swap mutation (RSM). Selects two genes at random and then *swap* genes positions [157]. For example, consider a parent chromosome $p_i = (5, 6, 8, 2, 3, 7, 9, 1, 4)$. If positions 2 and 8 are selected at random, gene 6 and gene 1 interchange their positions then the mutated child chromosome $o_j = (5, 1, 8, 2, 3, 7, 9, 6, 4)$.

2. Inversion mutation (IM). Selects two positions within a chromosome at random and then *rearrange* the elements in the segment between these two positions [56]. For example, consider a parent chromosome $p_i = (2, 3, 4, 5, 6, 1, 7, 9, 8)$. If a segment $(4, 5, 6)$ is selected, the mutated child chromosome $o_j = (2, 3, 6, 5, 4, 1, 7, 9, 8)$

3. Scramble mutation (SM). Selects a segment of elements and randomly rearrange the order of these genes [157]. For example, consider a parent chromosome $p_i = (8, 6, 1, 7, 3, 6, 5, 1, 9)$. If a segment $(1, 7, 3, 6)$ is selected, the mutated child chromosome $o_j = (8, 6, 3, 1, 6, 7, 5, 1, 9)$.

This operator tackles a single individual at a time, while a crossover operator, on the other hand, tackles two (or more) grouped individuals. Problem specific knowledge is usually integrated into mutations because it provides local alterations on solutions [3].

Operators types

The combined work of the crossover and mutation operators is the key strength of a GA. Thereupon, selecting the best combination of operators is essential for a GA capability to solve any given problem.

Many operators have received much attention in the literature, such as uniform crossover and k-Point crossover. However, as GAs are expanding into different areas in optimisation, new operators are constantly being introduced in the literature in order to produce better results by restricting mating to the feasible region only [76].

In this thesis, there are different types of operators. These operators are grouped by the mechanism adopted in each operator as follows.

1. **Cut-points selection from a parent individual**.
   (a) Number of cut points, fixed (1PX) or flexible (UX).
   (b) Random selection, such as 1PX, 2PX and RSM.
   (c) Cost-based selection, where the selected genes affect the overall cost of a chromosome.
2. **Modification on an individual**.
   (a) Reorder parent genes to generate an offspring (swap, exchange, scramble, shuffle, inverse and replace).
   (b) Introduce new values to be inserted into the offspring genes (flip, insertion).

Traditional operators use step (1a) or (1b) with (2a). Most cost-based operators, existing in the literature, use step (1c) with (2a), i.e., selecting specific genes that affect the overall cost of an individual and change their order to generate more cost-effective values. A fully cost-based operator, on the other hand, uses step (1c) and (2b).

In this thesis, two types of cost-based crossover operators are applied. The first type is an extension of the existing cost-based operators that use traditional and cost-based functions, introduced in Chapter 4. The second type is a fully cost-based operator,

introduced in Chapter 5. Mutation operators requires a randomness factor, in order to delay the convergence and encourage the search. Hence, cost-based mutation applied in Chapter 4 is a combination of step (1b) and (2b).

3.2.5 GA Parameters

Distinctive features of a GA define the space of possible solutions, such as the population size M, the type of selection process, the type of search operators used and the values of P_c and P_m. These features are denoted as *algorithm parameters* or *strategy parameters* [145].

Parameter values affect the search direction and its capability to find an optimal or near-optimal solution [47]. However, selecting the right parameters requires many experiments, especially for real-world problems, where adjusting parameter values is an important factor for the algorithm success.

Parameter values are categorised into two groups: **parameter tuning** and **parameter control** as illustrated in Figure 3.4 [47]. Parameter tuning involves finding the best combination of settings among a static set of values *before* the run. These parameters are set by an *ad-hoc* trial-and-error approach of experimenting with different values and selecting the ones that give the best results [46]. Thus, parameter values are set to fixed values throughout the run. This method is used in Chapter 4. However, there are countless combinations of operators with different parameter values. Hence, a large number of tests are required.

Parameter control involves changing in parameter values *during* the algorithm run. Adaptive parameters assess the different behaviour of a GA at different stages of the search. The study by [47] has classified parameter control techniques with regard to different design aspects as follows:

- **Elements to be changed**, i.e. representation, evaluation, operators or rates.
- **Method of change**, i.e. deterministic, adaptive and self-adaptive.
- **Level of change**, i.e., population-level or individual-level.
- **Triggers of change**, i.e., quality of individuals or the diversity of the population.

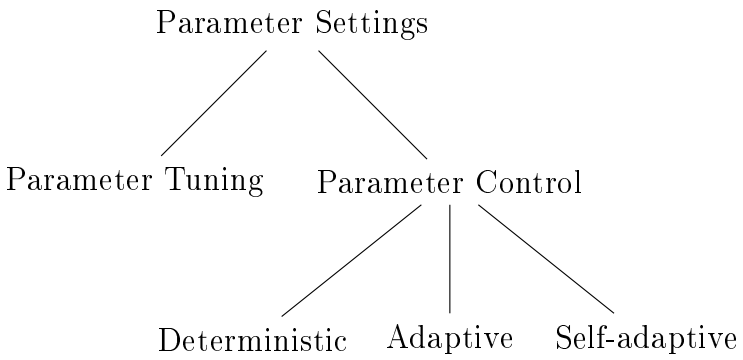


Figure 3.4: Parameter setting classifications in EA's [47].

The deterministic parameter control involves changing parameter values based on a deterministic rule without using any feedback from the search. This method utilises a domain-specific knowledge as **adaptive parameter control** which allows alteration based on a feedback process from the search which determines the value of change. **Self-adaptive parameter control** involves encoding the parameter values into the chromosomes. Hence, the values are influenced by the GA operators.

In the early years of designing EAs, adaptive concepts were used [34] with a few attempts to utilise parameter control for real-world problems [82, 121, 169]. Dynamic population size has been used less frequently than adaptive rates control in GAs. Nevertheless, the effects of changing population sizes have been studied and first introduced by [62]. In the early study by [147], it was claimed that adaptive sizing of GA populations might be a useful extension to the typical GA.

Another aspect of the adaptive method is using different operators during the search. This is because different operators can have different performances. Also, the performance of the selected operator can decline at any point of the search. Hence, using different operators, selected randomly or according to some feedback, has been suggested to have better solutions than using one operator throughout the GA [123].

The focus of this thesis is to change the parameter settings dynamically. To do this, adaptive parameter control methods are proposed in Chapters 6 and 7. The parameters adjusted are population size, operators functions, and crossover/mutation rates.

3.2.6 Replacement strategies

The *replacement strategy* preserves the best-found individuals from each population while weak individuals are eliminated [97]. This method is either biased to the quality of solutions or unbiased to the overall fitness value (random replacement). In this thesis, two replacement strategies are implemented: a *steady state replacement* and *elitist strategy.*

The steady state replacement strategy always replaces a small segment of the population, usually the worst individuals, during each iteration [170]. The two best individuals, out of the two parents and two children, are added back into the population.

An elitist strategy, on the other hand, requires that the original population is replaced entirely, except for a fixed number of best individuals (fit) of each generation. The fittest individuals are carried over to the next generation [97].

3.2.7 Optimising the GA performance

The GA efficiency is evaluated by: the *solution feasibility* and the *population diversity* [25]. Controlling these features can improve the GA performance. This section explains approaches used in this thesis.

Diversity Measurements

This element is referred to the distribution of pairwise differences between individuals in a population. In general, the diversity can be very high in the beginning of the evolution process due to the random initialisation. To ensure convergence, the diversity is reduced drastically in the later generations. The loss of diversity in early stages can cause premature convergence and can lead up to stagnation [150]. Hence, population diversification improves the performance of the algorithm by allowing dissimilar individuals to recombine and then produce better and more diverse solutions. The diversity needs to be measured and controlled throughout the evolutionary process to encourage exploration and exploitation.

Different methods were used in measuring the diversity. Such as, difference-based, distance-based (pair-wise hamming distance), entropy-based, probability-based and ancestry (History)-based [32]. Nevertheless, diversity measures are problem-specific and there is no single measurement that can be applied to all problems [31].

Diversity measures are categorised with respect to the individual representation, on genotype space or phenotype space. The genotype space measurement involves calculating the differences between the genes within all the individuals in a population. The phenotype space involves calculating the differences between fitness values in a population. The composite measure is a combination of both the genotype and the phenotype measurements [31].

The genotype measurement calculates the pair-wise hamming distance (non-similarity) between two integers in a solution [111]. This distance-based measurement is widely used for evaluating the variation of solutions in a population and been proven to be useful for representation schemes such as the space allocation problem [81]. Thus, this measurement is used in this thesis to maintain a balance between having diverse individuals and the algorithm convergence.

$$Variation_{(M)} = \frac{\sum_{j=1}^{n} \frac{D_{(j)}-1}{(M-1)}}{n} \times 100 \tag{3.1}$$

Figure 3.5 show the method for measuring the diversity of a population P for a permutation of visit assignments and works as follows. The level of variation of individuals is computed by calculating the percentage of non-similarity as given in the equation (3.1). Where $D_{(j)}$ is the number of different genes in the j^{th} position and n is the length of an individual in a population P of a size M. In the case of WSRP, $n = |V|$. The diversity value for P is $Diversity(P) = Variation_{(M)}/100$.

Constraint handling methods

In general, solutions are generated at random. As a result, there is a chance of having infeasible solutions that restrict the GA capability, especially when tackling highly constrained problems with a small feasible search space [51].

To improve the solution quality and increase the number of feasible solutions, problem-specific information is used in different GA component as constraint handling techniques while maintaining a diverse population [25].

Constraints handling methods are applied to enforce constraints satisfaction, such

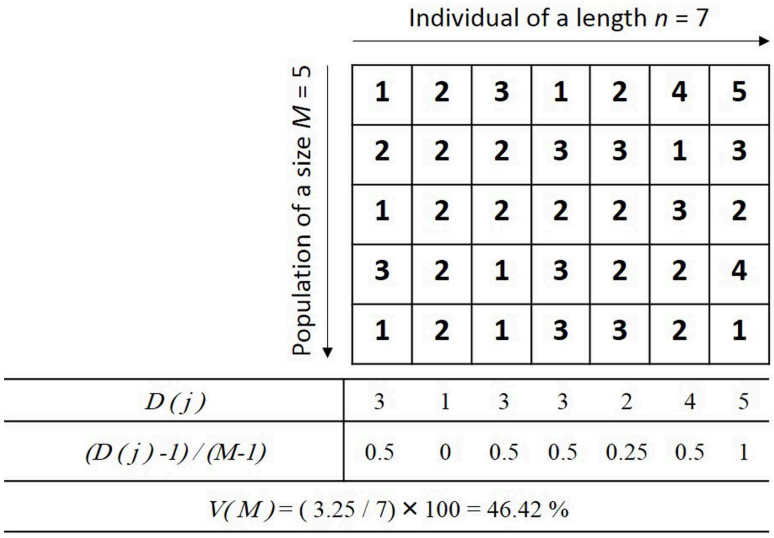


Figure 3.5: Variation calculation for a population P of a size M.

as applying special encoding schemes, preserving feasibility by using repair heuristics, penalty functions and tailored genetic operators. Methods used in this thesis are reviewed in the following sections.

A *repair* heuristic aims to reduce the number of infeasible solutions in a population. This heuristic is very problem-specific that varies according to the problem in hand [5, 66, 178].

A repair operator is recommended to be used for constraint satisfaction problems, to reduce the number of constraints violations [100]. If a constraint is violated (not satisfied), then the chromosome is repaired by either adjusting the value of a randomly selected gene or by using a backtrack repair approach (gene by gene). The latter is computationally expensive because it requires repairing the entire chromosome. Nevertheless, backtracking repair can ensure feasibility of solutions by repairing all constraints violations [100].

A *penalty function* transforms constrained problem to unconstrained problem [25]. Thus, this method is used as a constraint handling technique. An individual is penalised, according to some weight, if found to be infeasible. According to [15], the penalty function approach is well-suited for real-world problems.

Cost-based genetic operators are also used in generating feasible offspring within the

constraint boundaries. To do so, an operator restricts the search to a feasible area only by allowing offspring that have fitness improvements. This method is explained in more details in Chapter 4 and Chapter 5, while using initialisation heuristics.

3.3 Research Review on WSRP Related Problems

According to [61], GAs are used as robust problem solvers that show more or less the same performance over a wide range of benchmark problems. Nevertheless, with the emergent of real-world problems, problem-specific elements are implemented, especially when a specific problem is tackled.

The next sections review EA components in WSRP related problems: scheduling problems, routing problems and the combined setting (scheduling and routing). The focus is to review the following aspects: 1) representation schemes, 2) feasibility handling through search operators and repair heuristics, and 3) parameter control methods.

3.3.1 Scheduling Problems

Many researchers have applied different forms of EA variations on scheduling problems, including resource-constrained scheduling problem (RCPSP), job shop scheduling problem (JSSP), flow shop scheduling problem (FSSP), nurse rostering problem (NRP) and task allocation optimisation problem [45], supply chain management, exam timetabling, scheduling computing tasks, scheduling laboratory equipment, crew scheduling, maintenance/rehabilitation scheduling and talent/project scheduling [110] among others.

To solve real-world problems, tailored design elements were needed to fit the problem domain. For example, different chromosome representations were designed, based on the problem being tackled. In the early stages of the GA developments, binary encoding scheme was used widely [173]. Nevertheless, more optimisation problems emerged with more constraints that required different representation approaches, such as, permutation representation (order of machines) in makespan scheduling problem [105], Gantt-chart representation JSSP [103], priority values [38] and machine assignments [178].

Well-known operators were utilised in various problems. The most used crossover operators in scheduling problems were UX, 1PX, and 2PX. According to [69], the 2PX is

the most effective crossover for RCPSP, as proven later for FSSP [115]. Recent studies have also extended well-known operators for the problem tackled, such as, order crossover (OX) extension that was named linear order crossover operator [52] and 1PX extensions [99]. Note that OX was originally an extension of k-point crossovers.

Several mutation operators were used and were well established in a variety of studies. Such as, RSM [171], SM [157], IM [173] and flip mutation [114]. A shift operator was also used for scheduling problems that select a random job in the current sequence and change that job position to another random but different position [173].

When traditional operators were used, repair operators were needed to maintain feasibility. For example, a study by [66] that has tackled a real-life employee-scheduling problem, with the 2PX operator, and a problem-specific repair operator that resulted in good solutions with faster convergence. A GA-based hyper-heuristic and a memetic algorithm (MA) were used by [29] for scheduling training staff and courses. The MA is an extension of a GA with a local search heuristic, executed after the GA operators. The aim in [29] was to maximise the total priority and minimise the time traveled. The GA has used a direct representation with a random swapping crossover combined and a repair operator. When these methods were compared, the GA outperformed the MA in different instances, indicating that the local search might impede the search operator performance. Using repair operators is computationally expensive as stated in Section 3.2.7. This has encouraged a number of authors to hybridise the GA with problem-specific components. For instance, heuristic techniques were embedded into the GA to generate feasible initial solutions that improved the algorithm efficiency [73, 138].

Real-world applications have incorporated problem-specific information in the representations to restrict the search to feasible schedules. The study by [4, 99] used an indirect encoding of a permutation of the nurses for an NRP. Then, a heuristic-based decoder was also used to construct a roster, subject to various constraints. Another study by [167] investigated different solution representation schemes with different *degrees of directness*. The algorithm with indirect solution representation required less computation time than those with direct representations. It was also shown that indirect representation could achieve better results in scenarios with large search spaces. Another study

by [180] described an encoding scheme designed for machines selection and operations sequencing. The encoding scheme ensured the generation of a good initial population with better performance when compared to previously published algorithms.

Problem-specific operators were also utilised in the literature to maintain feasibility between generations. The study by [76] used a GA for a real-world airline crew-scheduling problem, which included a cost-based uniform crossover that selected genes based on their costs. Genes with lower cost was passed to the offspring, provided that the offspring is feasible. A repair operator was also used to avoid infeasibility. Accordingly, the mutation step was eliminated from the algorithm without any clear guidance on how they tackled the diversity issue. A peak crossover was introduced by [168] in a hybrid GA for RCPSP. This operator proved to be very efficient only for difficult instances of the problem, while simpler instances were not optimality solved.

A number of studies have confirmed the effectiveness of adaptive parameter control for scheduling problems. A study by [99] has used three heuristic approaches to select a crossover operator out of 16 different crossover operators. The aim was to reduce the time considered in tuning the algorithm for a specific crossover. This method selects the best operators with the best solutions. The results have outperformed any previously found solutions. Various GAs have considered similar concepts for selecting a parameter, especially for real-world problems, where time is more crucial. For example, the study by [124] that attempted to evaluate the impact of adaptive GA (AGA) for the flexible job-shop scheduling problem (FJSP). The study has reported that AGA dealt with any fitness landscape change by increasing the population search coverage and delaying the convergence. Thus, AGA was able to overcome the traditional GA shortcomings.

3.3.2 Routing Problems

Some real-world problems have similar characteristics to the WSRP, but still not considered as WSRPs in this thesis. One example is the family of routing problems that require assignment to different locations. Therefore, this section focuses on EAs applications in two variations of routing problems, the travelling salesman problem and vehicle routing problem.

The Travelling Salesman Problem (TSP)

The Travelling Salesman Problem (TSP) is a NP-hard problem. To solve this problem, a path between a list of cities must be found with a minimal cost, and then returning to the city of origin [85].

Similar to WSRP, TSP is considered a cost minimisation problem and requires travelling between different locations. Thus, it can be claimed that TSPs are a generalised setting for the WSRP, as the case with many variations of the scheduling and routing problems [84]. Many EAs have been employed to solve the TSP, as reported by [1, 44, 84, 86, 95, 96].

Scheduling problems have used a representation scheme that was sensitive to the relative order of elements, e.g. resources, jobs or visits. Routing problems, on the other hand, require a representation scheme that is sensitive to the absolute positions of the elements, e.g., in TSP, a city j is listed in the position i if there exist a tour leads from city i to city j [1, 176]. Various chromosome representations were developed for routing problems, such as path [1], ordinal, adjacency and matrix representations [176].

When the initialisation results in infeasible offspring tours, repair algorithms were necessary to assure feasibility. For example, a random key representation was used by [148] for the generalised travelling salesman problem (GTSP), with a local tour improvement heuristic. Solutions were encoded using random keys, with genes filled with random numbers drawn uniformly from [0, 1) that guaranteed feasibility. This process has demonstrated the merit of using a simple encoding scheme, in addition, to repair operators on the performance of the GA.

Based on the representation scheme implemented, different crossover operators were utilised for the TSP, such as 1PX [65]. Nevertheless, this crossover, in particular, was reportedly not suitable for the TSP because the sequence of cities was destroyed, resulting in near to random solutions. Accordingly, there was a large volume of published studies on problem-specific crossover operators for the TSP in order to preserve the order of genes. These operators were grouped with respect to their representation as follows:

- **Position-based crossovers**

  Preserve the absolute positions of the cities from the parent chromosomes.

1. Partially matched crossover (PMX) was developed by [64] for the TSP. This crossover eliminates duplicate genes and recovers all position information from the parent's tours by preserving the absolute positions of the cities when they are copied [130].
2. Cycle crossover (CX) was proposed by [119]. This crossover copies a subset of cities from one parent with their positions from the other. The initial subset of cities is not necessarily located at consecutive positions in the parent tours. Hence, many edges can be broken in the process [130].

- **Order based crossover**

  Preserve the relative order of the cities from the parent chromosomes. One example was the order crossover (OX) [61], which was an extension of the 2PX, that uses random cut points.

The most widely used operators in TSPs were OX, CX, and PMX. It is important to realize that order-based crossover was better than position-based crossovers, as reported by [153]. According to [1, 130], the OX was considered one of the best genetic operators used for the TSP and has been commonly used in GAs [10, 77].

Researchers have also extended the implementation of these crossovers to increase their capabilities, such as the uniform order crossover (UOX) [24] and uniform partially matched crossover (UPMX) [23]. In the study by [16], a multiple travelling salesmen problem (MTSP) was tackled with two-part chromosome crossover (TCX). The OX was used for the first part of the chromosome along with an asexual crossover [20] for the second part of the chromosome. The TCX improved the GA search performance, overcame the representation limitations, and improved solution quality. Hence, tailoring the operators to the chromosome encoding can enhance the GA performance.

Problem-specific crossover operators were also used for TSPs. For example, the study by [164] that presented an edge assembly crossover (EAX). This crossover was the best among other crossovers for large TSP problem sets, when combined with local search method, indicating the benefit of using repair operators in TSPs.

Mutation operators, such as random bit flip; using a binary string, was not applicable

to the TSP. Therefore, swapping and scrambling genes was applied in the literature, such as IM and insertion mutation (ISM) [84]. They also extended these operators, such as multiple extension of the IM, proposed by [137] to explore the search space efficiently. Another widely used mutation operator for routing problems was the 2-opt mutation, proposed by [92]. Every gene is examined to be swapped and then reordered.

A combined mutation method was presented in [35] that included inverted exchange mutation (IEM) and inverted displacement mutation (IDM). The IDM was the best operator among a group of well-known operators. However, it was computationally expensive. This further proves that using a process that is more problem-specific in a mutation can produce better results but affect the algorithm speed.

Tuning parameters require a large amount of time to optimize the GA performance. Thus, adaptive control methods were used for the TSP. Researchers have claimed there was no superior crossover over the others when using GAs for the TSP. Therefore, some studies have investigated the use of multiple crossover processes to combine the work of a group of operators. A random operator selection was first applied by having different performances, at each iteration. A multi-operators GA has been presented in [27] with a variation of 14 crossover operators and 14 mutation operators to solve the TSP with permutation representation. Researchers have reported the effectiveness of using automatic combinations on the GA performance. However, finding the adequate proportion of operators for one combination required more time than using one operator.

The Vehicle Routing Problem (VRP)

This is a combinatorial optimisation NP-hard problem, that aims to obtain optimal routes for vehicles to perform a task or service a number of customers [163]. According to the problem constraints, different variations of VRP have been presented in the literature, including time restrictions, vehicle limitations (capacity and maximum distance to travel) [163]. Variants of VRP studied are capacitated vehicle routing problem (CVRP), distances of a constrained vehicle routing problem (DCVRP), vehicle routing problem with time windows (VRPTW), vehicle routing problem with backhauls (VRPB) and vehicle routing problem with pickup and delivery (VRPPD) [163].

The routing component of the WSRP is related to variants of the classical vehicle routing problem (VRP), in particular, the vehicle routing problem with time windows (VRPTW). Thus, EA applications are reviewed throughout this section to identify the best evolutionary components that can ensure the algorithm performance.

Even though that there are many similarities between VRPs and TSPs, more algorithm specifications were needed due to the variety of constraints in the VRP. For example, most research on VRP has used an indirect representation scheme that needs to be interpreted as a concatenation of vehicle routes as follows. First, a permutation of integers, was created. Then, the chromosome was broken into several different routes, equal to the number of vehicles [19]. Each route is composed of an ordered subset of locations to be visited according to some constraints [132, 158, 172]. The splitting process has allowed the evolutionary operators to transform a solution into a number of permutations, and therefore generate feasible individual routes [134].

According to [181], conventional k point crossover operators were not applicable to VRPs because it produced invalid offspring with have duplicated genes. Thus, most studies have utilised routing crossovers which have initially been developed for the TSP but can also be used for vehicle routing. The study by [133] has suggested that OX has proved to be the most suitable crossover operator for VRP-like problems. Mutation operators, on the other hand, used in VRP, were IM, RM and RSM [74].

A vast majority of the literature has emphasised the importance of hybridising the GA with heuristic repair operators, penalty function [102] as well as problem-specific operators [12, 149] to provide better GA performance [19, 132]. Thus, heuristic-based operators and improvement methods were utilised, which were designed for a specific constraint [65, 84, 159].

Repair operators were usually used for "fixing" un-routed customers in a route or reduce duplicated information. This was the case in [131], that has proposed two tailored crossover operators that divided the arcs selected into two solutions, sequence-based crossover (SBX) and route-based crossover (RBX). The solutions provided by the operators were infeasible. Thus, a repair operator was applied to the offspring to generate a new feasible solution with a local search mutation to split each route of an offspring randomly

into at most three routes. A problem-specific mutation was also used in [12] to reduce the number of routes by utilising Solomon nearest neighbour heuristics [149].

Different studies have explored the idea of using a mix of different crossovers. A study by [134] included a mix of the eight crossovers or the three mutations, where one operator was activated randomly with equal probability and all combinations were ranked according to their performance. The study suggested that this approach has produced better results than using any operator in isolation.

Another study by [123] proposed a novel adaptive multiple crossover population algorithm. Three different routing problems (TSP, CVRP, VRPB) were selected, Crossovers selected were OX, modified order crossover (MOX) [137] and OBX, for the TSP with 2-opt mutation. On the other hand, half uniform crossover (HX) [49] and half random crossover (HRX) [123] were chosen for the CVRP and VRPB with vertex insertion routes mutation[123]. The algorithm obtained solutions in less run-time than classical GA because it made fewer crossovers.

Multiple-operators were also used in VRP with heuristic-based crossovers. For example, the study by [26] for the capacitated arc routing problem that used an MA, which was incorporated with a local search heuristic, and four problem specific self-adaptive multiple crossovers method, based on diversity. The ranking depends on the fittest individuals along with the crowding distance ensured exploring the search space further. This method has improved convergence speed as a result of balancing the diversification methods with operators cooperation, and intensification methods with local search.

3.3.3 The Combined Scheduling and Routing Problem

Many optimisation problems are considered to be a variation of WSRP, starting with the minimisation variation of the generalised assignment problem (MINGAP) [55], resource-constrained scheduling problem (RCPSP) [7] and VRPTW [37].

According to [30], RCPSP is assigning a set of *tasks* to be performed by a finite set of *resources* according to their capacity. Routing problems, on the other hand, focus only on minimising the travelling costs. These descriptions have a similar outline to WSRP, with an addition of designing a minimum cost-plan for workers.

In this section, the problems reviewed include travelling between locations, with objectives similar to the WSRP and constraints such as skill level, working regulations, and customer/worker preferences. Different methods were applied, such as exact, heuristic and metaheuristic methods [116, 139]. The later is the focus of this section.

Some methods used metaheuristics approaches that enforced scheduling workforce and route planning simultaneously [13]. Other approaches used a two-stage method to tackle the scheduling after the routing [6]. Constructive heuristics were used to generate feasible initial solutions in a fast manner. Then, improvement heuristics were used later to improve the quality of solutions [6, 13, 141].

Constructive search and variable neighbourhood search (VNS) were established to tackle a real-world workforce scheduling problem in the study by [140] on the same set of problems used by [30]. The goal was to enhance these methods by using variable fitness function (VFF) with evolving weight. The method improved the quality of solutions and allowed a wider exploration of the search space to consume less time for improvements. Although VFF is a very effective approach that outperforms other approaches, it is a difficult method to design and require more time to adapt the objective weights.

In the study by [93], a permutation chromosome was generated with a combination of constructive heuristics. This method has used OX crossover operator, mutation operators (1-1exchange, 1-0 relocation, 2-Opt exchange, and 2-Opt* exchange) and local search improvements. However, researchers have claimed that there were no real-world home care problems instances, thus they tested their method on test instances based on existing VRPTW benchmarks as variants of HHC problem. On average, the proposed GA was slightly worse than the tabu search, also proposed in that study.

A study by [107] has tackled six benchmark problems of the security personal scheduling problem (SPSP) with two selection mechanisms of hyper-heuristics. They proved that using online learning provided better selections of heuristics than a random method for WSRP. They extended their work in [106] by using a dynamic hyper heuristic on HHC problem with connected visits, that was first presented by [39]. In the later study, they solved the problems with a branch-and-price framework. Both methods provided better performances than the manual method used in practice.

There are a few established applications of EAs for the combined problem. Nevertheless, special attention was given to the methods that solve the problem optimality, considering its nature yet without any considerations of the recombination operators that were not discussed explicitly in the literature.

A study by [30] has used a multi-objective GA (MOGA), on a real-world workforce scheduling problem. The main objective was to maximise the scheduling priority while minimising the travelling time and constraint violation, caused by non-optimal allocation. A permutation of tasks was used along with search operators, without stating their types or rates. A comparison was held between two MOGA and seven weighted sum objective function. The MOGA outperformed the weighted sum methods because it eliminated the difficulty of defining a set of weights. Applying this method to different scenarios might require changing the objective function to a simpler form.

A particle swarm optimisation (PSO) was utilised in the combined settings to focus on solving the problems without any considerations to the operators. The study by [5] tackled scheduling home care workers problem by minimising the distance travelled providing that the capacity and time window constraints of services were not violated. Indirect representation techniques have also been used. However, solutions still required more improvements. Researchers used a local search heuristic in the initialisation step to evaluate the earliest start time priority with minimum distance assignment (ESTP-MDA). This was followed by the application of the heuristic assignment technique for sequential allocation of tasks to each worker based on priority. In order to find the best combination of parameter values, parameter tuning has been performed using Taguchi design of experiments. The PSO has obtained better results than the manual approach and ILOG solver. The integration between PSO algorithms with local search heuristics by the adoption of a local improvement procedure (LIP) technique was successful in making the algorithm more efficient. Nevertheless, applying local search techniques can cause fast convergence to sub-optimal solutions. In order to avoid this problem, establishing effective population diversity strategies was required.

Another attempt to tackle the combined settings by PSO was the study by [67], which tackled the assignment of technicians in British Telecom (BT) with hard constraints such

as job duration, travel times, working times and staff qualifications. This was carried out in order to minimise incomplete work by considering coordinates of the service centers and job locations. The authors proposed a modified version of the traditional PSO by excluding the velocity calculations and using six different neighbourhood topologies for determining the optimal position of the particles. A two-dimensional permutation representation and a repair heuristic were used. The PSO was compared with three different methods [67, 113, 165], and obtained the best solution, highlighting the advantages of hybridising EAs for this type of problems.

A metaheuristic for routing real-world home service transport systems was proposed by [54] with interdependencies between assignments. The method consisted of two stages, creation of walking-routes and optimisation of the transport system. Improvement heuristics were required to find a suitable solution, especially for larger instances.

The patient assignment problem for a HHC problem was tackled in [179] using a GA to estimate the travel times of caregivers. The GA was initialised with the randomly generated matrix chromosomes, OX crossover and RSM mutation with high crossover rates and low mutation rates. The authors have reported a need for improvements. Hence, more problem-specific heuristic approaches should have been presented in this work, e.g. repair heuristics or heuristic-based operators.

Four metaheuristics methods were compared for the multimodal HHC scheduling problem, i.e. different modes of transportation [70] on a single instance set. This study utilised two initialisation methods to generate a solution: a constraint programming (CP) based approach that uses a decomposition method and a simple random construction procedure to seed the metaheuristics. After generating initial solutions, greedy heuristic was used to improve the jobs starting times. Then, the following methods were applied one at a time: VNS, simulated annealing hyper-heuristic, MA and scatter search (SA). MA used problem-specific recombination and mutation operators with high rates. The results showed that population-based meta-heuristics, MA and SA, have obtained the best results. The researchers claimed that the problem-specific operators can be used in similar problems.

3.4 Guidelines for designing a GA for real-world problems

Table 3.2: Evolutionary characteristics considered in works related to scheduling and routing of workforce.

		Scheduling	Routing	Combined
Representation	*Direct Encoding*	[29, 120, 167]	[1, 19, 122]	[30, 67, 179]
	Indirect Encoding	[4, 167]	[93]	[5, 70, 70, 93, 101]
Crossover Operators	*One-Point (1PX)*	[120]		
	Two point (2PX)	[66, 115]		
	Uniform (UX)	[69]	[1]	
	Half-Uniform (HUX)		[123]	
	Partially-Matched (PMX)	[4]	[1]	
	Cycle (CX)		[1]	
	Order Crossover (OX)	[115]	[1, 16, 19]	[93, 179]
	Multiple Crossover		[27, 123]	
	Problem-Based	[76, 168]	[122]	[70]
Mutation Operators	*Random Swap (RSM)*	[171]	[74]	[179]
	Scramble (SM)	[157]		
	Inversion (IM)	[171, 173]	[1, 74, 137]	
	Flip (FM)	[115, 120]	[36]	
	Problem-Based		[19, 93, 123, 132]	[70]
Constraint Handling	*Repair heuristics*	[29, 66]	[83, 129, 148]	[5, 70]
	Penalty Functions	[29, 99]	[129]	[30]
Adaptivity	*Parameters*	[120, 124]		
	Operators	[27, 48]	[1, 123]	[107]

Table 3.2 shows evolutionary characteristics considered in works related to scheduling and routing of the workforce. As shown from the table, the literature has explored various concepts to tackle real-world problems, such as, special representations schemes, different constraints handling methods, problem-specific operators and different adaptive parameter/operators selections. Three characteristics have emerged from the literature that define how to use GAs as problem solvers for real-world problems:

1. Hybridising a GA with heuristics in order to maintain a level of feasibility.

   For example, using initialition heuristic in chromosome representations to build feasible initial solutions, with respect to constraints satisfaction [5, 70, 70, 93, 101].

   Heuristic-based repair operators were also used to "fix" infeasible solution [5, 70]. However, one disadvantage of this operator is that a GA might be dependent on

the repair function to improve the performance, in which it can be computationally expensive. Additionally, heuristic-based repair operators can restrict the search to a local minimum.

2. Genetic operators must be adapted to the problem.
   In the literature, many operators were extended from generic operators or problem-specific operators were designed to suit the problem criteria [70]. The more difficult the problem, the more complicated is the operator process. A well-designed operator can encourage solution space exploration and enhance the performance of the GA.

3. Using adaptive components enhances the algorithm performance and its efficiency. A traditional parameter tuning approach was used for scheduling and routing problems separately with parameters values used for solving the combined settings problem instances [30]. Different parameter values might have provided better results on those instances. An adaptive parameter control method has been proven to improve the algorithm speed, especially if an adaptive operators selection was used with well-designed operators [1, 123].

3.5 How to design a GA to solve WSRPs?

The process of designing a GA for a given practical problem starts by understanding the solution space for that problem. To do so, methods for solution representation, fitness calculations, and parameter values must be identified clearly. If these elements are implemented correctly, they can contribute to the GA efficiency and the quality of the results.

The study by [63] has recommended integrating problem-domain information in the GA design to be able to tackle a challenging problem. Therefore, this thesis further investigates the suitability of a set of problem-specific heuristics and genetic operators in solving WSRPs. The aim is to construct better solutions from partially good WSRP plans, passed from previous iterations. Genetic components used in this thesis are listed below:

1. Two chromosome representations are used: direct representation in Chapter 4 and an indirect representation in Chapter 5.
2. One selection method is used throughout the research: tournament selection in Chapters 4,5,6 and 7.
3. Seven well-known crossovers are used: k-point crossovers, UX, HX, PMX and OX in Chapters 4, 5 and 7 and a CX is used in Chapter 4.
4. Three well-known mutation operators (RSM, IM and SM) are used in Chapters 4 and 5.
5. Two replacement strategy are used: a steady state strategy in Chapter 4 and an elitist strategy in Chapter 5,6 and 7.
6. Two repair heuristics are used in Chapter 4.
7. A penalty function is used in Chapters 4,5,6 and 7.
8. Five problem-specific crossover operators are used in Chapters 4,5,6 and 7.
9. Three problem-specific mutation operators are used in Chapters 4,5,6 and 7.
10. A parameter tuning method is used in Chapters 4 and 5. A parameter control method is used in Chapters 6 and 7.

Note that each component, well known or problem specific will be adapted to specific settings and assumptions of WSRP. Despite the fact that each component is designed individually, all elements combined to act as one algorithm that performs better than independent heuristics [72]. This thesis will provide an important opportunity to investigate design methods of components for WSRP. To improve a GA performance, the following concepts are considered to ensure the GA success.

1. Define each genetic component in the evolutionary process, where individuals evolve through these stages.
2. Understand each genetic component challenges in the combined settings.

3. Ensure a good mixing (crossovers) of selected solutions.

4. Supply diverse and feasible chromosomes, as the outcome of the recombination process and the mutation application.

5. Balance solution quality and convergence times.

Figure 3.6 illustrates an input-process-output model (IPO) of GA components for the WSRP. The inputs are the raw data used as XML files. Each instance is interrupted by a data retrieval tool developed by [128]. This tool is recommended to be used with different solution methods to reduce the time spent on implementing the same set of objectives and constraints.

The GA starts by initialising candidate solutions, for each instance of the WSRP, to go through the evolutionary process. When the stopping criteria are met, the best solution is generated as an output as a one-day plan. As far as the GA process is concerned, selecting suitable elements, which are coherent in the design context, can enhance the performance.

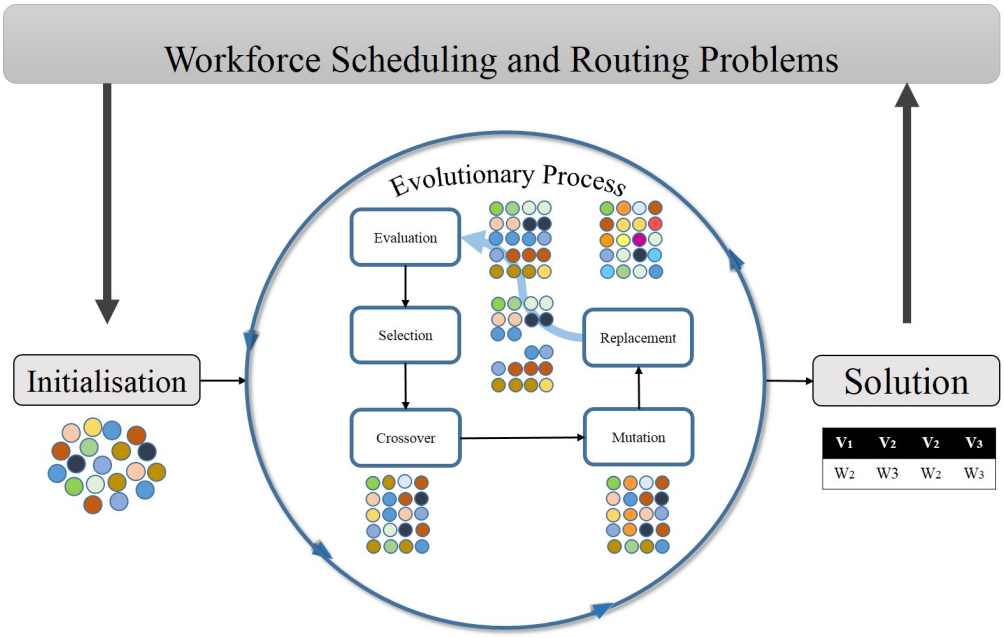


Figure 3.6: Input-process-output model (IPO) of GA components for WSRP.

3.6 Summary and Final Remarks

Basic GA components, contributing to the overall performance, were reviewed in order to outline the background and definitions of methods that will be discussed in later chapters. In general, GAs are used as problem solvers that include different operations, such as generating new offspring and evaluating individual quality.

This chapter explores the implantation of EAs in works related to scheduling and routing of the workforce. In real-world problems, the stochastic approach of a GA does not guarantee feasibility. As a result, various ways have been identified for dealing with constraints. Usually, special encoding schemes, problem-specific operators, repair heuristics, and penalty functions are used. It is suggested by the literature survey conducted in this chapter that heuristic-based approaches and problem-specific information are incorporated in the design of the genetic components. The more complicated is the problem domain, the more is the necessity to apply problem-specific elements.

Using suitable parameters affects the GA components performances, which contribute to the algorithm efficiency. Tuning the parameters is one of the major research questions in the GA community, nevertheless in real-world optimisation problems [47]. Thus, adaptive methods are usually used to make the GA more robust.

This chapter also presents design guidelines to tackle real-world applications, such as WSRP. Numerous studies have specifically utilised algorithms with fewer operators, such as PSO. Even when GA's operators existed in the literature, exact parameters values were usually used without any justification. There was a few of research on what special characteristics should genetic components have to solve WSRPs.

Given to these guidelines, this thesis will present an investigation on different GA elements under the WSRP. Moreover, this thesis will focus on the importance of presenting an effective algorithm. Gathering experimental evidence in the following chapters can provide an understanding of the effect of using different design elements on the algorithm performance for WSRP.

Chapter 4

Selecting Genetic Operators for WSRP

4.1 Introduction

According to the literature survey conducted in Chapter 3, it can be concluded that EA applications, developed for WSRP, focused on developing algorithms only to produce good solutions, without investigating the suitability of operators involved. Nevertheless, repair heuristics have been widely used to deal with infeasibility issues in such problems [5, 67]. Heuristic operators benefit the performance of EAs when tackling WSRP [117].

This chapter conducts an in-depth study of the performances of different combinations of operators and heuristics when tackling instances of the WSRP. Their behaviour is analysed within relatively straightforward implementations of two GA variations. The specific objectives of the research presented in this chapter are:

1. To examine the effect of repair heuristics on reducing constraint violations, in a randomly generated individuals.

2. To understand the relationship between the repair and the genetic operators.

3. To identify the best combination of operators, that maximises the satisfaction of customers and workers preference constraints, in simple GAs.

4.2 Genetic Algorithm and Operators

The implemented GAs work as follows. First, an initial population P of M individuals or one-day plans are *randomly* created, which can result in infeasible solutions with hard constraints violations. Thus, a *time-conflict reduction* (TCR) operator is applied to each individual in the initial population to reassign visits and reduce the number of time-conflicts. Next, a *worker suitability repair* operator (WSR) is applied to reassign workers and ensure the suitability of the worker for each visit. Details of the evolutionary process, operators used, TCR and WSR are given later.

4.2.1 Chromosome Representation

A simple *direct representation scheme* is proposed in this chapter. In this solution scheme, all visits are assigned. Figure 4.1 illustrates an example of solution representation of a vector of integers of a length $|V|$, shown in subplot **(a)**, for a WSRP instance plan, shown in subplot **(b)**. Coloured nodes are workers w_1, w_2, and w_3 while white nodes are visits $1 \ldots 9$, scattered in different locations.

Indexes in the chromosome indicate the set of visit assignments T. The size of the chromosome is the number of assignment $|V|$. For example, the i^{th} gene in the chromosome means the corresponding visit with index $i \in T$.

The number of visits are known but the time separation is needed to be determined by optimisation. Thus, indexes are associated with visits in a non-decreasing order of start time to increase the possibility of obtaining a feasible initial plan. In this way, index 1 is for the visit with the earliest start time and index $|V|$ is for the visit with the latest start time. In case of similar starting times, the algorithm ranks the visits according to their duration [117].

The content of a chromosome is workers assigned to that visit. i.e., one worker per visit. Thus, the worker w at the i^{th} position shows a worker assigned to visit i. For each visit in the vector, a worker w is selected at random from the set of workers W. A worker may undertake more than one visit, and some workers may not be utilised as a part of a particular one-day plan.

The order of the visits based on their starting time can influence a worker's selection

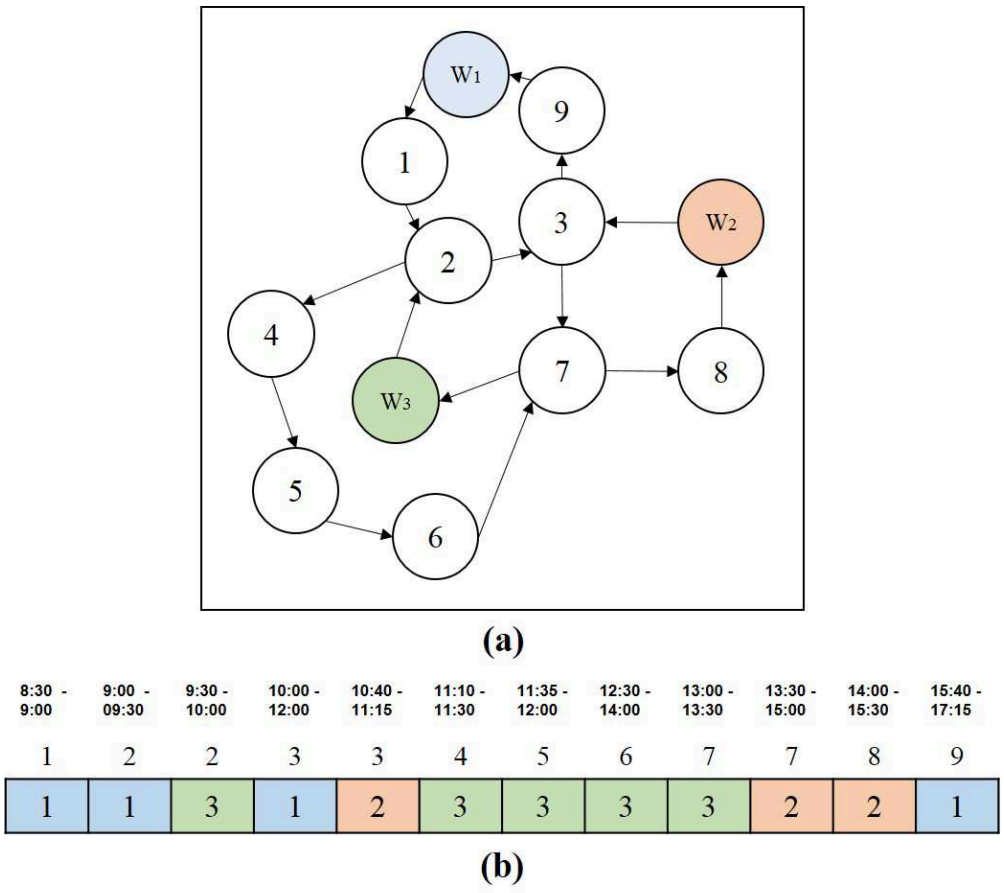


Figure 4.1: An illustration of a WSRP chromosome representation (a), for an instance with 9 visit assignments and 3 workers (b).

(first visit, first assignment). For example, given a chromosome $k =< 3, 1, 2, 3 >$, the first visit with the earliest time is assigned to worker 3, visit two is assigned to worker 1.

4.2.2 Steady State GA vs. Generational GA

Two GAs are implemented in this study: a steady state genetic algorithm (SSGA) and a generational genetic algorithm (GenGA) [170]. Such relatively simple algorithms were selected to analyse the emergent behaviour and performance of the operators on a straightforward GA implementation.

After the TCR, the evolutionary process for each GA is executed as follows. For the *SSGA*, there is only one population P of size M during the whole evolutionary process, where parents are selected, and the offspring are inserted.

Two parents i, j are selected by tournament selection from the parent lists L_1 and L_2, each of size $M/2$. Six different individuals are selected at random from P and split into two groups of three, to create a parent list. The best individual of one group is added to L_1, and the best individual of the other group is added to L_2. This process is repeated until the two lists of parents are complete. Then, for i, j from $1, \ldots, M/2$, parent i in L_1 and parent j in L_2 are combined through crossover, producing two offspring. The next step is to apply mutation operator. A mutation operator is applied with probability P_m to the generated offspring. The mutated individual k' replaces k, regardless of the objective function value. The recombination plus mutation process is implemented on the two parents; the best two individuals out of the two parents and the two children are added into P, while replacing the worse individuals, for M to remain constant. Then, the WSR operator (described below) is applied to infeasible individuals within the population, according to the hard constraints violations.

For the *GenGA* a new population is created at the start of each generation. The recombination plus mutation process is repeated $M/2$ times until the new population P' is complete. At this point, individuals in P' are sorted in non-decreasing order of their fitness. The best 10% of solutions found are never removed from the population. However, the worst 10% of individuals in P' are replaced by randomly generated individuals to introduce diversity into the population. After this, the new population replaces the old one, i.e., $P = P'$. Then, the WSR operator is applied to infeasible individuals. Finally, P' is passed to the next generation.

4.2.3 Repair Operators

Time-conflict reduction (TCR) operator

Algorithm 2 shows the steps for repairing time-conflict violations. Each pair of visits, i and j, are compared to identify any time-conflicts, i.e., the same worker w being assigned to the two visits at the same time (lines 1–6). If there is a time-conflict, worker w is replaced in visit j by another worker $w' \in \Omega^j$, selected from a list of *preferred workers* Ω^j for a visit j (line 10), if that list exists, or selected at random otherwise (line 8).

Because TCR is applied only once on an individual, by removing one pair of visits at

a time, it cannot ensure that all conflicts are removed, but it does reduce their number.

Algorithm 2 Time-conflict reduction repair (TCR)

Require: individual k'
1: **for** $i \leftarrow 1$ **to** $|V|$ **do**
2: **for** $j \leftarrow 2$ **to** $|V|$ **do**
3: Let $w^i_{k'}$ be the worker assigned to visit i in k'
4: Let $w^j_{k'}$ be the worker assigned to visit j in k'
5: Let Ω^j be the list of preferred workers for visit j
6: **if** $w^i_{k'}$ is conflicting with $w^j_{k'}$ **then**
7: **if** Ω^j is Empty **then**
8: Choose a random worker $w' \in W$
9: **else**
10: Choose a random preferred worker $w' \in \Omega^j$
11: $w_{k'} := w'$
12: **end if**
13: **end if**
14: **end for**
15: **end for**

Worker suitability repair (WSR) operator

Algorithm 3 shows the steps for WSR that seeks to improve the suitability of workers for each visit, and it works as follows.

For each visit in an individual, the assigned worker is checked against the skills requirements, maximum hours constraints and time-conflicts, i.e., the hard constraints listed in Table 2.1 (lines 1–3). If the worker does not satisfy these requirements, the operator aims to find another worker who is feasible for that visit (lines 4–6). If no such worker can be found, the operator leaves the original value in place.

The WSR operator cannot ensure that all visits have a suitable worker, but it does improve the overall assignment concerning the constraints. Note that WSR implementation occurs in two stages of the GA: after the TCR, and then again after the mutation operator in the optimisation process.

Algorithm 3 Worker suitability repair (WSR)

Require: An infeasible individual k
Ensure: An individual k'
1: **for** $i \leftarrow 1$ **to** $|V|$ **do**
2: Let w^i be the worker assigned to visit i in k
3: **if** w^i is violating constraints for visit i **then**
4: **for** $w' \leftarrow 1$ **to** $|W|$ **do**
5: **if** w' is suitable for visit i **then**
6: $w^i := w'^i$
7: **end if**
8: **end for**
9: **end if**
10: **end for**

4.2.4 Genetic Operators

This chapter aims to select the best configuration of crossover and mutation operators that can tackle the WSRP within the SSGA and GenGA. Twelve operators are implemented, well-known operators plus cost-based operators tailored for the WSRP.

Ten well-known operators were chosen after a literature survey of operators applied in WSRP-related problems. The operators selected are divided into two groups. One group has five *scheduling operators*: single-point crossover (1PX), uniform crossover, two-point crossover (2PX), half-uniform crossover (HX) and random swap mutation (RSM). The other group has five *routing operators*: order crossover (OX), cycle crossover (CX), partially matched crossover (PMX), inversion mutation (IM) and scramble mutation (SM). Note that all considered crossovers work with chromosomes that are represented as vectors of customer visit assignments.

Cost-based operators have been applied in the literature to produce improved results by restricting mating to the feasible region only [76]. Two cost-based operators have been purposely designed to improve the satisfaction of soft constraints in the WSRP. Nevertheless, other constraints are ignored in the operator's design. Thus, the total cost of the solution can be increased.

Cost-Based Uniform Crossover (CBUX)

Shown in Algorithm 4, CBUX operator works as follows. Each position i for the two parents, corresponding to the worker assigned to visit i, is processed one at a time (line 2). The availability, regarding time and area, of the worker assigned to visit i, is examined for each parent. If both parents have an available worker in that position, their gene is copied (denoted as $\cup$) to the corresponding offspring (lines 5–7). If no parent has an available worker in that position, offspring 1 gets the gene from parent 2, and offspring 2 gets the gene from parent 1 (lines 13–14).

Algorithm 4 Cost-Based Uniform Crossover (CBUX)

Require: parent p_1 and p_2
Ensure: offspring o_1 and o_2
1: $o_1, o_2 \leftarrow$ new empty individuals
2: **for** $i \leftarrow 1$ **to** $|V|$ **do**
3: Let w_1^i be the worker assigned to visit i in p_1
4: Let w_2^i be the worker assigned to visit i in p_2
5: **if** One w_1^i or w_2^i, called w^i, is available for visit i **then**
6: $o_1 \leftarrow o_1 \cup w^i$
7: $o_2 \leftarrow o_2 \cup w^i$
8: **else**
9: $o_1 \leftarrow o_1 \cup w_2^i$
10: $o_2 \leftarrow o_2 \cup w_1^i$
11: **end if**
12: **end for**

Cost-Based Mutation (CBM)

Shown in Algorithm 5, this operator seeks to ensure that workers assigned to visits are among those considered as *preferred workers* for that visit.

As part of the input data in the problem instances considered here, a list of preferred workers is given for each visit as defined by the patients. Then, for position i in the individual, CBM tries to assign one of the preferred workers for that visit i, only if the one already assigned is not a preferred worker (lines 4–6).

Algorithm 5 Cost-Based Mutation (CBM)

Require: individual k
1: Choose a random mutation point $i \in \{1, \ldots, |V|\}$
2: Let w^i be the worker assigned to visit i in k
3: Let Ω^i be the list of preferred workers for visit i
4: **if** $w^i \notin \Omega^i$ **then**
5: Choose a random preferred worker $w' \in \Omega$
6: $w^i := w'$
7: **end if**

4.3 Experimental Study and Results

Experiments were conducted to evaluate the performance of different operators, described earlier in this chapter, aimed at satisfying customers' and workers' preference constraints (mentioned in Chapter 2). The goal of these experiments was to select the best suitable combinations of operators, that can be later embedded in a more efficient approach. Each combination was one crossover operator with one mutation operator.

Table 4.1: Operators and parameter settings used in the experiments

	Parameter	Settings
M	Population size	100
	Crossover operators	1PX, 2PX, UX, HX, PMX, OX, CX, CBUX
P_c	Crossover rate	1%, 10%, 50%, 100%
	Mutation operators	RSM, IM, SM, CBM
P_m	Mutation Rate	0.01%, 1%, 10%, 30%
	Running time	5 minutes

GA required a set of parameters to be initialised to represent the GA population. Thus, all combinations were initially implemented using different configurations as shown in Table 4.1 on the problem instances (from A to E) described in chapter 2. However, the algorithms were unable to produce acceptable results for problem set F.

The running time was limited to 5 minutes in the initial experimentation in order to understand the behaviour of the combinations rather than obtaining good solutions. The best performing configurations were selected in further experimentation by increasing the computational times, to compare the performance of the selected configurations. See Appendix B for all results using the above parameter settings.

One of the major issues in the random direct representation was allowing infeasible individuals throughout the search process so that the end result can have individuals with a high number of constraint violations. Thus, reducing hard constraint violations, such as skills required, maximum hours requirements and time-conflicts, was needed to maintain feasibility in WSRP solutions. Thus, additional settings were implemented to examine the effect of the TCR and WSR operators on the quality of solutions.

As GAs are stochastic techniques, it required a number of runs to average the algorithm performance, to reduce the effect of having initial random seed. For each GA, 32 mutation-crossover combinations with 16 different rates were applied, while including or excluding WSR. Hence, $32 \times 16 \times 4$ algorithm configurations were executed 8 independent runs. The best cost value was obtained from each set with the same amount of time. This means a massive amount of computation time but since it was possible to execute parallel runs, the total time was drastically reduced but still accounted for many days. The machine used had a four-core processor with hyper-threading enabled and 16GB of RAM, and the code was implemented in Java.

Since each mutation operator was combined with one crossover operator at a time, to compare the results for each dataset, average values of runs were computed. These values were normalised over all instances for one problem set.

For instance, given the maximum and minimum values among all the results $x = (f(S)_1, \ldots, f(S)_n)$, the normalised value was calculated according to equation (4.1). Where $f(S)_i^{normalised}$ means the i^{th} normalised data. The reasoning behind value normalisation was because there was a different scale of the results for different groups of values for instances in one problem set.

$$f(S)_i^{normalised} = \frac{f(S)_i - Min(f(S))}{Max(f(S)) - Min(f(S))} \tag{4.1}$$

The percentage gap between the best-known values was also calculated according to equation (4.2), to compare a method y performance to a method x performance. Since WSRP was a minimisation problem; the smallest cost value out of all methods was considered as the best cost-value. Every comparison will subtract the best value from

the current value and normalised according to the best value. The absolute value was calculated to avoid negative value.

$$gap = \frac{ABS(y - x)}{x} \times 100 \tag{4.2}$$

In order to compare between different combinations, the following three metrics were used to measure the performance of the combinations of operators:

Dev% Average percentage deviation from the best objective function value (best solution of all the configurations applied).

Best A fraction of instances in a set for which a configuration matches the best objective function value (best solution of all the configurations applied). This performance metric was absolute and can be compared across existing results in different tables.

Score A fraction of the instances for which the current method produces better solutions than the other configurations, i.e. 'wins'. This score was calculated as $((q \times (p-1)) - r)/(q \times (p-1))$, where p is the number of configurations compared, q is the number of problem instances, and r is the number of instances in which the $p-1$ competing configurations find a better result. Hence, the best score value is 1, when $r = 0$, and the worst score value is 0, when $r = q \times (p-1)$. This is a relative measure of performance. Hence, these values are only meaningful within a table of results and not across different tables.

4.3.1 Performance of Repair Operators

The purpose of the first set of experiments was to assess the contribution of the repair operators, TCR and WSR, on the overall evolutionary process.

Time-conflict Reduction (TCR)

Figure 4.2 shows the average number of the total time-conflicts after initialisation, in each of the problem instances, without (blue striped bars) and with (solid grey bars) applying TCR. The lower the bar, the better, i.e., the fewer conflicts. The percentages

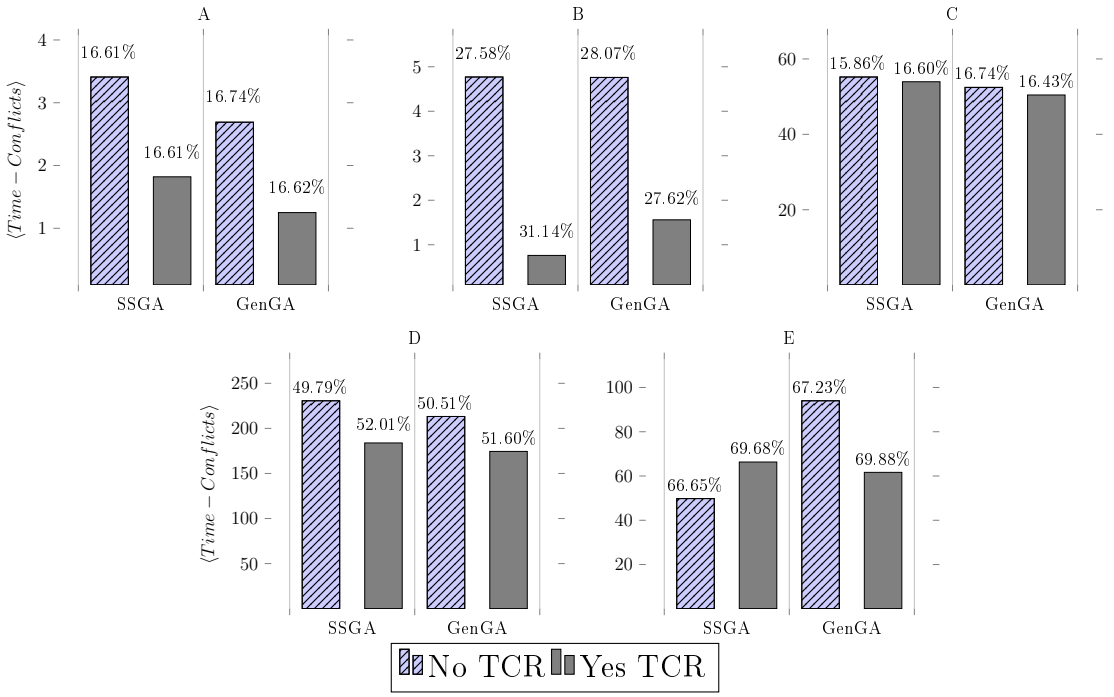


Figure 4.2: Effect of TCR on the initial population, shown for SSGA (blue stripes bars) and GenGA (grey solid bars) with the diversity percentage.

shown on the top of each bar are the average population diversity for all instances (calculates as explained in Section 3.2.7).

Overall, TCR has managed to reduce the number of time-conflicts significantly. Except for problem set C and E. The two GAs have succeeded in obtaining less conflict violations in A, B, and D. When TCR was used, SSGA obtained less violations than GenGA in B instances while GenGA provided better initial solutions for A and D.

In problem set C, there were no significant differences between using TCR or excluding it, for either GAs, with a marginal decrease of less than 0.04% when TCR was used. In the case of problem set E, SSGA had resulted in more time-conflicts violations with an increase of 24.93% when TCR was used. It can be argued that reducing the number of violations for problem sets C and E was a difficult task, especially when SSGA was applied due to the random initialisation.

Even though TCR was applied on different configurations, it resulted in different results, because of using WSR after the TCR. These results illustrate the GenGA superiority over SSGA while using TCR. A closer examination of the result was required

to understand the different behaviour of the GAs, especially on problem sets C and E.

Generating an initial population was computationally expensive when TCR was applied. On average, for SSGA, the computation times to obtain initial solutions increased by 0.54% for A, 18.23% for B, 16.52% for D and 20.06% for E while there was an increase in computation-time for C with random initialisation. For GenGA, the times to obtain initial were increased by 17.02%, 32.59%, 2.18%, 18.84% and 17.88% for all instances sets respectively. This result was expected, as TCR aims to find a better assignment of workers when time-conflicts arise. The more time spent on reducing the violations, the better the solution quality. Nevertheless, on problem set E, SSGA has required more time when TCR was used that resulted in more time-conflicts violations.

Since WSR was still applied for both cases, WSR attempted to reduce the skill requirements in addition to TCR reductions of time-conflicts. For problem set C and E, when TCR was applied, less time was required, as a result of the SSGA early convergence to local minima, which prevented WSR to improve the solution's quality.

The main idea of random initialisation was to generate solutions that cover the search space. Repairing violations on random solutions might restrict the search space to a feasible area only. Therefore, Figure 4.2 show the population diversity, to understand the TCR effects on the population.

On average, the population diversity for SSGA increased by 0.77%, 1.71%, 2.69%, 4.28% and 4.48% in A to E instances respectively. On the other hand, there was an increase of diversity for GenGA of 0.02%, 2.11% and 3.79% in A, D and E instances, with a decrease in percentage in B by 11.31% and C by 1.03% instances.

The loss of diversity has affected the quality of the solutions, especially in the problem set B, where TCR have obtained more violations. Thus, using diversity maintenance in the initialisation method was required to ensure better GAs for WSRP.

Worker's Suitability Repair (WSR)

As explained in Section 4.2, WSR was applied to individuals that present hard constraint violations. To investigate the effect of using WSR, the μ value was calculated as the average total number of WSR used for each instance run, up to the last iteration,

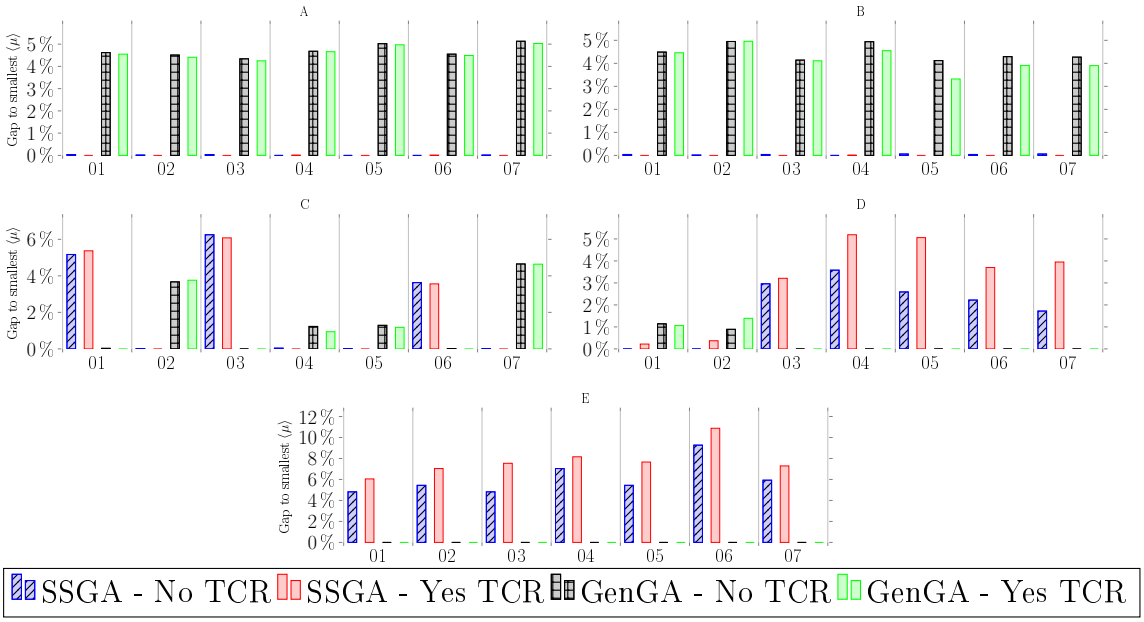


Figure 4.3: Effect of TCR on the GA dependency on WSR by using the gap to the smallest average value of μ, shown for SSGA (blue and red bars) and GenGA (grey and green bars).

i.e., the average total number of repair needed for each instance.

Figure 4.3 presents the relative gap percentages to the smallest μ value for all instances. The bars in the graphs illustrate the values of each method when TCR was not applied (blue and black striped bars) and when TCR was applied (red and green solid bars). The bars are only used to visualise the overall performance of the WSR across all sets. The lower the bar, the better, i.e., the closer to the best-known results.

Regardless of the method applied, there was only a slight reduction of WSR usability when TCR was used as shown in Figure 4.3, compared to when not using it. For example, the absolute differences between SSGAs were: 0.01, 0.03, 0.02, 1.23 and 1.70 for all problem sets. The absolute differences between GenGAs were: 0.07, 0.30, 0.06, 0.06 and 0 for problem sets from A to E.

Note that the there was no difference between GenGAs for problem set E. This means the same amount of repair was used in both algorithms. At the same time, more apparent differences were identified for instances A_1 to D_2. These observations are explored further in the next section.

For smaller problems, i.e., problem sets A and B, the SSGAs utilised less WSR than

GenGAs. This is because SSGA integrates the offspring into the population immediately, as part of the mating pool. Moreover, SSGA only generates two individuals at each iteration. Consequently, WSR was only used to improve the quality of one population.

On the other hand, in GenGAs, the population was reproduced at each generation, with 10% of random individuals. This process has encouraged the search by increasing the diversity. Thus, WSR was used more in GenGA as a result of having more time to apply more improvements.

The effect of having different problem sizes was shown further in Figure 4.3, in the case of the problem set C. When the number of assignments $|V|$ was small, the chromosome size was limited. Hence the SSGAs get stuck in local minima with less WSR usability. As in the case of problem sets C (C2, C4, C5, and C7).

The opposite occurred when $|V|$ was large, the mating pool was more diverse, and SSGA has more time to improve the quality of solutions, yet more violations occur in comparison to GenGA, due to an increase in diversity. Thus, GenGA requires less improvements in larger problems when compared to SSGAs, as in the case of D and E problem instances in addition to C1, C3, and C6.

Another factor in using less WSR in SSGAs was related to steady state replacement process that only repaired new individuals, which was inserted back into the same population that was seeded with TCR. This method was computationally faster in large instances, resulting in more WSR used. This resulted in a drift in the quality of the solutions that led to higher time-conflict constraints violations.

To investigate the overall effect of WSR on solutions quality, the number of hard constraint violations for the best solution found was recorded after a running time of 5 minutes. Each bar in Figure 4.4 represents the average number of hard constraint violations ϕ, skill requirements violations (grey stripes bars) and time-conflicts violations (black solid bars). The lower the bar, the better, i.e., the less violations.

Since maximum hour requirements constraint was satisfied for each problem instance; it was not included in the Figure 4.4.

In scenarios A and B, weights associated with hard constraints violations in the objective function were high enough only to provide feasible solutions, especially when

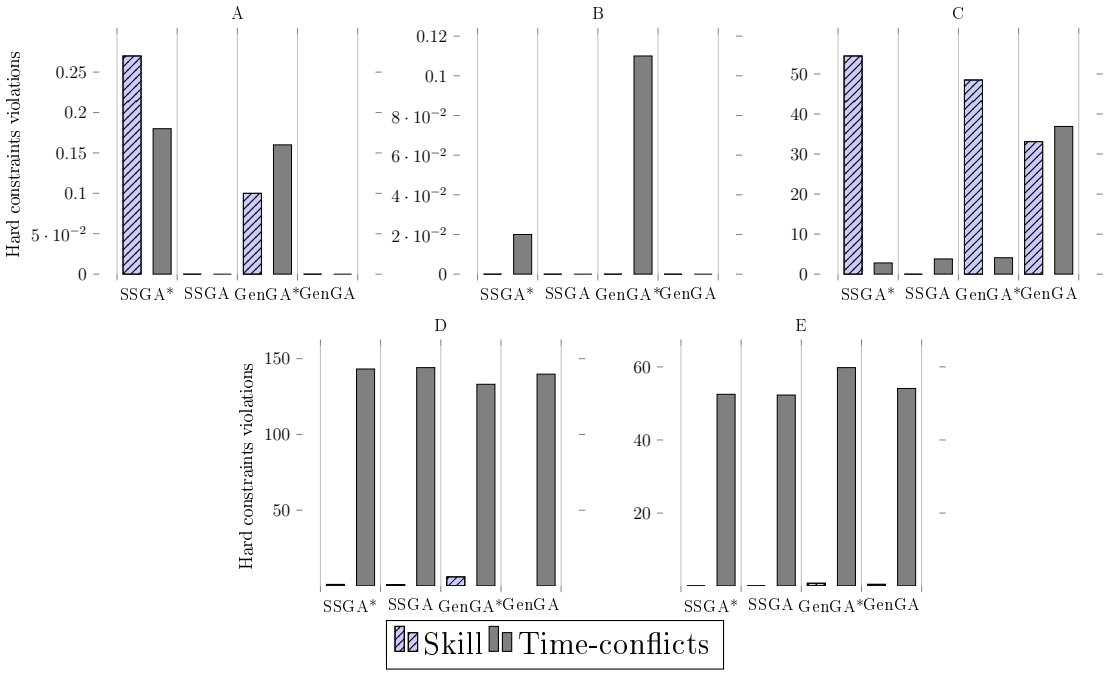


Figure 4.4: Hard constraints violations occurrences in the best solution found by SSGA*, SSGA, GenGA* and GenGA, as reported in the last iteration.

WSR was applied. However, in scenarios C, D, and E, there was more tendency to obtain infeasible individuals with high constraint violations, in particular, time-conflicts.

In problem set C, skill requirements constraint violations were reduced. At the same time, there was a slight increase in time-conflicts constraint violations when using SSGA and a considerable amount of violations while using GenGA. This was also reflected in scenarios D and E as presented in Figure 4.4.

Skill requirements were constraint violations reduced when using WSR. However, time-conflicts seemed to be more challenging, with a slight increase in the problem set D. It appears that WSR prioritised skill requirements constraints over time-conflicts, and therefore both algorithms encountered difficulties to obtain feasible solutions under a limited computational time. Hence, each constraint must be repaired individually.

The results in this section suggest that using repair operators have succeeded in reducing constraint violations on various problem instances, however, more time was needed. Their overall performances were affected by the problem dimensions and the GA design. For example, some instances had obtained more violations on hard constraints

when WSR was used, in particular on time-conflicts, while other violations were reduced. The reason for this was even though time-conflicts have been reduced by the TCR in the initial population. However, these violations were not eliminated.

Utilising generic operators in the evolutionary process have contributed in increasing time-conflicts, even though WSR was used. Hence, a more efficient approach needs to be used to restrict initial solutions to a feasible area only and to seed the GA with better solutions in that respect. Also, the algorithms were limited to only five minutes as a running-time, indicating that the algorithms require more time, especially in larger problems.

In these experiments, the goal was to understand the repair operators behaviour rather than to study the convergence rate under the limited time. Given more time, more WSR will be used to repair a larger amount of constraints violations on those set of instances with additional time for the evolution process.

4.3.2 Maximising Preferences-Constraints Satisfaction

The second set of experiments was designed to select the best combination of the genetic operators that maximise customer/worker requirements and preferences satisfaction. To do so, operators listed in Table 4.1 were examined by statistical analysis to determine their performance as explained next.

Four mutation operators (RSM, IM, SM, CBM) and eight crossover operators divided into three different groups: routing crossovers (OX, CX, PMX), scheduling crossovers (1PX, 2PX, UX, HX) and a cost-based crossover (CBUX). Each of the crossover was combined with one mutation at a time. However, only the best-performing crossover from each group was presented in Table 4.2, while using the performance matrices measurement (explained in Section 4.3).

The values were calculated based on the average preferences satisfaction values for each run. The aim was to identify the best crossovers for each mutation by grouping crossovers based on their category, the best routing operator, the best scheduling operator and the CBUX operator presented here. Each mutation has three crossover operators values, and the best crossover out of the three was highlighted in **bold**. Thus, mutation

Table 4.2: Performance comparison of crossover operators grouped by category (routing, scheduling, cost-Based) under a mutation operator.

	Mutation	RSM			IM			SM			CBM		
SSGA	Crossover	*1PX*	***OX***	*CBUX*	***PMX***	*UX*	*CBUX*	***PMX***	*UX*	*CBUX*	***PMX***	*1PX*	*CBUX*
	Dev%	2.09%	**0.84%**	4.00%	**0.57%**	1.45%	2.98%	**0.73%**	1.31%	3.65%	1.21%	**1.17%**	2.78%
	#Best	0.37	**0.54**	0.09	**0.71**	0.17	0.11	**0.66**	0.20	0.14	0.37	**0.46**	0.17
	Score	0.53	**0.71**	0.26	**0.79**	0.50	0.21	**0.79**	0.53	0.19	**0.63**	0.59	0.29
GenGA	Crossover	*PMX*	***1PX***	*CBUX*	*PMX*	***2PX***	*CBUX*	*PMX*	*UX*	***CBUX***	*PMX*	***UX***	*CBUX*
	Dev%	7.50%	2.31%	**2.23%**	7.92%	3.09%	**2.46%**	6.81%	1.88%	**1.21%**	11.85%	**1.72%**	3.16%
	#Best	0.23	**0.51**	0.40	0.23	**0.51**	0.37	0.31	0.43	**0.54**	0.26	**0.46**	0.37
	Score	0.34	**0.71**	0.56	0.37	**0.61**	**0.61**	0.43	**0.66**	0.63	0.31	**0.70**	0.54

Bold text refers to the best result.

operators were not compared in this table.

The GA was executed using the highest crossover rate $P_c = 100\%$ to ensure crossovers utilisation on the whole population. On the other hand, $P_m = 30\%$ was selected to allow a balance between the search and convergence, i.e. if the P_m value was high; there exists a risk of that a GA might "jump" over a promising area with good solutions. If the P_m value was small; there was a chance to get stuck in local minima.

The differences between the SSGA and the GenGA were highlighted in Table 4.2, where different operators performed better for each algorithm. This indicates that the operators perform differently under different GAs. For SSGA, OX and PMX provided the best scores, with the highest number of best values and the lowest deviation when combined with all mutation operators.

For GenGA, 1PX and 2PX provided the best score for RSM and IM with the highest number of the best values. However, CBUX obtained the lowest deviation for those mutations. With SM, CBUX was selected as the first competing crossover because it has a lower deviation, even though UX has better score value. Additionally, UX provides the best score value with the highest number of best solutions and lowest on deviations among the compared crossovers for CBM. The best combinations were considered in the next overall comparison.

Even though the results were not shown in the table, the CX crossover has obtained the lowest value of the preference satisfaction results when combined with all mutations. This result was caused by the CX process that preserves the relative order of genes in a chromosome [153]. Nevertheless, in this problem, keeping the same order of genes from parents have affected the overall cost and resulted in poor offspring.

Table 4.3: Performance comparison between the best combinations of operators (mutation - crossover)

	SSGA				GenGA			
Procedure	**RSM-OX**	**IM-PMX**	**SM-PMX**	**CBM-1PX**	**RSM-1PX**	**IM-2PX**	**SM-CBUX**	**CBM-UX**
Dev%	**1.44%**	10.78%	3.46%	14.51%	**2.37%**	7.31%	2.77%	5.67%
#Best	**0.40**	0.00	0.31	0.29	**0.46**	0.14	0.34	0.31
Score	**0.72**	0.23	0.62	0.43	**0.67**	0.31	0.63	0.52

Bold text refers to the best result.

Table 4.3 shows a comparison between the chosen combinations from Table 4.2 to identify the best combination for each GA by using the same performance measurement matrices explained above. The best combination was highlighted in **bold**.

The results indicate that RSM–OX and RSM–1PX obtain the highest score and the smallest deviation value among all methods, with the maximum fraction of the best solutions of 0.40 and 0.46 for SSGA and GenGA respectively.

Cost-based methods failed to achieve good results in comparison to the generic operators. The CBUX selects workers based on their availability, as discussed in Section 4.2. Also, CBM has selected workers based on customer's preferences; it was not the best nor the second best based on performances matrices. This result was due to the search space restrictions, that led to an infeasible area, by ignoring other constraints. However, when combined with more generic operators, they generated more diverse individuals with high deviation among all mutations.

The best values obtained by the combinations of operators were compared for each instance. Best combinations for SSGA were: scenario A, SM–PMX; scenarios B and C, RSM–OX and for scenarios D and E, CBM–PMX. Best combinations for GenGA were: scenarios A and C, SM–CBUX; scenario B, RSM–1PX; scenario D, RSM–1PX or IM–2PX and scenario D, CBM–UX. This result further confirm that larger problem instances of WSRPs require more domain knowledge when tackled with a simple GA.

4.3.3 Effect of Using Different Combinations of Operators with Different Rates

Suitable parameter settings ensure better performance of the GA [146]. The best combinations identified for customers'/workers' preference constraints, might not be applicable for SSGA and GenGA with different settings. Therefore, more experiments were

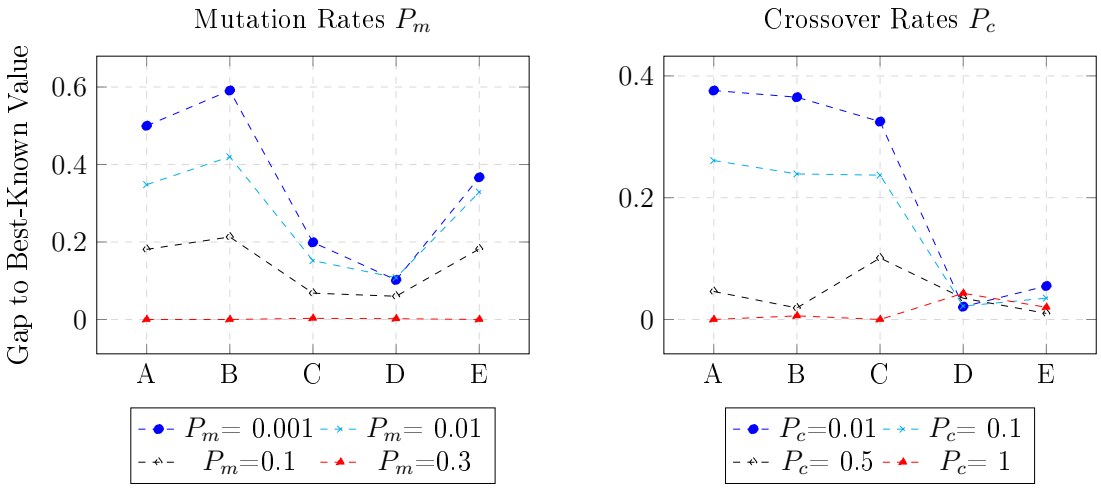


Figure 4.5: Performance comparison between different rates. Each line represents the relative gap between the best known rate, mutation rate P_m (left plot) or a crossover rate P_c (right plot), corresponds to the problem.

conducted to understand the overall effect of different operators.

Each mutation-crossover combination was tested with different crossover rates P_c and mutation rates P_m to establish significantly relevant results in selecting the best combinations of operators.

Figures 4.5 and 4.6 present a clear understanding of a specified rate/ operator performance. Each point in the graphs corresponds to the average gap value for all methods implemented (SSGA*, SSGA, GenGA* and GenGA). The x-axis shows the problem sets while the y-axis shows the average gap percentage to the best average values. The dashed lines in the graphs, joining 5 points, illustrate the average values across all problem sets and were only used to visualise the overall performance.

Figure 4.5 illustrates the performance comparison of all combinations for mutation rate P_m (left-hand plot) and crossover rate P_c (right-hand plot). The lower the line, the better, i.e., the closer to the best-known results.

Mutation rates in Figure 4.5 was drifting towards high mutation rate value (P_m =30%) thus proving high rates to be considerably more effective than a low mutation rate. The reason for this was that high mutation rates increase the diversity and avoid premature convergence, henceforth improving the solutions' quality in the long run.

The right-hand plot shows a comparison of different crossover rates. Similar to

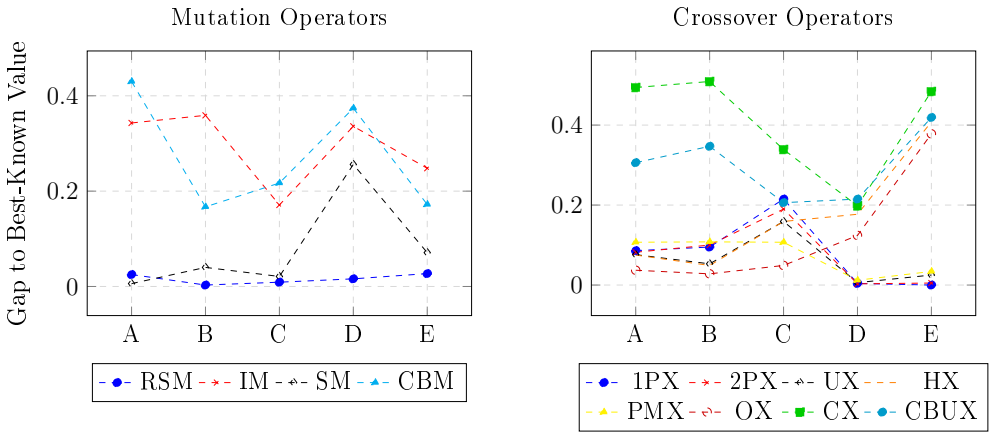


Figure 4.6: Performance comparison between different genetic operators. Each line represents the relative gap between the best known solution, produced by a mutation operator (left plot) or a crossover operator (right plot)., corresponds to the problem.

mutation rates, the lines were aiming towards higher rates (50% and 100%) to enforce recombination; hence better results were obtained. Therefore, high rates, P_m =30% and P_c=100%, were the key factor to better performance regardless of the GA used.

Figure 4.6 illustrates the performance comparison of all combinations of mutation operators (RSM, IM, SM, and CBM) in the left-hand plot, as well as all crossover operators (1PX, 2PX, UX, HX, PMX, OX, CX, and CBUX) in the right-hand plot. The lower the line, the better, i.e., the closer to the best-known results.

Based on the lines left-hand plot, it was clear that RSM and SM succeeded in obtaining the best values in regard of mutation operators. The dashed lines for RSM and SM were relatively similar in all methods except for D, where an apparent gap between RSM and other operators. Reallocating existing genes, in two positions or more, boosts the diversity and henceforth provide better results. However, IM have obtained poor results as a consequence of the inefficient time spent in building an offspring.

The study of the performances of the crossover operators presented in the right-hand plot shows that 1PX, 2PX, and OX achieved the best cost values. There was a defendant evidence that OX was the best crossover operator for smaller instances as a result of its overall process, where a sub-sequence of genes were selected from one parent and preserving the relative order of nodes from the other parent.The genotype disruption has managed to obtain unbiased individuals that were different than their parents.

Larger problems dimensions required more time to build offspring. Hence, they needed faster functions, such as 1PX and 2PX. Those set of instances have less necessity to use problem-specific operators, such as CBUX and CBM, while generic operators succeeded when combined with RSM and SM respectively.

These results prove previous conclusions from Table 4.3. Therefore, these operators were the best combinations to maximise the preferences or for the best overall performance of the GA.

Table 4.4 presents the selected operators and rates for each problem set in WSRP when using SSGA and GenGA. Apparently, when WSR was not used, a high P_c and P_m value was required. No WSR means less time to repair and more time for the operators. Another observation concluded from this table was that different operators were suitable for various instances. This was because of the different operator process that works differently for each instance. For example, problem set uses routing crossover operators PMX and OX while problem set E uses scheduling crossover operators.

Table 4.4: Selected settings for 35 WSRP problem sets used.

		A	B	C	D	E			A	B	C	D	E
SSGA*	*Crossover Operator*	PMX	HX	PMX	1PX	1PX	**GenGA***	*Crossover Operator*	UX	OX	PMX	2PX	1PX
	P_c	1.00	1.00	1.00	1.00	1.00		P_c	1.00	1.00	1.00	1.00	1.00
	Mutation Operator	SM	CBM	SM	RSM	SM		*Mutation Operator*	SM	RSM	RSM	RSM	RSM
	P_m	0.30	0.30	0.30	0.30	0.30		P_m	0.30	0.30	0.30	0.30	0.30
SSGA	*Crossover Operator*	2PX	PMX	PMX	2PX	1PX	**GenGA**	*Crossover Operator*	OX	OX	OX	PMX	2PX
	P_c	1.00	0.50	1.00	0.01	0.50		P_c	1.00	1.00	1.00	0.50	0.50
	Mutation Operator	RSM	RSM	RSM	SM	SM		*Mutation Operator*	SM	RSM	RSM	RSM	RSM
	P_m	0.30	0.30	0.30	0.30	0.30		P_m	0.30	0.30	0.30	0.01	0.30

4.3.4 Computational Results for WSRPs Instances

The suggested parameter settings were implemented on WSRP problem sets with more running-time to allow more search operators to converge to better results. For A and B, 10 minutes, for C, 15 minutes and D and E, 60 minutes.

Table 4.5 presents the best results for all instances. The columns under SSGA* and GenGA* show the cost-values obtained by different combinations while excluding the WSR operator, while the columns under SSGA and GenGA provide the best values obtained under the stopping criterion for each combination while including WSR. All values were averaged, and only the best combinations were presented. The remaining

Table 4.5: Cost-values $f(S)$ and computational-time Cpt (in seconds) for WSRP instances, produced by SSGA and GenGA selected operators, crossover probabilities P_c and mutation probabilities P_m.

		SSGA*		SSGA		GenGA*		GenGA	
		$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt
	1	5.58	249.94	5.16	56.61	**3.49**	181.04	3.50	328.17
	2	4.98	204.95	3.38	39.15	2.93	174.92	**2.82**	285.04
	3	6.41	253.44	7.44	111.22	3.65	323.58	**3.12**	507.95
A	4	2.72	248.62	2.80	44.17	**1.44**	207.77	1.44	386.99
	5	3.85	42.94	3.21	4.40	**2.42**	4.27	2.45	60.23
	6	5.48	194.39	4.17	55.90	**3.56**	182.88	3.56	276.54
	7	4.96	15.94	4.25	4.31	**3.71**	7.84	3.73	177.87
	1	3.83	25.94	2.11	78.08	**1.77**	305.78	1.78	413.14
	2	2.65	3.47	2.72	4.21	1.76	58.48	**1.75**	128.18
	3	8.36	175.81	2.85	550.78	2.04	435.89	**1.89**	573.99
B	4	5.51	27.38	2.93	44.85	**2.10**	306.30	2.10	345.31
	5	7.03	113.24	3.41	335.18	**1.94**	418.09	1.94	547.77
	6	7.40	86.73	2.16	273.79	**1.70**	418.94	1.72	460.10
	7	8.21	110.96	3.20	351.44	1.93	475.04	**1.89**	504.27
	1	394341.83	798.36	324555.66	866.74	662253.71	589.90	**75208.26**	571.95
	2	5.18	4.62	4.12	1.26	**3.55**	409.96	3.78	353.33
	3	261333.44	638.02	211405.22	816.67	382247.00	576.78	**61351.66**	569.88
C	4	440.26	289.84	312.23	59.03	222.60	367.07	**120.71**	522.53
	5	530.79	185.45	364.46	58.34	120.64	341.05	**80.76**	400.95
	6	293445.66	722.82	237271.89	816.86	467311.69	534.08	**70786.55**	532.68
	7	4.98	2.33	5.08	0.99	4.30	43.63	4.30	215.92
	1	**1560506.99**	3587.93	2742067.51	3155.68	2530135.41	3596.58	2189058.06	2645.13
	2	**955096.64**	3518.80	1655755.36	2804.58	1321745.19	3596.89	1129456.71	2847.66
	3	**6084146.85**	3582.11	7973238.46	2814.94	7793159.03	3594.57	6671199.21	2833.55
D	4	**2588691.22**	3593.84	4150410.90	3253.95	3766073.54	3594.14	2978199.33	2735.71
	5	**3720708.50**	3584.42	4713830.98	2582.71	5051573.31	3597.80	3907863.67	2792.75
	6	**8014609.32**	3582.28	9889700.46	2451.58	9910391.21	3585.91	8689761.69	3183.55
	7	**7842175.01**	3600.27	9440076.60	3129.99	9346339.16	3590.40	7890003.91	2801.70
	1	275371.06	3583.78	90868.20	3586.18	205793.51	3597.53	**63658.08**	3579.86
	2	447792.39	3586.77	251200.99	3554.57	154360.39	3596.89	**115080.39**	3573.83
	3	367955.32	3579.78	227864.93	3564.41	588658.24	3596.89	**150792.89**	3521.00
E	4	4151.48	3580.78	66.10	3587.40	22.93	3595.39	**20.57**	3591.63
	5	516214.01	3597.31	223424.70	3552.50	223395.15	3598.28	**164878.31**	3584.25
	6	47.86	3564.09	46.48	3579.99	7.61	3585.40	**6.31**	3586.50
	7	657978.25	3582.37	400784.56	3531.31	548383.93	3597.16	**257467.44**	3541.26

Bold text refers to the best result.

column Cpt show the computation-time where the best value was found in seconds.

It appears that GenGA* and GenGA have provided better results than SSGA* and SSGA for A, B, C and E. However, to verify the above observations, three issues were taken into consideration: the GAs behaviour with each instance, the WSR effect on their performance and the best computational-time.

Genetic operators performed differently when WSR was excluded. Not only there was an evident change in the performance, but also some instances have obtained better overall $f(S)$ when excluding WSR, such as GenGA* in scenarios A and B and SSGA* in scenarios D. Not using the WSR gave more time for the evolution process.

In D and E, high rates required more time, and hence the repair was not applied. Instances in problem sets C needed WSR to minimise constraint violations. Therefore, combining WSR as a repair operator with any combination of generic operators was equally important in this GA design, particularly for the more complicated instances.

Having steady state replacement benefited the SSGA to provide better performance by obtaining the best results for problem set D. As previously illustrated in Figure 4.3, SSGA utilised more WSR on problem set D. These conclusions also apply to the results in Table 4.5, where better results were provided in these instances by allowing WSR to improve the quality of solutions and escape local minima.

Another previously stated conclusion, from Figure 4.6, was the best-performing mutations under the methods applied. The results in Table 4.5 show that PMX, UX, and HX obtained the best results across all instances when combined with RSM and SM; however, CBM managed to get some of the best results. This indicates that problem domain knowledge needs to be incorporated in the operators for more complicated instances.

Results reported in Table 4.5 show the average computation-time when the best solution found over all the runs. The average computation-time was as follows: SSGA*, SSGA, GenGA* and GenGA were 1528.69 s, 1515.69 s, 1583.49 s and 1556.96 s , respectively. WSR was computationally expensive when comparing it to not using it. This outcome might be caused by the time elapsed to minimise the constraint violations.

Clearly, SSGA provided solutions in less time than GenGA. Regarding algorithm speed, the SSGAs was faster, with poor results in problem sets A, B, C, and E. Despite the fact that the cost values need to be improved, this study has helped in understanding the performance of a set of combinations and rates under two different simple GAs.

4.4 Summary and Final Remarks

This chapter investigated the suitability of a set of operators, by a performance comparison between twelve operators, on WSRP instances. Thirty-five HHC scenarios were tackled by a steady state genetic algorithm (SSGA) and a generational genetic algorithm (GenGA) with common objectives. Constraints in these problems included those related to workers' skill qualifications, time availability and time-conflicts.

A time-conflict reduction heuristic (TCR) has managed to improve the quality of the initial solution by reducing time-conflict occurrences. A worker suitability repair operator (WSR) was used to reduce the violation on constraints into two stages of the GAs, after the TCR and then again after the mutation operator in the optimisation process. The GAs obtained good results that can be enhanced further by utilising the WSR operator, regardless of the combination of genetic operators used. The GA provided solutions faster when using WSR than when not using a repair process. However, other constraints were ignored initially, and the conflicts were not reduced enough to provide feasible solutions. Hence, other constraint handling methods must be applied to restrict the initial solutions to a feasible area only, such as special encoding schemes.

Another factor in reducing the constraints violations used in this chapter was the weights associated with constraints violations in the objective function. They were able to help the GAs in reducing the number of constraint violations. For example, in small instances, weights were high enough to only provide feasible solutions with the help of WSR. However, when the number of visits ≥ 100, many constraint violations was not reduced, and time-conflicts constraints violations were increased.

As mentioned in section 4.2, a chromosome was a vector of integers of length equal to the number of assignments, $|V|$, consequently, the GA's speed was affected by the instances dimensions. Even though the same repair operators were used, the overall performances vary according to the problem dimensions and the GA design. Any violation in the solutions delays the algorithm speed by performing more repairs. This was an important consideration, especially in the case of real-world problems, where generating good solutions was critical. This tendency to obtain more violations can be controlled

by regulating the quality of individuals in the early stages of the GAs.

In this chapter, two new variations of operators were proposed with problem domain knowledge for solving the WSRP. Additionally, existing well-known generic operators, three mutation operators and seven crossover operators, which have performed well on related problems were selected for WSRP.

From the results, existing operators such as RSM and OX performed the best under WSRP; however, the proposed operators were neither the best nor the next best in all WSRP scenarios. To establish a better performance of a simple GA, mutation operators with limited changes were needed that allow small variations on the long run. At the same time, crossovers that create more disturbance in an offspring might undertake more time to converge but guarantee dissimilar offspring than their parents.

Achieving feasibility in solutions was a serious challenge for most known genetic operators. Traditional operators obtain better results but still far from the optimal. The cost-based recombination operators find it challenging to combine good features of the parents as a result of disorganised chromosome encoding.

The GA design was restricted to utilising basic evolutionary procedures without any improvements. Even after incorporating a repair operator, infeasibility was a serious issue. This was apparent in the results when compared to the optimal values obtained by the MIP solver. Thus, using more efficient methods is recommended to create feasible initial solutions. The process of algorithm tuning was also time-consuming, and the selected parameters values were not necessarily the best. While this study goal was not to present a competitive GA design, the results did partially substantiate of the possibility of getting better performance while using the ideas of repair heuristics, TCR and WSR. The next chapter intends to present a design of a more tailored algorithm to solve the WSRPs, which uses problem-specific representation and operators to exclude infeasible solutions. Additionally, parameter tuning will be conducted to identify the optimal settings for a combination of operators.

Chapter 5

A Baseline GA for a WSRP

5.1 Introduction

A comprehensive study of genetic operators within two simple GAs for WSRP was presented in Chapter 4. That study showed that achieving feasibility in solutions to this problem is a serious challenge for most known genetic operators. Even after incorporating a repair operator, infeasibility was a serious issue. Given this, it is recommended to use an efficient and straightforward method to create feasible initial solutions.

As mentioned in Chapter 3, some authors have identified the effects of hybridising the GA with problem-specific heuristics on generating high-quality initial solutions that ensure the GA efficiency [73, 138]. In the work described by [143], a schedule encoding scheme was presented, while using several heuristic methods into the decoding procedure. Based on the literature survey, it was suggested to use an efficient and simple constructive heuristic to create initial solutions for a GA that aim to solve real-world problems.

The intended contribution of this chapter is to present an efficient, customised chromosome representation combined with problem-specific genetic operators for tackling real-world WSRP instances. Such approach has not yet been proposed in the literature for HHC instances. The proposed solution encoding is capable of producing feasible competitive solutions for a highly constrained WSRP.

This chapter also presents a study of genetic operators using the proposed solution encoding. Well-known genetic operators, performed better than cost-based operators

presented in Chapter 4), plus five proposed tailored operators are implemented. The proposed GA harnesses the power of these genetic components to solve the WSRP instances considered. The specific objectives of this chapter are:

- To develop an indirect chromosome encoding scheme that ensures the feasibility of solutions potentially improving the exploratory capabilities of the algorithm.
- To develop cost-based genetic operators tailored for the indirect chromosome encoding scheme by restricting the search to the feasible region.
- To carry out computational tests to evaluate the proposed GA and compare the performance of the various genetic operators considered.

5.2 Tailored Genetic Algorithm for WSRP

An effective baseline GA to tackle the HHC problem instances is described in this section. The proposed GA uses an indirect chromosome encoding and incorporates various genetic operators including some new problem-specific ones that utilise heuristics to generate improved offspring. The proposed GA works as follows. First, an initial population of M individuals (one-day plans) is created based on the indirect chromosome encoding to ensure solutions feasibility. At the start of each generation, $9M/20$ pairs of parent individuals are selected using binary tournaments. With some probability, one of the crossovers is applied to each pair of parents to generate two offspring. With some probability, each offspring goes through one of the five studied mutation operators. At the end of each generation, the population is updated using an elitism strategy, where the $M/10$ best individuals from the current population are kept. Along with the $9M/20$ offspring individuals generated, the new population of M solutions is formed. The indirect chromosome encoding and genetic operators are described next.

5.2.1 Indirect Chromosome Encoding

The primary goal of the indirect representation is to reduce the number of infeasible solutions. A *worker suitability list (WSL)* is created for each visit k, , where $k = 1, 2 \ldots, |V|$. The WSL contains suitable workers only, ranked by a penalty score $M_{k,j}$,

i.e., the lower the score value, the better suited is a worker. The score is calculated for a worker w_j by estimating the impact of assigning that worker to that visit, considering incurred operational cost and penalty cost due to preferences and availability restrictions.

Initial population is created randomly by generating a vector of $|V|$ integers between 0 and $L_k - 1$, where L_k is the length of WSL for a visit k. When a worker is being considered for a visit, the solution is evaluated for time-conflicts. If it occurs, by the random assignments, the next worker in the WSL for that visit is considered until a suitable worker is found with no time-conflicts arising.

Figure 5.1 illustrates an example of the indirect chromosome encoding for a day plan with seven visits. The colours are used to visualise the solution as a chromosome, with each colour presenting a different gene. Each visit has a WSL of four suitable workers, with the best worker at the top. Hence, the first worker in the WSL is the most suitable one for that visit, followed by the next best worker and so on. Below the chromosome, the decoded solution shows the actual worker assigned to each visit.

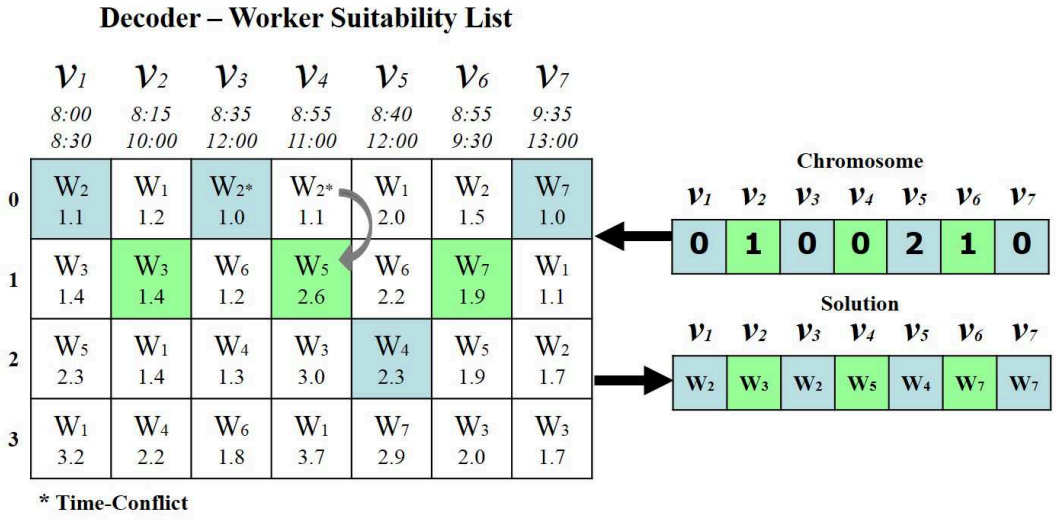


Figure 5.1: An indirect chromosome encoding scheme illustration.

Time-conflicts are indicated in Figure 5.1 with *. For example, w_2 is assigned to both v_1 and v_3 while w_3 is assigned to v_2. No time-conflict arises because v_1 and v_3 do not overlap. However, w_2 is assigned to v_4 and a time-conflict arise as v_3 overlaps with v_4. Then, the next most suitable worker that does not provoke a time-conflict, in

this case w_5, is assigned to v_4. The WSL decoder in this indirect chromosome encoding scheme helps to assign suitable workers to visits while avoiding time-conflicts.

The penalty scores are shown in Figure 5.1 are not used during the generation of initial solutions, they are utilized for the tailored genetic operators described below.

5.2.2 Genetic Operators

Several crossover and mutation operators are implemented in this chapter. Some of these operators were proven to perform well under WSRP in Chapter 4 and others are specially designed for the problem and solution representation considered here.

Seven well-known operators are implemented: 1PX, 2PX, UX, HX, RSM, IM, and SM. Four problem-specific crossover operators are proposed: Cost-Based Greedy (CGX), Partially Matched Greedy (PMGreedyX), Flat-Costs (FCX) and Partially Matched Flat-Costs (PMFCX). Two problem-specific mutation operators are designed here: Flat-Cost flip mutation and Zero-Flip mutation. The indirect chromosome encoding scheme described above allows implementing all genetic operators while maintaining the feasibility of solutions. The problem-specific operators developed here are described next.

Cost-based greedy crossover (CGX)

In Chapter 4, a CBUX crossover was implemented that involved restricting mating so that only feasible assignments of workers to visits are copied from parents to offspring, as much as possible. However, this crossover has not ensured that all workers in an individual were suitable for each visit. Thus, a greedy-based crossover is proposed in this section that builds each assignment based on worker's suitability for a given visit.

$$f(i) = \frac{f(w^i)}{|V|} \tag{5.1}$$

A cost-based greedy crossover (CGX) is essentially a variation of a local search constructive heuristic that uses the objective function to transfer the best genes from each parent to one of the offspring individuals. The cost value is normalised per-visit, during the initial solution construction, by using equation (5.1). Thus, the value of the assignment of one worker to perform a visit i is denoted as $f(i)$.

Algorithm 6 Cost-Based Greedy Crossover (CGX)

Require: Pair of parents p_1 and p_2
Ensure: Pair of offspring o_1 and o_2
1: $o_1, o_2 \leftarrow$ new empty individuals
2: $o'_1, o''_1 \leftarrow$ temporary individuals
3: **for** $i \leftarrow 1$ **to** $|V|$ **do**
4: $o'_1 \leftarrow o_1 \cup p_1^i$
5: $o''_1 \leftarrow o_2 \cup p_2^i$
6: **if** $f(o'_1) < f(o''_1)$ **then**
7: $o_1 \leftarrow o_1 \cup p_1^i$
8: $o_2 \leftarrow o_2 \cup p_2^i$
9: **else**
10: $o_2 \leftarrow o_2 \cup p_1^i$
11: $o_1 \leftarrow o_1 \cup p_2^i$
12: **end if**
13: **end for**

Algorithm 6 shows how this CGX crossover operator works. The operator goes through each of the V positions in the chromosome. Each i^{th} gene in the parent individual is copied to offspring o_1 plus the corresponding gene from each parent respectively to o'_1 and o''_1 (steps 4-5). Then, the parent's gene that results in a better $f(i)$ is added to offspring o_1 while the other parent's gene is added to offspring o_2 (steps 6-12). Thus, CGX produces one offspring (o_1) with the best parent's gene selected at each step and one offspring (o_2) with the other parent's gene. This mechanism restricts the search to better offspring than the parents by maintaining a flow of minimum cost genes. Nevertheless, the method is very time-consuming as a result of the gene-wise calculations during the application. This is also explained in section 5.3.

Partially-Matched Greedy Crossover (PMGreedyX)

In Chapter 4, the partially matched crossover (PMX), which uses a position wise-exchange, was proven to be one of the best crossovers used for WSRP. Thus, in this section, PMX crossover is extended to suit the proposed chromosome representation customised for WSRP. The PMGreedyX is a hybrid crossover of PMX and the greedy process presented in section 5.2.2. Algorithm 7 shows how the PMGreedyX crossover works. The crossover works on two parent chromosomes and selects a segment within two cutting points named as $Point_1$ and $Point_2$, to reverse the positions of genes between these points (steps 14-17). The cut-points are proportional to an arbitrary interval R, selected from the set of all real numbers between $[0, 1)$, such that $0 \leq R < 1$. The selected positions calculations are given in equation (5.2) and equation (5.3).

$$Point_1 = 2 + |R * (V - 4)| \tag{5.2}$$

$$Point_2 = Point_1 + 2 + |R * ((V - Point_1) - 2)| \tag{5.3}$$

The second step is to fill the rest of the offspring. After applying the position wise-exchange, a greedy process is applied outside the selected cut-points (steps 3-13), as explained in CGX. In a traditional PMX, the positions outside the segments are only filled with genes if there are no conflicting values to the ones within the segment. This method is not applied in PMGreedyX to generate offspring faster with fewer calculations.

Algorithm 7 Partially Matched Greedy Crossover (PMGreedyX)

Require: Pair of parents p_1 and p_2
Ensure: Pair of offspring o_1 and o_2
1: $o_1, o_2 \leftarrow$ new empty individuals
2: $o'_1, o''_1 \leftarrow$ temporary individuals
3: **for** $i \leftarrow 1$ **to** $Point_1 - 1$ **and** $i \leftarrow Point_2 + 1$ **to** $|V|$ **do**
4: $o'_1 \leftarrow o_1 \cup p_1^i$
5: $o''_1 \leftarrow o_1 \cup p_2^i$
6: **if** $f(o'_1) < f(o''_1)$ **then**
7: $o_1 \leftarrow o_1 \cup p_1^i$
8: $o_2 \leftarrow o_2 \cup p_2^i$
9: **else**
10: $o_2 \leftarrow o_2 \cup p_1^i$
11: $o_1 \leftarrow o_1 \cup p_2^i$
12: **end if**
13: **end for**
14: **for** $i \leftarrow Point_1$ **to** $Point_2$ **do**
15: $o_2 \leftarrow o_2 \cup p_1^i$
16: $o_1 \leftarrow o_1 \cup p_2^i$
17: **end for**

Figure 5.2 illustrates an example of a PMGreedyX for a day plan with $V = 7$ visits. Given that the selected points are $Point_1 = 2$ and $Point_2 = 4$. Genes within the cut points are copied from the parents and exchanged in the offspring. Genes outside the cut points go through a gene-wise comparison (indicated by an arrow). Any gene with a better $f(i)$ is transferred to the first offspring (o_1). Otherwise genes are moved to the second offspring (o_2). This process continues until both offspring are constructed.

The PMGreedyX generates offspring in fewer steps than CGX. This is the result of mixing the greedy heuristic in CGX, which is a cost-expensive method, with the PMX position-wise exchanges. Hence, this crossover can produce offspring faster than CGX where a full greedy method applied.

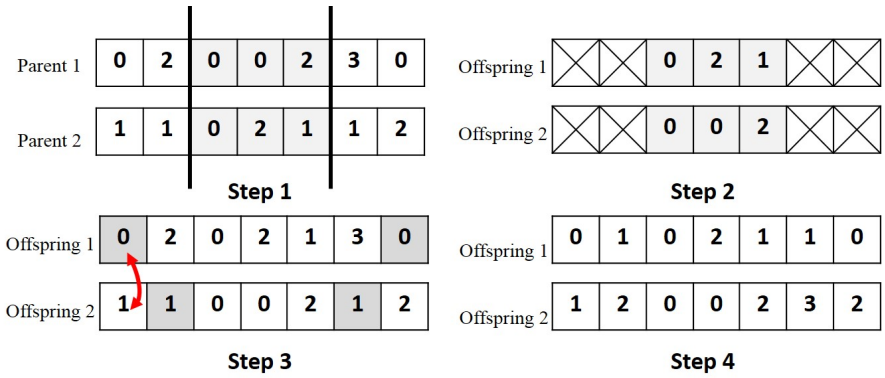


Figure 5.2: A PMGreedyX crossover illustration.

Flat-Costs Crossover (FCX)

The WSL capabilities are extended in this crossover. The FCX uses the flat-costs approach, adopted from the WSL, to generate offspring with better assignments than the parents. Algorithm 8 shows the steps of this FCX crossover operator. Rather than computing the $f(i)$ for each visit in the offspring, as in CGX, or for the partial solution, as in the case of PMGreedyX, the FCX uses the penalty scores that were initially calculated in the WSL at the start of the GA. These values are denoted as 'flat-cost' where each worker w_j has a penalty score according to their suitability to work in visit i.

Algorithm 8 Flat-Costs Crossover (FCX)

Require: Pair of parents p_1 and p_2. The set of *WSL* for all visits, $M_{i,j}$ is the penalty score for the j^{th} worker in the *WSL* for visit i.
Ensure: Pair of offspring o_1 and o_2.
1: $o_1, o_2 \leftarrow$ new empty individual
2: **for** $i \leftarrow 1$ **to** $|V|$ **do**
3: **if** $(M_{i,p_1^i} < M_{i,p_2^i})$ **then**
4: $o_1 \leftarrow o_1 \cup p_1^i$
5: $o_2 \leftarrow o_2 \cup p_2^i$
6: **else**
7: $o_2 \leftarrow o_2 \cup p_1^i$
8: $o_1 \leftarrow o_1 \cup p_2^i$
9: **end if**
10: **end for**

The FCX crossover operator goes through each of the V positions in the parent's chromosome. A gene-wise comparison is enforced, for each gene in i^{th} position, with respect to the WSL estimated penalty scores M_{i,p^i}. The best suitable worker is given to offspring (o_1) and the other worker for offspring (o_2) (steps 2 to 10). This mech-

anism has fewer steps than CGX and PMGreedyX combined. Consequently, the FCX is more efficient than those crossovers, computationally-wise. On the other hand, the fitness evaluations are initial estimations, and therefore less accurate than the greedy calculations. This consideration is explored in the next section.

Algorithm 9 Partially Matched Flat-Costs Crossover (PMFCX)

Require: Pair of parents p_1 and p_2. The set of *WSL* for all visits, $M_{i,j}$ is the penalty score for the j^{th} worker in the *WSL* for visit i.
Ensure: Pair of offspring o_1 and o_2
1: $o_1, o_2 \leftarrow$ new empty individuals
2: $o_1', o_1'' \leftarrow$ temporary individuals
3: **for** $i \leftarrow 1$ **to** $Point_1 - 1$ **and** $i \leftarrow Point_2 + 1$ **to** $|V|$ **do**
4: $o_1' \leftarrow o_1 \cup p_1^i$
5: $o_1'' \leftarrow o_1 \cup p_2^i$
6: **if** $(M_{i,p_1^i} < M_{i,p_2^i})$ **then**
7: $o_1 \leftarrow o_1 \cup p_1^i$
8: $o_2 \leftarrow o_2 \cup p_2^i$
9: **else**
10: $o_2 \leftarrow o_2 \cup p_1^i$
11: $o_1 \leftarrow o_1 \cup p_2^i$
12: **end if**
13: **end for**
14: **for** $i \leftarrow Point_1$ **to** $Point_2$ **do**
15: $o_2 \leftarrow o_2 \cup p_1^i$
16: $o_1 \leftarrow o_1 \cup p_2^i$
17: **end for**

Partially Matched Flat-Costs Crossover (PMFCX)

PMFCX is a hybrid crossover between PMX and FCX. Algorithm 9 describes how the PMFCX works. Cutting points are selected similarly to PMGreedyX, as shown in section 5.2.2. Once the points are selected, the positions of genes between these points are exchanged (steps 14-17). Outside the cutting points, it applies the FCX mechanism to fill the rest of the offspring (steps 6 - 14). As an illustration, Figure 5.3 shows an example of a crossover process using PMFCX for a day plan with $V = 7$ visits. Given that the selected points are $Point_1 = 2$ and $Point_2 = 4$. Similar to PMGreedyX, genes within the cut points are copied from the parents and exchanged in the offspring. Genes outside the cutting points go through a gene-wise comparison (as shown with an arrow).

Genes contents are indexes decoded as workers scores in the WSL. For a visit k, if $M_{k,p_1^i} < M_{k,p_2^i}$, then p_1^i is selected to be transferred to the first offspring (o_1) because it has the least penalty cost score. Otherwise, genes are transferred to the second offspring (o_2). This process continues until both offspring are constructed.

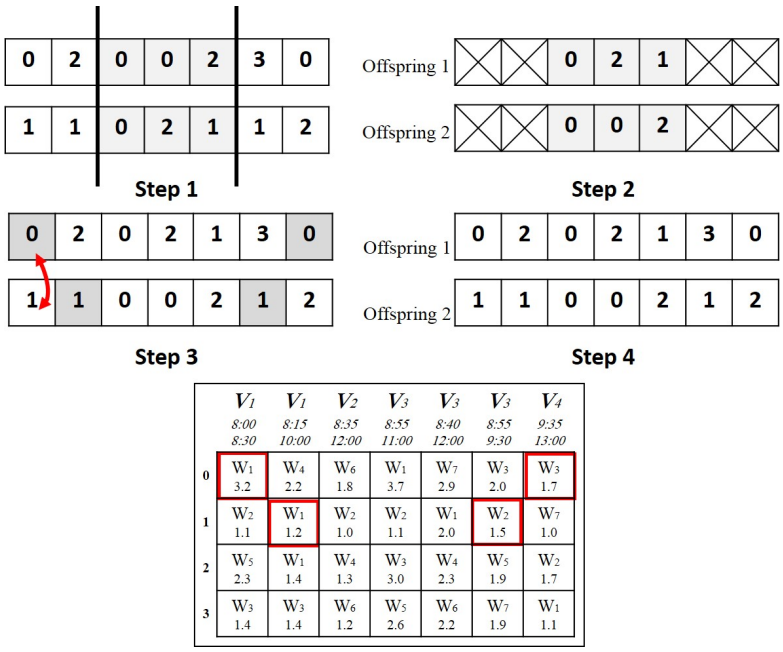


Figure 5.3: A PMFCX crossover illustration.

Flat-cost flip and Zero-flip mutation

Two mutation operators were extended from the flip mutation [114] to fit the proposed chromosome representation. The WSL is also utilised in these operators by simply replacing a worker, selected at random, and currently assigned to a visit by another worker.

1. **The flat-cost flip mutation (FCF)** would introduce new workers to the chromosome, even if the worker were not suitable for the corresponding visit. Algorithm 10 shows how this operator works. A random position i of the chromosome is replaced by a random number within the list of visit i in WSL. For example, if a worker w at the second gene in a chromosome is selected to be altered, a random worker $w' \in WSL$ replaces the w in the chromosome. The FCF increases the diversity by the random process, even if the best worker for visit i is not selected.

Algorithm 10 Flat-Costs-Flip Mutation (FCF)

Require: individual k. The set of *WSL* for all visits, $M_{i,j}$ is the penalty score for the j^{th} worker in the *WSL* for visit i.

1: Choose a random point $i \in \{1, \ldots, |V|\}$
2: Let w^i be the worker assigned to visit i in k
3: Choose a random point $w' \in WSL$
4: $w^i := w'$

2. **The zero-flip mutation** always replaces a random gene with the most suitable worker for that visit, i.e., the first worker in WSL, regardless of time-conflicts, as presented in Algorithm 11. This operator can confirm further that the best worker for a given visit is not always the best for the entire plan. On the contrary to FCF, the Zero-flip mutation tend to generate similar individuals by selecting the same worker more often.

Algorithm 11 Zero-Flip Mutation

Require: individual k. The set of *WSL* for all visits, $M_{i,j}$ is the penalty score for the j^{th} worker in the *WSL* for visit i.

1: Choose a random point $i \in \{1, \ldots, |V|\}$
2: Let w^i be the worker assigned to visit i in k
3: Select $w_0 \in WSL$
4: $w^i := w_0$

5.3 Experimental Study and Results

Experiments were conducted to assess the performance of the proposed GA on the 42 real-world WSRP scenarios from Table 2.4. The methods used to evaluate the results were used as explained in 4.3. See Appendix C for all results using parameter settings.

Table 5.1: Preliminary parameter settings for the baseline GA

Variable	Value
Population Size M	100, 250 and 500
Mutation Rate P_m	1%, 10%, 30%, 40% and 50%
Crossover Rate P_c	10%, 50% and 100%
Stopping Criterion	5 minutes
Selection Strategy	Elitist

The first set of experiments goal was to identify the best genetic operators and parameter settings for each problem instance except those in problem set F (the largest ones). The different parameter settings in Table 5.1 were tested. This means that for each of the 35 problems instances in groups A to E, there were 1800 configurations (8 crossovers $\times$ 5 mutations $\times$ 45 parameter settings).

Each configuration used the same initial population and was executed eight times; each one ran for 5 minutes. This means a massive amount of computation time but since it was possible to perform parallel runs, the total computation time was drastically reduced but still accounted for many days.

The observations made based on the experimentation below will identify the best operators, parameter rates and population sizes for A, B, C, D and E problem sets. Therefore, parameter setting was later applied for longer computation time for all problem sets including the F problem instances.

5.3.1 Mutation Operators Performance with Different Rates

The performance of each mutation operator was shown in the left plot of Figure 5.4. The x-axis presents the instance sets while the y-axis presents the average gap to the best-known solutions. The calculations for the cost-values normalisation and relative gap percentage were calculated as described in Chapter 4.3.

For each mutation operator, a point in the graph corresponds to the average gap considering all instances in the corresponding set. The dashed line joining the 5 points was only used to visualise the overall performance of the mutation operator across all sets. The lower the line, the better, i.e., the closer to the best-known results.

The left plot of Figure 5.4 shows that for all instances the FCF mutation operator produced better results on sets A, B, and C. It also performed better on sets D and E but by a smaller margin in comparison to the other mutation operators. FCF mutation operator was the only operator that introduces new genes (using the *WSL*) to the chromosome encoding. The other mutation operators only reallocate existing genes. This may be the contributing factor that encourages more diversity that enhances the overall performance of the FCF mutation.

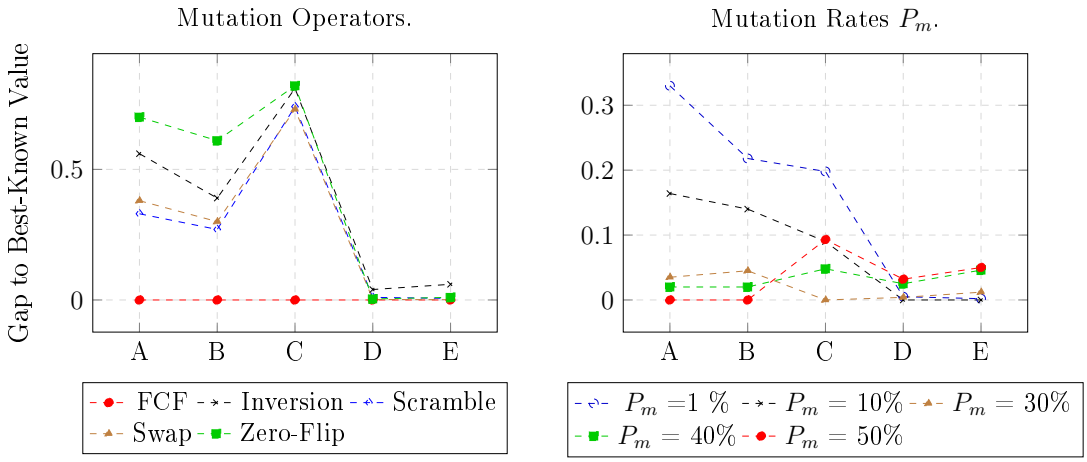


Figure 5.4: Performance comparison between mutation operators. Each line represents the relative gap between the best known solution corresponds to the problem set and a solution by a mutation operators (left plot) or a mutation rate P_m (right plot).

Better results obtained by the FCF mutation on sets A, B, and C because the algorithm was able to execute many more generations as a consequence of the chromosome size that reflects the number of assignments. This increased the algorithm's ability to bring new genes into the population faster and boost its performance. For sets D and E, FCF mutation has succeeded by a smaller margin also as a result of the chromosome size. With a larger size, the solution was diverse beforehand and increasing the diversity can delay convergence. Thus, parameter settings must be implemented with more running-time to gather evidence on the efficiency of the GA under WSRPs.

To investigate whether a particular mutation rate performed better than others on each set of instances, results were presented in right plot of Figure 5.4 for each mutation rate. Setting an appropriate mutation rate helps a GA not to get stuck in local optima. Rates order in Figure 5.4 for sets A and B was inverted concerning sets E and F, with set C being the inflection point. For the smaller instances, a high mutation rate was considerably more effective than a low mutation rate. However, as the instances grow in size, the opposite happens, with a low mutation rate being more effective. The reason for this was also related to the problem size.

In general, a high mutation rate reduces the convergence rate of the algorithm but increases diversity in the population [60]. Hence, on the long term, having a high muta-

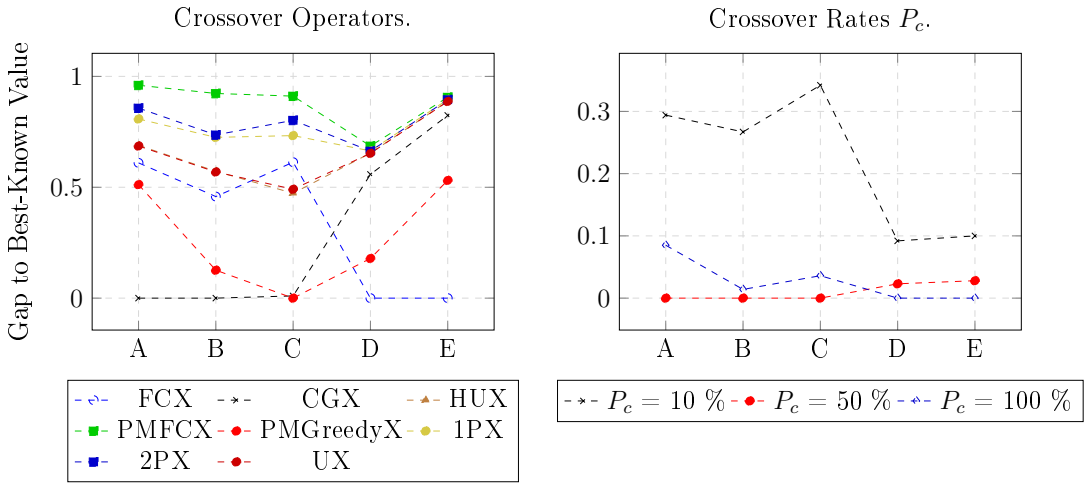


Figure 5.5: Performance comparison between crossover operators. Each line represents the relative gap between the best known solution corresponds to the problem set and a solution by a crossover operator (left plot) or a crossover rate P_c (right plot).

tion rate aids the overall performance of the GA. Although the algorithm was converging slower, it was exploring a wider region of the search space and avoiding local optima with a higher probability. A low mutation rate, however, helps the algorithm to converge much faster. But the likelihood of getting stuck into local optima was higher.

Given the time limit set for each run in these experiments, many more generations take place for sets A and B than for sets D and E as a result of the problem dimensions. Hence, higher mutation rates provide better results on sets A and B, and the lowest mutation rate produces better results on sets D and E. Then, on smaller problem instances, higher mutation rates give better results because there was a need for an increase in diversity to allow the algorithm to avoid local optima.

On larger problem instances, lower mutation rates give better results; this was because, with few generations, the speed of convergence was more important than adding more diversity by the mutation. With large chromosome size, the diversity was high enough. The aim of the study was to understand the operators behaviour rather than to study the convergence rate under the limited time. Perhaps with much longer computation time, higher mutation rates could produce better results on sets D and E, but such experiments were not conducted here.

5.3.2 Crossover Operators Performance with Different Rates

The performance of each crossover operator was shown in the left plot of Figure 5.5. The figure shows that two of the four proposed crossover operators, CGX and PMGreedyX, outperformed well-known crossovers such as 1PX and UX. These results support the idea of having problem-specific operators to help the GA by exploiting domain-knowledge and providing improved solutions, in comparison to traditional crossovers that use random crossing points regardless of the costs.

Left plot of Figure 5.5 shows that the performance of the crossover operators also depend on the problem size. For sets A and B, CGX obtained the best results. Since problem instances in sets A and B have less number of visits, the greedy evaluation in CGX was fast and helps in the convergence to better solutions. On sets A and B, PMGreedyX performed better than FCX and PMFCX but worse than CGX. This was the consequence of the segment selected between the crossing points used on constructing an offspring compared to CGX.

In problem sets D and E, FCX performed the best. It performed worse than other operators in sets A, B, and C. This indicates that instead of the greedy evaluation, using the pre-calculated penalty scores in *WSL* was more effective in larger problems, but it reduces the search space too much in smaller problem instances. However, one of the proposed operators with pre-calculated penalty scores, PMFCX, performed the worse across all sets. This was an indication that combining PMX with FCX to construct an offspring was not too successful.

The flat-cost evaluation affected the production of offspring with an increase in the number of generations. Thus, estimated costs were not accurate enough in the case of PMFCX. Note that for the set C, PMGreedy obtained slightly better results that CGX. This might be due to the much larger number of workers in the problem instances of set C. This was an advantage for the greedy evaluation in PMXGreedy.

The right plot of Figure 5.5 shows a comparison between crossover rates across sets. Results indicate that a low crossover rate of 10% was not successful for all problem sets. On higher rates of 50% and 100%, the following observations were made.

For larger problems (longer chromosomes) in sets D and E, the highest crossover rate

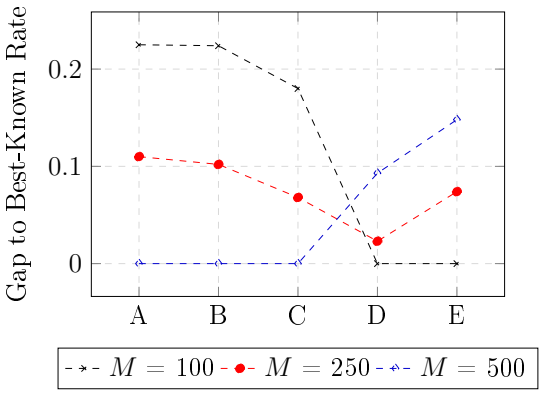


Figure 5.6: Performance comparison of different population sizes. Each line represents the relative gap between the best-known solution corresponds to the problem set and a solution by a population size M.

was better. But for smaller problems in sets A, B, and C, a rate of 50% gave better results. The number of generations played a role in those results. A lower crossover rate, with slower evolution speed, was good enough for smaller instances to exploit the search space. However, for larger instances, less number of generations was required, given the limited run-time and problem dimensions, hence higher crossover rate helped to speed up the evolution to generate an offspring.

5.3.3 Performance of Different Population Sizes

Figure 5.6 shows the comparison between population sizes for all problem sets. The lines reflect the gap to the best-known population size value for all instances. The lower the line, the better, i.e., the closer to the best-known results.

The size of the problem instance was of course reflected by the length of the chromosome. On the other hand, the number of different individuals in a population reflects the overall diversity. As it can be seen from the plot, smaller problems, such as problem set A, B, and C, have benefited from larger population sizes.

For larger problems, smaller population size was better. This was because, in smaller problem instances, a large population helped in having more diverse individuals, while in larger problems longer chromosomes have less need for a large population and therefore there was no need for more diverse solutions that cause a delay the evolution process, due to time limitations.

Based on the above findings, the combinations of genetic operators and their corresponding rates were shown in Table 5.2. Additional experiments were conducted while using these settings, for all problem instances but with longer computation times.

Table 5.2: Recommended settings for WSRP problem sets.

	A	B	C	D	E	F
Mutation Operator	*FCF*	*FCF*	*FCF*	*FCF*	*FCF*	*FCF*
Mutation Rate P_m	0.5	0.5	0.3	0.1	0.1	0.1
Crossover Operator	*GCX*	*GCX*	*PMGreedyX*	*FCX*	*FCX*	*FCX*
Crossover Rate P_c	0.5	0.5	0.5	1	1	1
Population Size N	500	500	500	250	250	250
Runing-time	15 Minutes	15 Minutes	15 Minutes	60 Minutes	60 Minutes	8 Hours

Given the findings from the previous experiments, parameters for F were set to the same settings as for sets D and E. The computation times used were as follows: 15 minutes per run for sets A, B, and C; 60 minutes per run for sets D and E; 8 hours per run for set F. Note that in Section 5.3.3, $M = 100$ was selected due to time limitations. However, the next experiments have more running time for those instances, thus the population size was increased to M =250 to allow more exploration.

5.3.4 Comparisons Between WSRP Solution Methods

Table 5.3: Cost-values $f(S)$ and computational-time Cpt (in seconds) for WSRP instances, produced by the GA, VNS and MIP DECOMP. on all 42 WSRP problem instances.

		Optimal		GA		VNS		MIP DECOMP.				Optimal		GA		VNS		MIP DECOMP.	
		f(S)	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*			*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*
	01	**3.5**	7.0	**3.5**	4.5	**3.5**	**3.0**	5.7	3.7		01	n/a	n/a	171.7	3414.5	**170.3**	3036.2	496.4	**1060.0**
	02	**2.5**	8.0	**2.5**	20.6	**2.5**	**0.3**	4.5	3.6		02	n/a	n/a	168.2	3456.5	**163.9**	2840.8	372.9	**1192.1**
	03	**3.0**	14.0	3.2	169.2	**3.0**	29.3	10.7	**3.7**		03	n/a	n/a	181.7	3473.7	**178.2**	3172.8	3213.3	**1209.0**
A	04	**1.4**	5.0	**1.4**	238.5	**1.4**	6.9	3.1	**2.9**	D	04	n/a	n/a	170.2	3438.0	**167.1**	3313.8	418.9	**3005.1**
	05	**2.4**	1.0	**2.4**	1.3	**2.4**	**0.1**	3.5	1.8		05	n/a	n/a	162.0	3499.8	**161.1**	2818.5	243.9	**1306.7**
	06	**3.5**	5.0	**3.5**	90.0	3.6	29.5	3.7	**2.4**		06	n/a	n/a	178.4	3532.3	**177.4**	2996.0	1411.3	**1222.0**
	07	**3.7**	1.0	**3.7**	0.9	**3.7**	**0.1**	4.8	1.6		07	n/a	n/a	178.7	3465.3	**177.9**	2930.4	753.3	**1361.9**
	01	**1.7**	21.0	**1.7**	449.0	**1.7**	156.5	1.8	**8.1**		01	n/a	n/a	**1.2**	**3435.6**	n/a	n/a	33.0	8408.0
	02	**1.8**	2.0	**1.8**	0.8	**1.8**	**0.0**	1.9	4.3		02	n/a	n/a	**1.2**	**3512.8**	n/a	n/a	26.0	12448.4
	03	**1.7**	6003.0	1.8	444.7	**1.7**	473.2	2.1	**32.9**		03	n/a	n/a	**1.2**	**3536.2**	n/a	n/a	29.0	20746.6
B	04	**2.1**	25.0	**2.1**	118.2	**2.1**	**9.0**	2.2	15.2	E	04	n/a	n/a	**1.3**	**3511.8**	n/a	n/a	28.5	15190.5
	05	**1.8**	585.0	1.9	328.2	**1.8**	135.3	4.7	**25.3**		05	n/a	n/a	**2.2**	**3460.7**	n/a	n/a	270.1	32619.2
	06	**1.6**	184.0	1.7	354.0	**1.6**	194.3	2.5	**24.1**		06	n/a	n/a	**1.3**	**3350.1**	n/a	n/a	24.6	24212.1
	07	**1.8**	300.0	**1.8**	315.9	**1.8**	305.0	4.1	**23.6**		07	n/a	n/a	**1.7**	**3502.1**	n/a	n/a	427.8	51057.3
	01	n/a	n/a	118.0	939.5	**114.2**	301.3	905.0	**211.6**		01	n/a	n/a	**2109.0**	**28628.6**	n/a	n/a	64305.1	3446.4
	02	**3.2**	6.0	**3.2**	**0.0**	**3.2**	**0.0**	3.6	0.6		02	n/a	n/a	**2116.4**	**28418.7**	n/a	n/a	73291.2	1111.0
	03	n/a	n/a	105.0	861.2	**103.5**	550.3	1186.3	**26.3**		03	n/a	n/a	**609.4**	**28591.2**	n/a	n/a	115235.2	4555.1
C	04	**11.1**	90.0	**11.1**	5.2	11.2	**0.9**	81.3	3.1	F	04	n/a	n/a	**1331.5**	**28256.4**	n/a	n/a	102994.2	4219.2
	05	**12.3**	55.0	12.5	96.7	**12.3**	**0.3**	68.9	1.1		05	n/a	n/a	**200.6**	**28667.2**	n/a	n/a	101438.2	6156.5
	06	n/a	n/a	142.0	847.0	**140.4**	323.5	3102.3	**47.0**		06	n/a	n/a	**627.2**	**28181.4**	n/a	n/a	76007.1	9695.6
	07	**4.3**	1.0	**4.3**	**0.0**	**4.3**	**0.0**	5.3	0.2		07	n/a	n/a	**3499.1**	**28262.4**	n/a	n/a	176540.6	3832.8

Bold text refers to the best solution.

Table 5.3 presents the best results obtained over different runs, with the proposed baseline GA using the settings recommended in Table 5.2. Results from the GA were compared to existing work on WSRP instances: the VNS by [127] and the MIP decomposition method by [79]. The later attempts to solve sub-problems to optimality in a time given, but when all the solutions were integrated, the final solution was not optimal.

For each problem instance, the best-known value $f(S)$ and the computation time in seconds Cpt, in which the best solution was found was shown and highlighted in **bold**. This is, of course, the optimal solution. When available, the optimal solutions obtained by the MIP solver [79] and corresponding computation time were also shown.

Results indicate that for the smaller problem sets, the proposed GA was quite competitive matching the optimal values for many of those instances. The VNS seems to provide better overall results but the GA outperforms the MIP decomposition method. For problem sets D, the proposed GA performed better than the MIP decomposition method but still worse than the VNS.

Table 5.4: Average relative gap to the best-known solution obtained by GA, VNS and MIP DECOMP.

Method	A	B	C	D	E	F
GA	1.10%	2.23%	1.08%	3.08%	**0.00%**	**0.00%**
VNS	**0.02%**	**0.00%**	**0.00%**	**0.00%**	-	-
MIP DECOMP.	85.41%	54.46%	710.31%	486.20%	6765.94%	14310.90%

Bold text refers to the best result.

The reason for obtaining better results with VNS was because it targets nearby local optima, achieved by constructive heuristics to build the initial solution, with one parameter that was easier to be adjusted [68]. Where the GA was a stochastic method with different parameter settings to be tuned through a large number of empirical test, that contribute in slowing down the GA.

The GA tardiness, in comparison to VNS, was because of the number of crossovers applied and evaluation throughout the evolutionary process. To improve the GA, fewer operations must be applied, during the search, and thus fewer crossovers. Note that the GA has obtained the best results in E and F, where results of VNS were not available.

Table 5.4 presents the average gap percentage to the best-known cost value $f(S)$

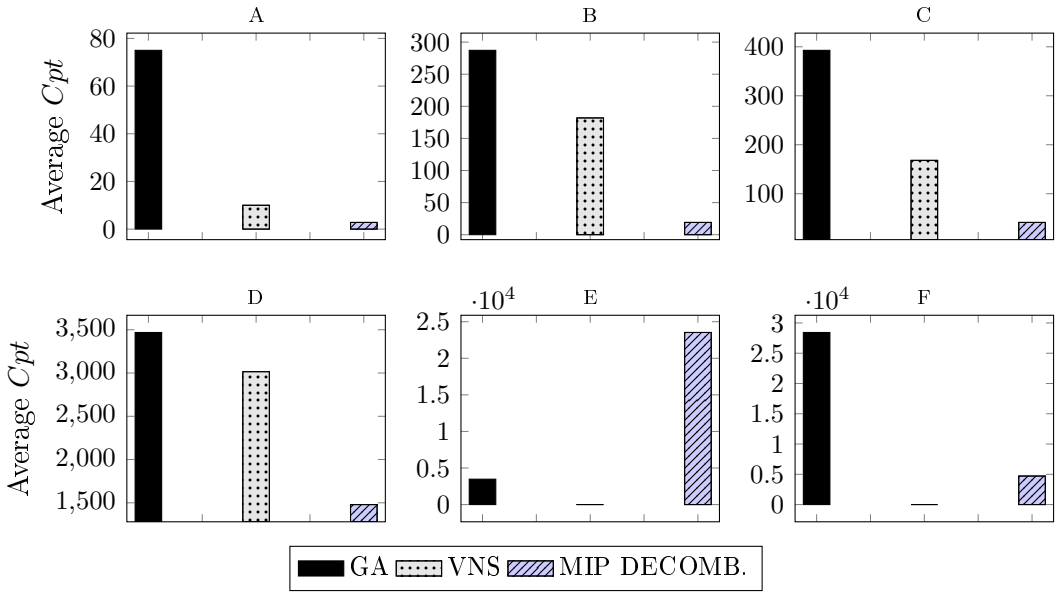


Figure 5.7: Average computation-time (in seconds) by GA, VNS and MIP DECOMP, on problem sets A to D and for GA and MIP DECOMP on problem sets E and F.

obtained by the solver when available). The best method out of the three methods was highlighted in **bold**.

The results show that VNS obtained the overall average values on A, B, C and D, when values were available, followed by the GA with a small marginal gap. The MIP with decomposition did not obtain any best results.

On problem sets E and F, VNS results were not available, hence GA obtained the best average gap percentage values.

When MIP with decomposition was used, there was a significant gap divergence from the GA results. Using the problem domain knowledge in a metaheuristic method, as the case in VNS and GA, have provided better outcomes in comparison to MIP with decomposition. The average gap percentage increased with the problem set size, on both the GA and MIP decomposition. Thus, the problem difficulty increases with the number of assignments, and therefore the size of the chromosome.

Figure 5.7 illustrates the average computation-time in seconds as stated in Table 5.3. Each sub-figure corresponds to a problem set and presents the average computation-time used by GA (solid black bars), VNS (gray bars with dots) and MIP decomposition, (blue

bars with inclined lines). The lower the bar, the better, i.e., the less computation time.

On problem sets A to D, all three methods were applied, and therefore their overall average computation-time was plotted. However, on problem sets E and F, the VNS values were not available.

As it can be seen from Figure 5.7, the GA has resulted in larger computation-time among all methods. The MIP with decomposition method obtained the lowest average computation-time, with an exception on E where MIP decomposition spends 5.7% more time than GA to compute the results.

For some of these small problem instances, the GA spends considerably more time than the other methods, however, still below 15 minutes which was quite acceptable. This result combined with the outcomes of Table 5.4 indicated that the MIP with decomposition method computes the solutions faster, but the quality of the solution needs to be improved.

On the other hand, GA spends more time to compute the solutions yet with better solution's quality. Hence, it was argued that to obtain good results with the GA, more computational-time was required, in particular on the largest instances, to employ the tailored indirect chromosome encoding scheme and genetic operators.

Given the results in Table 5.3, The fast process of VNS has obtained the best results with the best computation-time when results were available. In practical environments, where real-world problems require a balance between good quality solutions and fast computation, VNS was considered a better option than the GA for WSRP. Nevertheless, this thesis focuses on the need to understand the GA process under this combined settings while investigating different operators that can be integrated with different EAs .

5.4 Summary and Final Remarks

This chapter was undertaken to design problem-specific elements of a Genetic Algorithm (GA) to tackle a Workforce Scheduling and Routing Problem (WSRP) and evaluate their offline tuning. An indirect chromosome encoding scheme was designed to produce and maintain feasible solutions throughout the evolutionary process. In addition to four crossover and four mutation operators taken from the literature, four crossover

operators and two mutation operator were designed for the subject problem. Hence, eight crossover operators and five mutation operators were implemented in this study.

Comprehensive computational experiments were conducted using various settings of crossover rate, mutation rate, and population size, for each combination of one crossover operator and one mutation operator.

Initially, a total of 35 real-world problem instances, grouped into five sets, were used for to identify the recommended GA settings for each set of problems. The results from this experiment showed that the indirect chromosome encoding is effective in maintaining the feasibility of solutions.

Regarding the proposed search operators, CGX and FCX helped the GA to perform well depending on the size of the problem instances. Among the mutation operators implemented, the FCF mutation made a significant difference improving the performance of the GA. These results support the idea of having problem-specific operators to help the GA to converge to better solutions. Additionally, this chapter identified the best parameters for each WSRP instance. Therefore, the recommended settings were used in further experimentation.

Following the experiments on the GA settings, additional experiments with longer computation times were executed using the identified GA settings. Also, another set of 7 problem instances was used, these were the largest of all problems. Results obtained with the proposed baseline GA were compared to published results. The comparison showed that the GA is capable of finding some best-known results for the larger problem instances and matching the best-known results for some of the smaller ones.

The proposed baseline GA, aided with problem-specific components, is an effective evolutionary approach to solve the highly-constrained WSRP scenarios considered. Nevertheless, results have shown that there were still improvements to be made on the performance of this baseline GA when compared to other established algorithms, particularly for complicated instances. For example, experiments in this chapter required a large amount of time to find the best operators and rates for this algorithm. Thus, adaptive parameter control is suggested in the next chapters to adjust parameters for WSRP instances automatically.

Chapter 6

Adaptive Genetic Algorithm to Solve WSRP

6.1 Introduction

As mentioned at the end of the last chapter, a baseline GA for WSRP was proposed, and the best set of parameters were identified. Despite its success in obtaining good solutions, the GA performance was limited by using a conventional parameter tuning method, which was impractical and computationally expensive. The offline tuning of parameters might be valid before the run; however different parameters values might be better at other stages in the evolutionary process [47]. Thus, tuning an algorithm to a specific instance of a problem may not perform well on different problem instances.

Another concern raised by the previous chapter was the GA tendency to spend more time in obtaining solutions for WSRP than the VNS. Thus, it can be recommended to use a faster approach to increase the robustness of the algorithm.

This chapter proposes six adaptive GA variations to control representation independent parameters, particularly operator probabilities and population sizes.The aim is to develop a parameter-less GA to avoid parameter-tuning methods. If the adaptive control approach is applied, it will undertake more WSRP scenarios that can be proposed later.

Researchers have utilised adaptive parameter control to reduce the time spent on selecting the best parameters, especially in real-world applications [47]. This idea has been applied since the start of evolutionary computations field, i.e., the study presented by [34]. Although, there were a few attempts to utilise parameter control in EAs [82, 90, 121, 169], there is still a large number of studies that depended on a fixed set of values when applied in real-world problems settings [179].

A novel adaptive GA was presented by [60] to maintain a diverse population of highly-fit individuals. Crossover and mutation rates were adapted based on the Euclidean distance as well as controlling the tournament size to maintain a flow of healthy population diversity. This method selected more fit and diverse individuals with the adaptive method outperforming the traditional GA.

This thesis adopts the calculations presented by [60, 91]. The reason for selecting their methods is due to its proven capability of adapting quickly for highly multimodal fitness landscapes, as in many real-world problems such as in WSRP.

Another less explored element than the adaptive rates control was the effect of changing population sizes. This method was first introduced by [62]. Followed by a few attempts in the area of dynamic populations. The study by [147] claimed that adaptively re-sizing a population may be a practical extension to the traditional GA. Nevertheless, their method was affected by selection error with random deletion of individuals that failed to converge to the best individuals.

The study by [98] has investigated the effects of five dynamic population re-sizing approaches including random fluctuating population, increasing population, decreasing population, bell-shaped population and inverse bell-shaped population. The proposed method was compared with a fixed size population GA. The study argued that there was no universal, applicable procedure for sizing the populations. Hence, using dynamic population re-sizing was problem-specific.

Adaptive control of the genetic components has never been investigated for WSRP or any related problems. Given that, this chapter presents an adaptive GA customised for tackling the combined setting of scheduling and routing in WSRP. The specific objectives of this chapter are:

- To identify the most effective features that contribute in early convergence.
- To propose different variations of the adaptive parameter control method, concerning the identified features, that aim to enhance the algorithm efficiency.
- To analyze the impact of adaptive parameters on the feasibility of solutions as well as the algorithm speed.

6.2 Diversity-Based Adaptive GA

In a traditional GA, parameters are fixed in which they are used throughout the run time. Adaptive parameter values, on the other hand, are often adjusted to avoid premature convergence. The changes are triggered by the population state at a particular time, i.e., the population diversity or the fitness improvements [47].

This section describes the proposed adaptive aspects of the GA, as an extension of the study proposed in Chapter 5, using a customised solution representation and problem-specific operators in order to maintain solutions' feasibility.

In Chapter 5, the performance of several genetic operators, traditional and problem-specific, was studied in relation to the tailored chromosome representation. Various parameter settings were also investigated for the instances considered. Given that, the initial design of the method proposed in here is led by the observations derived from the previous chapter, that have affected the selection of the parameter setting. These features can be summarised as follows:

1. The operator's performances/parameters were dependent on the problem size (chromosome length).
2. With an increment in the chromosome length, a smaller size population size was required and vice versa.
3. There was an inverse relationship between the population size and the number of generations required.

Accordingly, this section explores three aspects of the adaptive parameter control, search operator rates and population sizes while randomly selecting a crossover function.

All proposed variations share some elements of the GA. First, the indirect representation is designed to include all visit assignments. Thus, the chromosome length varies according to the number of visits assignments $|V|$. For example, if instances are relatively small, the population size M must be large enough to obtain dissimilar individuals in population P. On the other hand, in large instances, there is a less chance to find similar solutions in P. Thus, a small M is recommended. Population diversity has affected some major choices in parameter tuning. Therefore, this chapter intends to propose a diversity-based adaptive GA for WSRP using these features.

Adaptive parameter update process involves multiple adjustments that aim to select the best parameters for the search to delay convergence. Algorithm 12, presents an adaptive parameter control procedure implemented in this chapter, including the process by which the current search information is fed back at every iteration to enable adjustment. First, an initial population of M individuals (one-day plans) is created based on the indirect chromosome encoding that constructs feasible solutions. At the start of each generation, $9M/20$ pairs of parent individuals are selected using binary tournament. Then, a new population is created by the two major process, crossover and mutation.

The suggested crossover, as stated in Chapter 5, is applied with probability P_c to each pair of parents to generate two offspring. Each offspring go through a flat-cost flip (FCF) mutation, with some probability P_m. At the end of each generation t, the population is updated using an elitism strategy. The $M/10$ best individuals from the current population are kept, and along with the $9M/20$ offspring individuals generated, a new population of M solutions is formed.

Operator's performances influence the information feedback process where the adjustment of a parameter is done. At each iteration, if there are no cost value improvements on the best so far solutions for a number of generations; the GA stagnates. Parameters values are updated according to the current diversity, calculated as denoted in equation (3.1), to generate the next population.

Initially, the crossover operator is changed by selecting a random crossover (step 1). Then, the crossover rate P_c and the mutation rate P_m are adjusted (step 2-11). Population re-sizing is done sequentially (step 12-16).

Adaptive rates, P_c and P_m, uses a resetting process (forgetting), where the rates history resets to the lower bound if the value is out of range. Thus, ignoring all the previous feedback process that led to the adapted value inflation. The details of the adaptive parameters implementation are described next.

Algorithm 12 Adaptive Parameter Control

Require: P_c, P_m, CrossoverOperator and M = ⟨ Predefined by the user ⟩
Pick-random := CrossoverOperator
if $P_c > 0.45$ AND $P_c < 1.0$ **then**
    Update P_c according to equation 6.1
else
    Reset P_c
end if
if $P_m > 0.1$ AND $P_m < 0.60$ **then**
    Update P_m according to equation 6.2
else
    Reset P_m
end if
if $M > 100$ AND $M < 550$ **then**
    Update M according to equation 6.6
else
    Update M according to equation 6.7
end if

6.2.1 Search Operators Rates Updates

Initial parameter values are adopted from Chapter 5. Parameter updates calculations are extended from the studies by [60, 91]. The reason for selecting their methods is that they are capable of adapting quickly for highly multimodal fitness landscapes, as in many real-world problems, such as WSRP. The details for parameter rates calculations are described next.

Crossover Parameter Control

Crossover converges a population considering there is no additional diversity added through its implementation [60]. As a result of only recombining existing genetic material, the similitude between individuals increases after a number of generations. Therefore, similar individuals are recombined, *inbreeding* and no additional diversity is added through crossover implementation. In this thesis, a diversity based adaptive parameter control is used to maintain a high level of diverse population to avoid early convergence.

To be able to control parameter values, upper and lower values must be identified. The generated P_c value is a number within the specified range, $Q_1 \leq P_c \leq Q_2$, *where* Q_1 is set to 45% and Q_2 is set to 100%. If P_c is 45%, the new generation is made as exact

copies of the chromosomes from the old population, i.e., no crossover. If P_c is 100%, then all offspring are made by crossover. Individuals that do not undergo crossover require higher mutation probability [60].

$$P_c = \left[\frac{Diversity_g}{Diversity_p} * (Q_2 - Q_1)\right] + Q_1 \tag{6.1}$$

The P_c value is updated according to equation (6.1). Based on the population diversity, P_c is reduced on convergence (low diversity), to increase the chance of exploration in the population. The value of P_c is increased if the population has high diversity value, to encourage exploitation, providing that the diversity value does not exceed $Diversity_p$, where $0 < Diversity_g \leq Diversity_p$.

Population diversity in this chapter is measured from two aspects. *The genotype space*, denote as $Diversity_g$, explained in Section 3.2.7, and *the phenotype space*, denoted as $Diversity_p$, as the population variance, i.e., how far each individual in the population is from the mean value. The two measurements accommodate different views of the loss of diversity. Hence, combining these methods can overcome those difficulties provided by one measurement used in isolation [31].

Mutation Parameter Control

Mutation is usually applied to introduce diversification to the population. Highly fit individuals are subjected to low levels of mutation (minimal disruption), while low-fitness individuals go through low levels of mutation, i.e., large rates of disruptive mutation [60].

The selected range for the mutation rate P_m is $Q_1 \leq P_m \leq Q_2$, where $Q_1 = 10\%$ and $Q_2 = 60\%$. Hence, a minimum of 10% of the population does not go through the mutation to avoid full disturbance. The overall mutation rate P_m is calculated with respect to the fitness improvements given by a mutation and the current diversity state of the population. Thus, the new P_m is the average value of the calculations in equations (6.3) and (6.4) as stated in equation (6.2).

$$P_m = \frac{P_m^{Fitness} + P_m^{Diversity}}{2} \tag{6.2}$$

Equation (6.3) calculates the mutation rate with respect to the diversity. A diverse

population ensures slow convergence; hence more individuals are explored. However, a highly converged population requires more mutation implementations. The value of P_m is equal to Q_1 when the population is maximally diverged, i.e., $Diversity_p = Diversity_g$.

$$P_m^{Diversity} = \frac{Diversity_p - Diversity_g}{Diversity_p} \times Q_1 \tag{6.3}$$

To encourage further exploration, mutation is applied to individuals that exist in low fitness areas. Equation (6.4) generates a value that uses the parent fitness $f(S)$, where $f(S)_{max}$ and $f(S)_{min}$ correspond to the best and worst individuals in the population respectively. The population maximally diverges when the selected parents are the worst individual in the population.

$$P_m^{Fitness} = \frac{f(S)_{max} - f(S)}{f(S)_{max} - f(S)_{min}} \times Q_1 \tag{6.4}$$

In multimodal fitness landscapes, different genotypes can share the same fitness value. Therefore, an average fitness value may be equal to the best fitness value. Thus, the population can be distributed widely. Using the averaged value of the $P_m^{Diversity}$ and $P_m^{Fitness}$ can provide a better feedback for the adaptive change.

6.2.2 Dynamic Population Re-sizing

Preserving diversity is a crucial element for increasing the population search coverage. Another key point to a diverse population is to encourage the algorithm to explore the fitness landscape more [98].

Population diversity is affected by the population size. If the size is not suitable; the algorithm gets trapped in a sub-optimal region of the search space [94]. This is because the algorithm was unable to find better quality solutions as a result of inbreeding.

The point often overlooked is that a population size might be good at some stage of the search but requires a change at different stages in the evolutionary process. Therefore, an adaptive population re-sizing is proposed to lower the chances of becoming trapped in low-quality areas.

The proposed method works as follows. In a population P of a size M, the size is

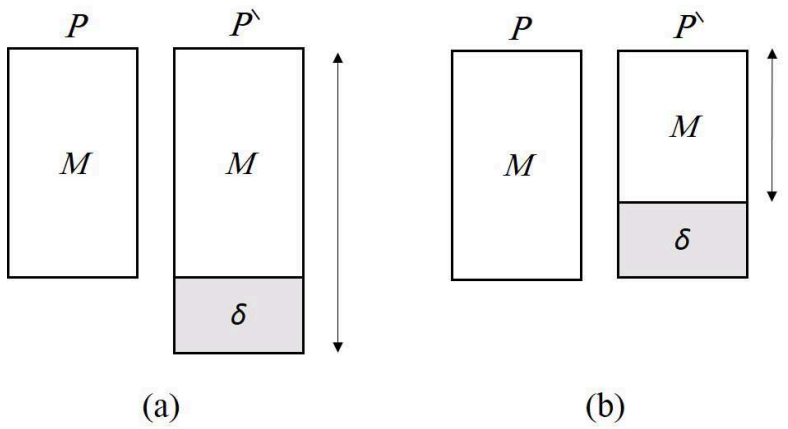


Figure 6.1: Graph illustrations of the dynamic population re-sizing approaches, (a) expanding and (b) shrinking.

changed dynamically during the run when there are no improvements. The deterministic change occurs with respect to a step-size δ, only if the population size ranges between $100 \leq M \leq 550$. The value δ calculates the step-size relative to the level of diversity as stated in equation (6.5). To preserve the best-found solutions, $M/10$ individuals of the population are always kept after sorting the population in a non-descending order.

$$\delta = Diversity_g^{(Current)} \times M \tag{6.5}$$

Two methods of population re-sizing are adopted. The first case is increasing population that allows adding a δ number of individuals to the next population as shown in Figure 6.1 (a). The addition occurs by constructing new individuals at the bottom side of the population.

The second case is shrinking population that releases a δ number of individuals from the population as shown in Figure 6.1 (b). The deletion occurs by deleting existing poor individuals.

$$M = \begin{cases} M + \delta & \text{if } Diversity_{Current-1} < Diversity_{Current-4}, \\ M - \delta & \text{if } Diversity_{Current-1} > Diversity_{Current-4}, \\ M & \text{if } Diversity_{Current-1} = Diversity_{Current-4}, \end{cases} \tag{6.6}$$

This method balance between having good solutions and a suitable diversity level. For example, when the population size is small, the diversity is lost between iterations. As a result, adding more individuals, regardless of their fitness value, promotes the diversity and encourages the search.

On the other hand, when the population size is large, more time exists to search for better solutions. Thus, decreasing the population size allows the crossover to focus on dominate areas and henceforth computing the results faster.

$$M = \begin{cases} M + \delta & \text{if } M < 100, \\ M - \delta & \text{if } M > 550, \end{cases} \tag{6.7}$$

The cases of controlling population size are shown in equation (6.6). They are all triggered by an increase or a decrease in the population diversity. If the diversity is maintained, the population size remains the same.

Equation (6.7) ensures keeping the population size closer to the range. Unlike the adaptive rates, dynamic adjustments are discrete and non-linear, to increase the effect of the step-size by making a larger impact on the environment.

6.2.3 Crossover Selection

In addition to the adaptive crossover rate, adaptive mutation rate and adaptive population size, a random crossover exchange is used. This method is based on the success of the reproduction process only to allow exploration of the solution space by using different operators when one operator fails to improve the search. The algorithm might benefit from the performances of a group of crossover functions which are alternated during the execution, rather than using one crossover throughout the run.

In general, an operator is selected in the literature at random [151] or according to rules [177]. In this chapter, an arbitrary crossover is selected, from a group of crossovers functions, if the iteration has no improvements, for the few generations. All moves are accepted until reaching the stopping criteria. Crossovers used are CGX, PMXGreedyX, FCX, PMFCX, 1PX, 2PX and UX.

Different operators require different rates at each stage of the search. Therefore,

further improvements must be applied on the global performance of the whole population by using adaptive crossover rates. The probability of each selected operator is modified at each iteration, according to the population diversity at that iteration.

6.3 Experimental Study and Results

Several adaptive GA implementation evaluated individually based on each adaptive element, i.e., 1) mutation/crossover rates, 2) dynamic population re-sizing and 3) random crossover exchange. Separate implementations provided an understanding of how each adaptive component perform under the WSRP.

Each experimental configuration was denoted in this section as a GA variation. Including adaptive mutation/crossover rates AGA, adaptive rates with dynamic population re-sizing AGADP, dynamic population re-sizing only GADP, adaptive rates with random crossover alterations RCAGA, random crossover with dynamic population re-sizing RCGADP, adaptive rates with dynamic population re-sizing and random crossover alterations RCAGADP.

Table 6.1: Initial parameter settings for WSRP problem sets (before the run).

Parameter	**A**	**B**	**C**	**D**	**E**	**F**
Mutation Rate P_m	50%	50%	30%	10%	10%	10%
Crossover Rate P_C	50%	50%	50%	100%	100%	100%
Population Size M	500	500	500	250	250	250
Crossover Operator	CGX	CGX	PMGreedyX	FCX	FCX	FCX

Table 6.1 shows the initial parameter values for each WSRP problem set *before the run*, as suggested by Chapter 5.

In addition to the parameter values, methods that excluded the random crossover exchange, i.e., AGA, AGADP and GADP, have used one crossover operator throughout the run, as shown in Table 6.1. Otherwise, a crossover was selected at random, when the algorithm was stuck at a local optimum.

During the run, crossover rates P_c, mutation rates P_m and population sizes M were updated on stagnation, with respect to the population diversity as explained in Section 6.2. Parameter ranges were empirically set due to preliminary tests as follows:

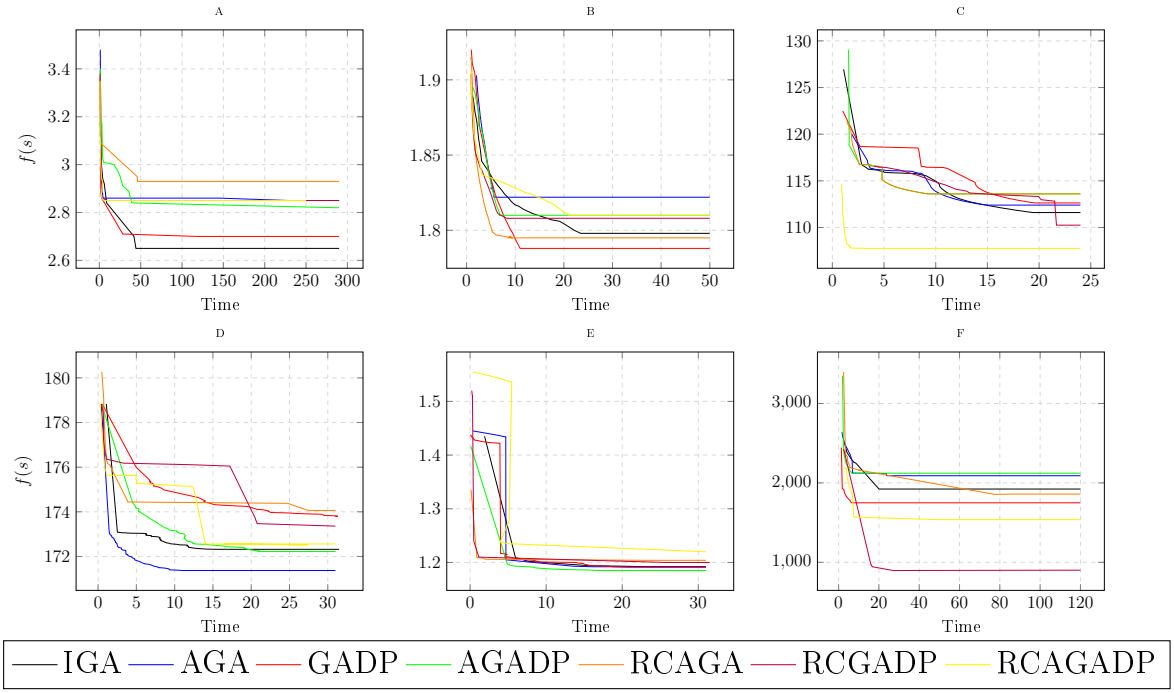


Figure 6.2: Change in cost values $f(s)$ over time (in seconds) shown for methods IGA,AGA, GADP, AGADP, RCAGA, RCGADP and RCAGADP.

mutation rates $P_m \in [0.10, 0.60]$, crossover rate $P_c \in [0.45, 0.100]$ and the population size $M \in [100, 550]$. The methods used to evaluate the results, are as explained in 4.3.

6.3.1 Performance of the Adaptive GA in Time

A common method for analysing the performance of a GA was by performing several runs and plot the average best cost values over time. Thus, all adaptive methods were implemented, with several runs, in addition to the baseline GA (IGA), presented in Chapter 5.

Each line in Figure 7.4 shows the change in the cost values over time, for problem sets A to F. The IGA was plotted in black lines; the AGA was plotted in blue lines, GADP was plotted in red lines, AGADP was plotted in green lines, RCAGA was plotted in orange lines, RCGADP was plotted in purple lines and RCAGADP was plotted in yellow. The lower the line, the better the cost value.

For A, B and C, a high initial value of the mutation rate P_m have delayed convergence in all methods, even when using IGA, as presented in the top plots on Figure 6.2.

However, there was a tendency to result in sub-optimal solutions. For Example, when GADP and RCGADP were used, the methods were able to escape local optima in an early stage of the search but can still become trapped at later generations. Adaptive mutation rate ensures convergence delay, especially in smaller instances as a result of a wider range of P_m values.

From the bottom plots in Figure 6.2, it can be seen that on problem set D, the IGA outperformed the adaptive methods except for AGA. This indicates that AGA has a faster response to the environment change. On the other hand, on problem sets E and F, the algorithms have stagnated faster than on smaller instances. Nevertheless, methods with dynamic population re-sizing have managed to converge to slightly better results on E and with a large margin on F.

The dynamic re-sizing variations, with the help of crossovers exchange allowed the crossover to progress in convergence with high levels of diversity for some instances by adding or eliminating individuals from the population. Combining the adaptive rates and crossover exchange on the other hand was applied when the population has fully converged to introduce diversity to the population. Further analysis is detailed next.

6.3.2 Effect of the Adaptive Control on the GA

Search Operators Rates

The number of P_c and P_m updates were recorded to analyse these parameters behaviour for each WSRP instance. To do so, the average parameter value of the adaptive settings over each iteration were presented for each problem instance. Figures 6.3 and 6.4 illustrates a group of box plots comparison between methods that utilised crossover rate P_c and the adaptive mutation rate P_m, i.e., AGA, AGADP, RCAGA and RCAGADP for WSRP scenario A, B, C, D, E and F respectively. The average change in the adaptive rate was shown for each method.

A change in the rate value means that the adaptive function has been triggered by a change in the current population state. A large box means widely distributed and diverged values. A small box means narrow distributed and similar values.

The box plots shown in Figure 6.3 reveal that P_c values were within the range

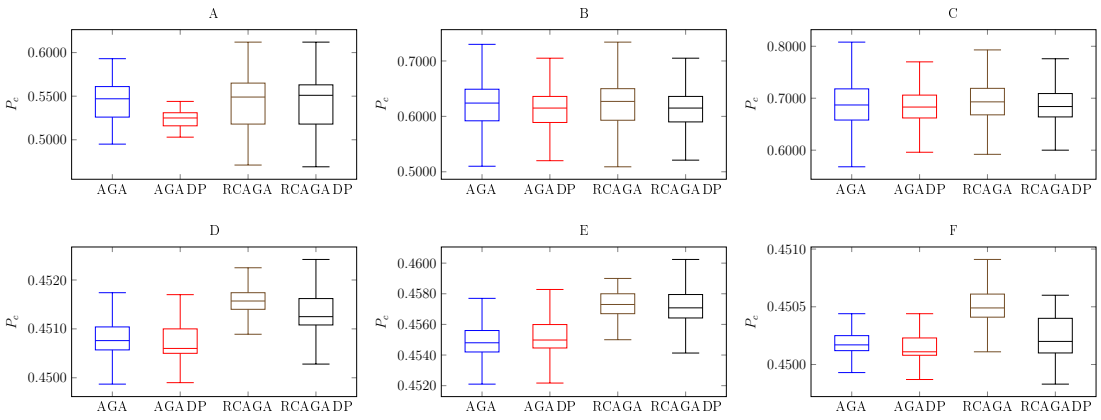


Figure 6.3: Box plots comparison of the average crossover rates P_c range, shown for methods AGA (blue), AGADP (red), RCAGA (brown) and RCAGADP (black).

$[0.45, 100]$. More distributed values were reported in A, B and C with updated P_c that ranges between $[0.48, 0.64]$, $[0.5, 0.73]$ and $[0.56, 0.80]$ respectively. On the other hand, P_c has a marginal change in D, E and F with values ranges between $[0.44, 0.45]$, $[0.45, 0.46]$ and $[0.44, 0.45]$ respectively. This observation was related to the population diversity effect on the adaptive rates. In problem sets A, B, D, E and F the random crossover exchange allows more changes in the P_c value with a wide range of different rates. However, in problem set C, AGA has obtained more adjustments with a wider range of P_c values. More information regarding this is explored in this chapter.

The P_c value was calculated based on the population diversity. Initially, in smaller problems, the population diversity was relatively lower than its variance. Hence, the change in the P_c value was large. However, the algorithm stagnated faster due to the similarities between solutions. As explained in the previous sections, smaller chromosome size allowed faster mixing while using the search operators. Therefore, more crossovers were applied with adaptive changes. This also explains the wider distribution of the rate values on smaller WSRP instances.

For large problems, the population diversity value was higher than the variance. At the same time, larger instances have long chromosomes, which required more time to apply a crossover. As a result, fewer crossovers were applied with adaptive rates that had tighter deviation. Hence, the P_c value was decreased gradually.

As it can be seen from the plots in Figure 6.4, P_m values were within the range

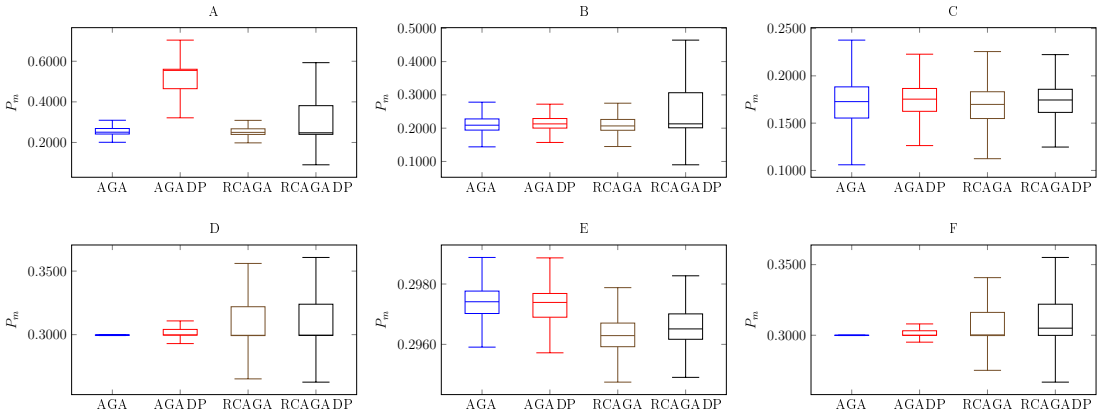


Figure 6.4: Box plots comparison of the average mutation rates P_m range, shown for methods AGA (blue), AGADP (red), RCAGA (brown) and RCAGADP (black).

$[0.09, 0.60]$. Similar to the observations derived from Figure 6.3, more distributed values were reported in A, B and C with updated P_m that ranges between $[0.10, 0.60]$, $[0.10, 0.46]$ and $[0.10, 0.23]$ respectively. On the other hand, box plots were narrower in D, E and F with values. The ranges were between $[0.26, 0.36]$, $[0.29, 0.30]$ and $[0.26, 0.35]$ respectively.

In smaller instances, when the diversity levels were low, more disturbances were required to ensure efficiency of the algorithm. Thus, more mutations were applied. However, in larger problems, the diversity levels were very high. Because of that, fewer rates adjustments were utilised in D, E and F.

The computation of the adaptive P_m value was time-consuming. Fewer generations were used as a result of the time spent in updating the rates. For smaller problem sets, the solution was found very fast as it will be discussed in the following sections. However, in larger problem sets, the algorithms were forced to stop before exploring other areas of the search, due to the time limitations. Thus, less P_m values updates were established in large problem sets.

Similar to the P_c values updates, in problem sets A, B, D, E and F, the random crossover exchange encouraged more change in the P_m values, especially in larger problem sets. Also, it has managed to converge to better results than AGA. This was because that the variation of the crossover function required an adaptive tuning of all rates. Hence, more changes occurred in those problem sets.

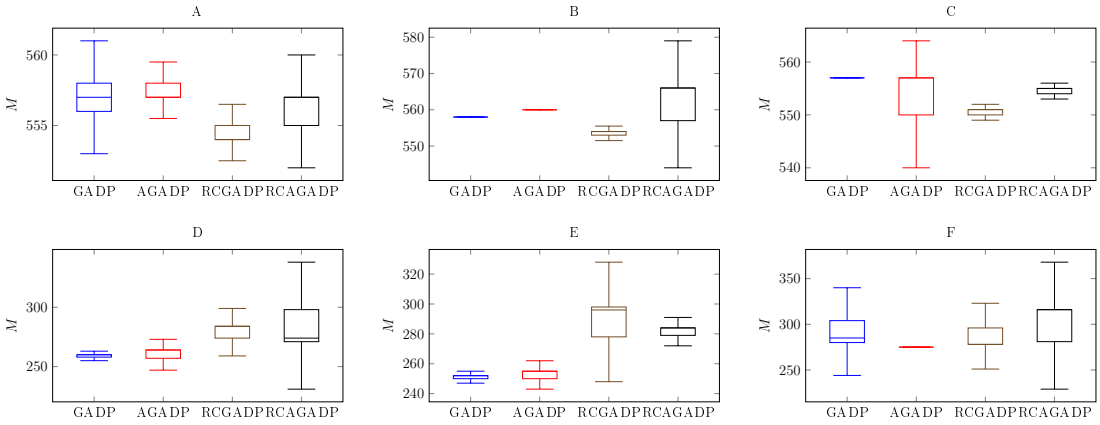


Figure 6.5: Box plots comparison of the average population sizes M range, shown for methods GADP (blue), AGADP (red), RCGADP (brown) and RCAGADP (black).

In problem set C, AGA has obtained more adjustments with a wider range of P_c values. Thus, an in-depth search was executed by the AGA to find good enough solutions. Therefore, more adaptive rates updates were established.

Population Size

A group of box plots, shown in Figure 6.5 illustrate a comparison between methods that utilised dynamic population re-sizing for a population of a size M, for WSRP problem sets A, B, C, D, E and F.

The average change in the M values were shown for GADP, AGADP, RCGADP and RCAGADP. The M values ranges between $[552, 560]$, $[550, 560]$, $[540, 568]$, $[230, 340]$, $[243, 328]$ and $[229, 370]$ respectively.

A highly diverged population used more crossover, less mutations and required less individuals and vice versa. This was further shown by Figure 6.5. Different from the adaptive rates, the changes were less in smaller problem sets. This was because adding random individuals was computed slower than deleting existing individuals, as the case in larger problem sets.

Initially, for larger problem sets, M is 250. The bottom graphs in Figure 6.5 show that there was a decrease in M that have occurred in the start of the algorithm. On later generations, the diversity decreased, and therefore there was a necessity to increase the number of individuals.

As previously mentioned, random crossover exchange promoted the diversity further when combined with population re-seizing process, as the case in problem set E. Also, with a full adaptive process, as the case for A, B, D and F. Interestingly, the GADP has a high level of similarity among values in B, C, D and E than the other three methods.

Another observation was taken from Figure 6.5. AGADP required more adjustments on the M value, with a wider range of values in the problem set C. With an increase in the population size, the diversity increases even though a low mutation was utilised in the initial values. This outcome was the result of updating the rates values during the run in the event of premature convergence. It can be argued that maintenance of mutation rates encourages the diversity further along with the population re-sizing. Henceforth, using adaptive rates combined with dynamic population re-sizing methods was capable of adapting quickly to the environmental change by maintaining a diverse population in the later generations.

Population Diversity

The diversity trends were recorded under all methods applied. Figure 6.6 presents 12 sub-plots, each plot represents the average diversity change for a problem instances. Each line in the sub-plot represents a GA variation as follows. IGA (black), AGA (blue), AGADP (light red), GADP (green), RCAGA (orange), RCGADP (dark red) and RCAGADP (yellow).

The top 6 plots reveal that the diversity was quickly converged by all algorithms in the initial generations with overlapping lines. Despite the fact that the initial diversity was decreased for all methods, in A and B, there was an apparent change in the diversity value while using GADP, RCGADP and RCAGADP. This result was achieved by the disturbance caused by the dynamic population re-sizing that resulted in having dissimilar individuals. Thus, a closer inspection of the values, in generations between 100 and 400, was displayed in Figure 6.6.

Adaptive rates, i.e., AGA and RCAGA exhibited loss of diversity and premature convergence at local optima in A and B with an apparent change in the problem set C while using RCAGADP, indicating that rates required more time to be adjusted.

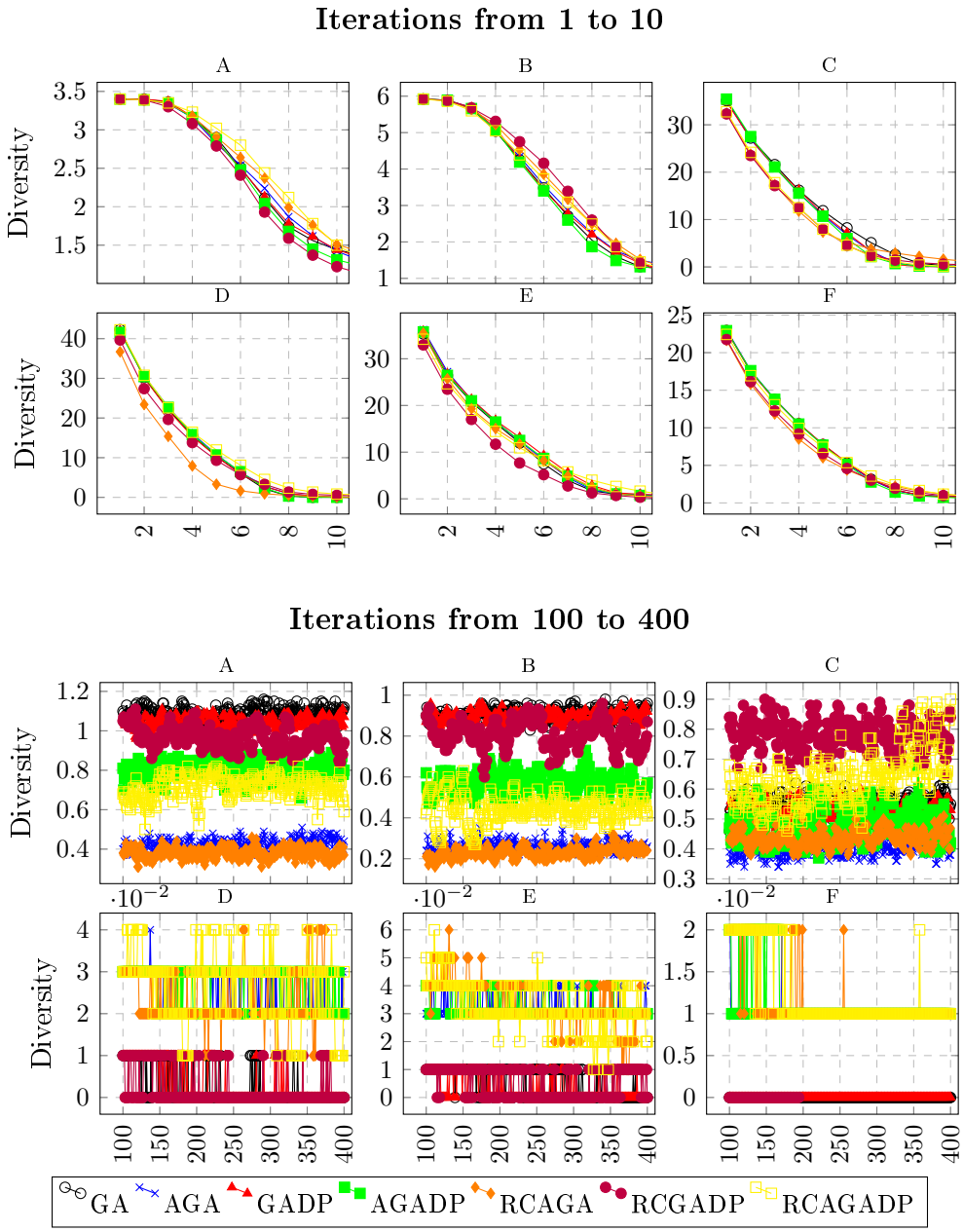


Figure 6.6: Change in diversity, shown for methods GA, AGA, GADP, AGADP, RCAGA, RCGADP and RCAGADP.

Dynamic population re-sizing methods were capable of adapting quickly to environmental change in comparison to methods using adaptive rates in smaller instances. This was the outcome of increasing the diversity by adding new individuals in the later generations that resulted in converging to better results as it will be explained in the next section.

From the bottom plots in Figure 6.6, it can be seen that there was a high level of intensification in the later generations. However, when random exchange crossover was used in RCAGA and RCAGADP, the diversity was promoted at all stages, in a counterproductive way, where high exploitation was needed.

Adaptive rates have also encouraged the search by increasing the diversity even though a low mutation rate was utilised as initial values. This outcome was the results of updating the rates values during the run in the event of convergence, as previously explained in Section 6.2. Even though dynamic re-sizing has succeeded in increasing the diversity in RCAGADP, GADP and RCGADP obtained a loss in the diversity as a result of a low mutation rate. Thus, it can be argued that the maintenance of mutation rates encourages the diversity further in larger problem sets along with the random crossover exchange.

6.3.3 Baseline GA vs. Adaptive Control Methods

To establish statistical significance in EAs, the study by [58] recommended the use of Friedman analysis as a non-parametric statistical test that was used to test for differences between groups when the dependent variable being measured was ordinal. This test was used in this thesis to compare between different methods, to test if there was a significant difference between compared methods or not and finally rank these methods.

A two-way analysis was used to compare the variances of seven related-samples, with a significant level of $\alpha = 0.05$ and 95% as a confidence interval. A total 42 problem instances were used to set the sample size of one method as large as possible. This increases the probability of accepting or rejecting the null hypothesis.

Table 6.2 provides the results generated by the Friedman analysis: the mean value, the standard deviation, the minimum cost-value, the maximum cost-value and the mean-

rank. The results presented in the mean-rank column show the methods order based on the statistical analysis, where a low rank indicates the best method while the high rank indicates the worse method ranked overall.

Three performance measurements matrices: **Dev%**, **Best** and **Score**, explained in Chapter 4, were also used in this section. An additional value presented in Table 6.2 was the average gap percentage for each algorithm to the best-known value denoted as the **Gap%**. The results in Table 6.2 had a significance level = 0.000, degrees of freedom = 6 and $\chi^2 = 55.507$. This indicates that there was a clear evidence of a statistically significant difference between all methods.

Table 6.2: Statistical results comparison generated by a non-parametric Friedman's test combined with performances metrics

METHOD	Mean	STD Deviation	Min	Max	Mean Rank	Gap%	Dev %	# Best	Score
IGA	332.49	785.87	1.42	3980.44	5.71	0.01	0.54%	0.26	0.56
AGA	289.00	696.68	1.18	3495.99	**3.25**	**0.00**	**0.36%**	**0.52**	**0.73**
GADP	292.40	705.05	1.18	3497.99	3.57	0.00	0.50%	0.36	0.59
AGADP	289.95	699.05	1.18	3496.18	3.26	0.00	0.39%	0.38	0.71
RCAGA	288.23	692.57	1.18	3500.00	3.87	0.01	0.72%	0.33	0.54
RCGADP	307.59	750.52	1.19	3649.19	4.37	0.02	1.71%	0.36	0.47
RCAGADP	**287.97**	**691.86**	1.21	3496.18	3.96	0.01	0.66%	0.33	0.61

Bold text refers to the best result.

Note that the algorithms applied in this chapter have obtained similar minimum values except for IGA, RCGADP and RCAGADP. This outcome further proves that there exists a loss of performance in the uniform crossover exchange. The result for AGA was promising by obtaining the lowest mean rank and the highest score.

RCAGADP, on the other hand, obtained the lowest mean value and the smallest standard deviation value. Result indicates that the dynamic population re-sizing and the random crossover exchange was more effective for WSRP scenarios only when combined with adaptive rates. The AGA obtained the best overall results statistically because of its fast response to any environmental change. Therefore, it finds better solutions more frequently than other adaptive methods.

On the other hand, IGA (not-adaptive) has presented really poor results in comparison to adaptive variations. Hence, all adaptive methods proved to be better methods for improving the efficiency of the GA in WSRP.

To examine where the differences occur, an additional post-hoc analysis was imple-

mented. The Wilcoxon signed ranks test was used as a non-parametric test to compare two sets of data that come from the different combinations of methods. These test analyses the algorithms by using a ranking based approach of the differences between methods independently. Hence, this test was influenced by the number of instances used.

The Bonferroni adjustment was used in post-hoc tests on the results obtained from the Wilcoxon, to perform multiple comparisons and avoid declaring non-existing significant. Thus, the new significance level was altered to $0.05/7 = 0.007$, *where* 7 was the number of the methods compared.

This means that if $p < 0.007$; the distribution of a method i was different across the values of a method j, henceforth rejecting the null hypothesis by being statistically significant. On the contrary, if $p > 0.007$; the distribution of a method i was the same across values of a method j, therefore retaining the null hypothesis by having no statistical significance.

Table 6.3: Cost-values grouped by GA methods generated by a non-parametric Wilcoxon sign rank test, where S+ (S -) means that method i is significantly better (worse) than method j within a confidence interval of 95%.

METHOD	Z	Sig. (2 tailed)	Sign	METHOD	Z	Sig. (2 tailed)	Sign
AGA - IGA	-4.64	0.00	S+	AGADP - GADP	-2.35	0.02	S -
GADP - IGA	-4.66	0.00	S+	RCAGA - GADP	-1.72	0.08	S+
AGADP - IGA	-4.60	0.00	S+	RCGADP - GADP	-3.48	0.00	S -
RCAGA - IGA	-3.90	0.00	S+	RCAGADP - GADP	-0.28	0.78	S+
RCGADP - IGA	-2.86	0.00	S+	RCAGA - AGADP	-1.46	0.14	S+
RCAGADP - IGA	-3.37	0.00	S+	RCGADP - AGADP	-3.47	0.00	S -
GADP - AGA	-2.68	0.01	S -	RCAGADP - AGADP	-1.44	0.15	S+
AGADP - AGA	-0.60	0.55	S -	RCGADP - RCAGA	-2.12	0.03	S -
RCAGA - AGA	-2.28	0.02	S+	RCAGADP - RCAGA	-0.19	0.85	S+
RCGADP - AGA	-3.35	0.00	S -	RCAGADP - RCGADP	-1.15	0.25	S+
RCAGADP - AGA	-2.26	0.02	S+				

Table 6.3 shows the Wilcoxon signed-rank test. Column Z shows the z-ratio as the unit for the normal distribution, column Sig. (2-tailed) shows the p value and column sign shows the statistical comparison of the null hypothesis h_0 that implies the significant difference between two methods. The S+ (S-) signs mean that a method i was significantly better (worse) than a method j, within a confidence interval of 95%.

In comparing the two related methods, it was apparent that IGA was outperformed by all adaptive methods and therefore shows the merit of the adaptive approach on

improving the IGA performance on WSRP instances.

The Wilcoxon signed-rank test showed that AGA, GADP and AGADP have a statistically significant advantage over RCGADP with existing (Z = -3.353, p = 0.0008), (Z = -3.476, p = 0.0005) and (Z = -3.473, p = 0.0005) respectively. However, there was no statistically significant difference across other methods. On the other hand, AGA, GADP and AGADP were worse than RCAGA and RCAGADP with no significant difference.

The full adaptive GA provided the best results for WSRP, where RCAGA was worse than RCAGADP with no significant difference. The AGA performance was enhanced further when combined with other adaptive aspects.

6.3.4 Computational Results of Adaptive Control Methods

This section describes the performance of each proposed variation with respect to the overall cost values for each instance, the computation time and the percentage gap to the best-known values.

Table 6.4 presents the results obtained with the proposed six variations, compared against the IGA, which was proposed in Chapter 5.

For each problem instance, the table shows the solution quality $f(S)$ obtained by each method and the computation-time in seconds Cpt, in which the solution by the method was found. For each problem instance, the best-known solution quality value and best computation time were highlighted in **bold**.

On a total of 42 instances, the number of best for IGA, AGA, GADP, AGADP, RCGADP and RCAGADP were 11, 22, 16, 15, 14, 15 and 14 respectively. This indicates the AGA has obtained the best values in 52.4% of all instances. More details on each problem set were discussed next.

Smaller instances have benefited from methods that utilised random crossovers exchange and dynamic population re-sizing. Adding new individuals has encouraged the diversity while random crossover exchange has managed to shift the search into new areas that were previously undiscovered. Hence, offspring were dissimilar, with the algorithm's ability to bring new genes into the population. Thus, boosting its performance. However, the three aspects of the adaptive methods were needed for A but not for B instances.

Table 6.4: Cost-values $f(S)$ and computational-time Cpt (in seconds) for WSRP instances, produced by the baseline GA (IGA) and six diversity-based adaptive GAs on all problem instances.

		IGA		AGA		GADP		AGADP		RCAGA		RCGADP		RCAGADP	
		f(S)	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*
A	1	3.49	4.50	3.49	14.45	3.49	4.52	3.49	4.69	3.49	2.43	3.49	2.96	**3.49**	**1.82**
	2	2.49	20.64	2.49	100.69	2.49	99.75	2.49	81.59	**2.49**	**2.42**	2.49	5.17	2.49	3.09
	3	3.14	169.25	3.08	11.50	**3.02**	**184.85**	3.13	47.12	3.13	158.73	3.03	274.01	3.02	206.43
	4	1.42	238.51	1.42	247.07	1.42	137.21	**1.42**	**15.23**	1.42	243.00	1.42	89.00	1.42	81.06
	5	2.42	1.28	2.42	1.11	2.42	0.96	2.42	0.89	2.42	0.96	2.42	1.52	**2.42**	**0.58**
	6	3.55	90.00	3.56	130.22	3.55	16.02	3.55	10.74	3.55	48.97	**3.55**	**8.64**	3.55	24.70
	7	3.71	0.94	3.71	0.72	3.71	0.70	3.71	0.66	3.71	0.75	3.71	0.72	**3.71**	**0.65**
B	1	1.74	449.01	1.75	84.00	**1.72**	**100.54**	1.74	113.38	1.74	284.26	1.73	59.52	1.74	274.61
	2	1.75	0.80	**1.75**	**0.56**	1.75	0.82	1.75	2.06	1.75	0.61	1.75	0.64	1.75	0.95
	3	1.80	444.72	1.82	280.35	1.82	317.76	1.81	369.56	1.79	361.33	**1.79**	**352.34**	1.80	229.30
	4	2.08	118.16	2.08	46.19	2.08	15.49	2.08	5.83	**2.08**	**1.98**	2.08	7.73	2.08	4.08
	5	1.92	328.17	1.93	98.85	1.91	249.27	1.93	144.05	1.91	492.17	**1.90**	**434.04**	1.92	266.73
	6	1.66	353.99	1.66	190.74	1.65	283.56	1.66	184.20	1.66	13.17	**1.64**	**280.54**	1.65	244.94
	7	1.81	315.89	1.81	154.78	1.80	233.83	1.80	168.27	**1.80**	**130.26**	1.80	149.92	1.81	239.79
C	1	115.01	846.87	115.07	703.71	**114.82**	**747.74**	114.87	710.21	114.87	700.69	116.52	821.82	116.77	812.12
	2	3.15	0.46	3.15	0.46	3.15	0.65	3.15	0.52	**3.15**	**0.42**	3.15	0.69	3.15	0.50
	3	105.07	657.11	105.08	677.45	105.09	722.49	105.33	782.01	**104.96**	**648.85**	105.57	814.91	104.97	698.98
	4	11.15	2.41	11.15	2.40	11.15	2.37	11.15	3.99	**11.15**	**1.87**	11.15	3.01	11.15	2.88
	5	12.41	3.38	12.60	33.70	12.47	5.25	12.53	3.73	12.41	224.04	**12.34**	**2.93**	12.41	177.26
	6	180.99	591.22	180.99	589.30	180.99	572.16	180.99	582.22	180.93	590.61	**180.92**	**321.51**	180.93	800.68
	7	**4.30**	**0.00**	**4.30**	**0.00**	**4.30**	**0.00**	**4.30**	**0.00**	**4.30**	**0.00**	**4.30**	**0.00**	**4.30**	**0.00**
D	1	171.21	3414.46	170.77	2978.51	171.02	3069.34	171.02	3271.25	171.34	2789.97	173.84	2422.53	**170.77**	**1663.00**
	2	168.44	3456.53	166.87	3228.52	167.56	3404.48	**166.25**	**3044.11**	176.68	3600.00	168.25	3378.45	176.98	3837.26
	3	180.92	3473.74	**180.60**	**3258.30**	181.48	3358.28	180.92	3323.86	181.56	2375.40	182.31	3508.37	181.87	2656.32
	4	169.80	3438.04	167.93	3465.12	170.24	3446.85	**168.74**	**2964.61**	172.83	2881.37	181.44	3600.00	169.13	3576.84
	5	161.80	3499.82	**161.73**	**3367.04**	161.86	3317.99	162.17	3156.39	161.81	2084.21	163.19	2338.83	165.14	3105.46
	6	178.41	3532.27	178.28	3200.96	178.28	3326.83	**178.03**	**3313.44**	178.54	2630.91	178.74	2631.11	178.81	3081.79
	7	178.77	3465.33	**178.40**	**2489.11**	178.83	3365.64	178.58	2998.59	178.85	2391.55	178.72	2699.83	178.92	5587.61
E	1	1.18	3435.61	1.18	3322.13	1.18	3292.45	1.18	3394.95	**1.18**	**3210.88**	1.19	2481.88	1.21	3426.49
	2	1.21	3512.79	1.21	3368.60	1.22	3280.13	1.21	3491.71	1.27	3615.84	1.23	2040.55	**1.21**	**1258.55**
	3	1.22	3536.23	1.21	3246.99	1.22	3462.66	1.21	3322.91	**1.21**	**2227.97**	1.24	3153.35	1.26	3292.24
	4	1.30	3511.84	1.29	3492.21	1.30	3486.28	**1.29**	**3480.68**	1.30	3445.58	1.35	3558.00	1.30	2665.07
	5	2.25	3460.75	2.24	3442.02	2.25	3347.91	2.24	3516.49	2.29	3592.91	2.26	2574.86	**2.24**	**2440.54**
	6	1.30	3350.12	1.30	3283.59	1.30	3459.45	1.30	3089.03	1.30	3062.06	1.30	3308.82	**1.30**	**2898.00**
	7	1.73	3502.10	1.73	3473.85	1.73	3421.94	1.73	3383.05	1.73	2044.52	1.75	3328.29	**1.73**	**2030.66**
F	1	2121.78	28628.61	2122.60	27647.63	2134.38	28204.99	2120.47	28456.45	**2029.05**	**28433.97**	2543.64	33148.07	2036.26	19988.47
	2	2134.17	28418.67	**2082.79**	**26318.27**	2199.13	28082.75	2123.65	27926.75	2092.49	20093.64	2221.52	23817.94	2088.17	24745.08
	3	608.83	28591.22	**607.26**	**27722.40**	608.76	25998.95	607.95	27170.44	609.21	19776.63	615.03	21935.82	619.77	28488.47
	4	1330.58	28256.37	1328.51	27229.31	1331.02	27632.07	**1327.45**	**27374.00**	1342.16	23617.59	1347.65	28800.00	1329.46	14089.44
	5	198.33	28667.17	**196.08**	**26934.63**	198.02	28113.68	196.71	26871.89	198.90	24010.24	205.71	10808.92	198.08	13129.58
	6	625.77	28181.44	624.77	25367.39	627.08	27582.25	**624.21**	**26629.60**	637.16	25892.93	632.53	18025.85	628.41	12428.78
	7	3497.86	28262.44	**3495.99**	**27225.77**	3497.99	28129.50	3496.18	26003.34	3500.00	15089.69	3649.19	25911.18	3496.18	7450.12

Bold text refers to the best result.

In C instance set, there was a tendency to premature convergence at a sub-optimal solution due to its mix instances sizes, under a similar time limit to A and B. The dynamic population was only required at C_5 and C_6. Adaptive rates and crossover exchange were being applied when the population has fully converged to introduce diversity to the population to enhance the overall performance.

On the contrarily of smaller instances (A, B and C), crossover exchange tended to have a negative effect on the fitness values in problem sets D and F. This observation was more apparent in Table 6.4, where there was a difficulty in obtaining good solutions, due to the time limitations that resulted in fast convergence.

Adaptive rates have encouraged the GA to converge faster without the loss of time and any disturbance in the explored area. To succeed in exploring the dominant peak,

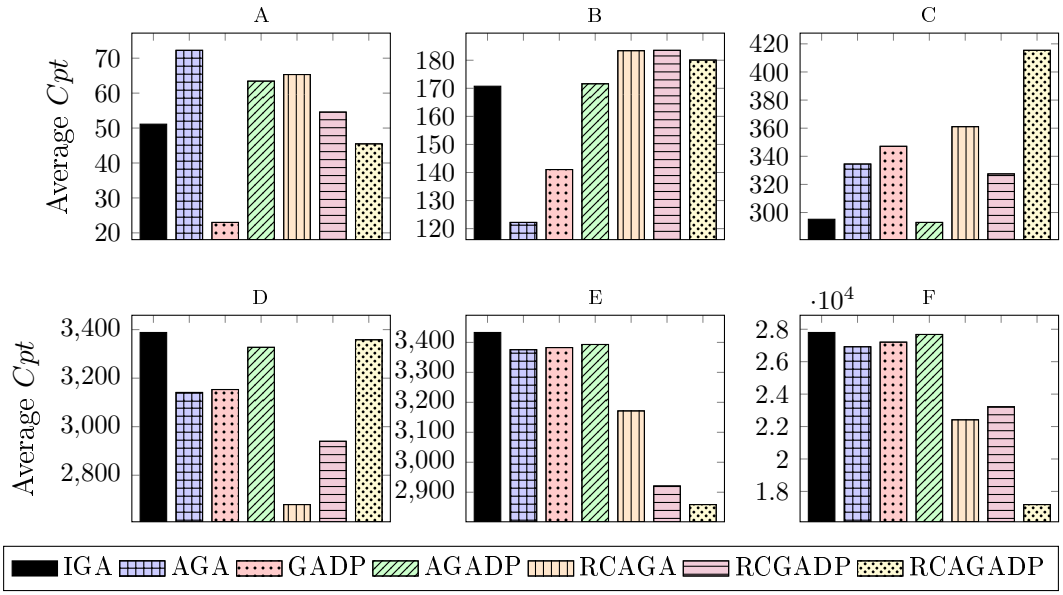


Figure 6.7: Average computation-time (in seconds) used in solving WSRP instances shown for methods IGA (black), AGA (blue), GADP (red), AGADP (green), RCAGA (brown), RCGADP (purple) and RCAGADP (yellow).

whether population re-sizing was used or not, adaptive rates must be used to avoid premature convergence, especially in D and F problem instances.

What was interesting about problem set E, RCAGA and RCAGADP had obtained the best values for these instances. Of course, problem set E have large instances, but the random crossover process was able to find solutions in the given time when combined with adaptive rates.

Nevertheless, there was a negative correlation between dynamic population re-sizing and the instances size. In larger instances, the GADP was unable to converge to good solutions by updating the population size. The adaptive population updates were computationally expensive in these instances. This was because the GADP used the time given to find the best size for the population, rather than searching for better solutions. This means that to provide better adaptive performance, in these instances, dynamic population re-sizing must be excluded or the running time needs to be increased.

Figure 6.7 illustrates the average computation-time in seconds as stated in Table 6.4. Each sub-figure corresponds to a problem set. Each sub-figure presents an average computation-time used by a method. The IGA (solid black bar), AGA (blue grid bar),

GADP (red dotted bar), AGADP (green with north east lines bar), RCAGA (orange with vertical lines bar), RCGADP (pink with horizontal lines bar) and RCAGADP (yellow with crossed dots bar).

All these methods were applied, and therefore their overall average computation-time was plotted. The lower the bar, the better, i.e., the less computation-time.

For problem set A, AGA has spent more time than other methods. Due to the time spent for adapting the mutation/crossover rates when its not necessary. Likewise, on problem set C, RCAGADP has spent more time on computing the results. Then again in problem set B, RCAGA and RCGADP have spent more time in computing the results, but they have converged to good solutions as previously explained. Results indicate that changing the crossover function delayed the convergence, and henceforth allowed to search for better solutions.

As it can be seen, the IGA has the largest computation-time among all methods for problem sets D, E and F. Followed by the GADP that obtained good-enough results on problem set E, but with larger computation-time in comparison to the adaptive methods. On problem sets D and F, GADP obtained poor results due to the time spent tuning the population that resulted in premature convergence.

In general, the time executed to tune the algorithm can lead to more disturbance in the solutions. Therefore, a balance between population diversity and solutions quality must be taken into consideration to evaluate the performance of a GA method for WSRP.

6.4 Summary and Final Remarks

Understanding the effect of different static parameter setting of various components for a given problem is not straightforward task [47]. Thus, it can be difficult for real-world problems, such as WSRP. This chapter explored possible elements that influence the baseline GA performance degradation in later generations.

Although a setting value can be good in the initial stages of the search, adjustments were required when shifting the search into different areas. Hence, the second aim of this study was to investigate the effects of using adaptive parameter tuning techniques in enhancing the performance of the GA for WSRP.

Three adaptive adjustments were considered: mutation/crossover parameters, dynamic population re-sizing, and random crossover exchange. Six GA variations were proposed in which they were triggered by stagnation, i.e., no improvements on the cost values. Different adaptive variations were suited for different WSRP scenarios and all methods proposed had outperformed the baseline GA.

Maintaining a diverse population was important for increasing the search coverage. Dynamic population re-sizing methods achieved the highest levels of diversity, especially in small instances, with fitness landscape change throughout the generations. This aided in obtaining good solutions quickly. Nevertheless, being limited to perform well only on those instances, it lacks fast adaptivity aspects in regards to the large problem sets.

The key strength of this study was adapting operators probabilities. This method has provided better results for WSRP scenarios, with the highest number of best solutions, especially in D and F, by avoiding premature convergence.

Multiple settings analysis revealed that the random crossover exchange allowed the recombination process to continue in finding better solutions, especially in A, B, C and E instances. In contrast, random crossover exchange was computationally expensive for D and F and unable to converge to good cost values. As an outcome of the time spent changing the crossover functions. Despite this method success in obtaining good results, its performance was statistically inferior to AGA, GADP and AGADP. An implication of this was the possibility to improve the crossover selection. More work was needed to advance the GA efficiency by achieving a good balance between population diversity and crossover selection. This will be presented in the next chapter.

Chapter 7

Multiple Crossover GA to solve WSRP

7.1 Introduction

Chapter 5, identified the best performing operators for WSRP. Chapter 6 extended this work by including an adaptive control method for mutation/crossover rates, population size and random crossover exchange during the run. The reason for using the later was because there was no evidence that the performance of one crossover operator was superior to the performance a group of crossover operators.

Using a learning method to adjust parameter values has proven to enhance the baseline GA performance in the previous chapter. Nevertheless, selecting a random crossover, without any prior knowledge of its efficiency, was a straightforward procedure. It is still not clear which operator was the best for each problem set, during the run. Further improvements were suggested.

In this chapter, a learning-based adaptive multiple crossover framework (AMC) is proposed to tackle the WSRP. According to [104], multiple operators GAs have been used on real-world applications to benefit from using different synchronised operators. Most researchers have utilised a group of operators, crossovers [26] or mutations [28], as part of the multiple operators algorithm.

This research is centered on the crossover operators rather than the mutation operators, due to the observations derived from Chapter 5, which established the FCF mutation as the best performing operator among all mutations for WSRP.

The study by [134] has investigated mixing eight crossover operators in a GA, for routing problems. One crossover was selected at random, with all the operators having the same probability. This operator was then applied at the current iteration. The study suggested that the multiple crossovers method has obtained better results than a traditional GA. Another study that proposed a multiple operator algorithm for routing problems was presented by [123]. The proposed algorithm has proven to be more efficient than using one crossover with one mutation.

When using a multiple operators framework, two main operations were applied in the literature, operators *selection* and operators *evaluation*.

Adaptive operator selection (AOS) is identified as the online change of a crossover function [11]. Each crossover has an application rate, that is relative to the crossover performance. The better the crossover, the higher the chances to be implemented more. Thus, providing a variety of performances in one iteration. Different AOS methods have been successfully used in the literature, including random operator exchange [123, 134, 151]. In later years, adaptive operator allocation rules were developed as a learning-based operator selection function, such as probability matching [166], adaptive pursuit [161], multiple armed bandit [8], and sliding multiple armed bandit [53, 89].

Operator evaluation process is identified as a method to measure a crossover effectiveness (performance), by analysing its impact on the current search [89]. This includes giving some reward to an operator according to the operator performance. The next crossover is selected based on rewarded value. Best operators are scored higher, and therefore they are selected more. As a method for operators evaluation process, fitness improvement was used, i.e., the better the new solutions, generated by an operator, the more chance of that operator to be applied. However, more performance measurements were explored and proven to be effective, such as distance-based measurement [26] and the execution-time of an operator [104].

Many considerations were taken into account when designing a multiple crossover

GA, including the type of the evaluation measurement used and the number of iterations required to obtain sufficient information on the operators' performance.

Existing research has focused on using a fixed number of iterations to determine "the best" operator. However, an AOS might converge to the best performing operator early in the search. Not to mention that different operators can provide different results at different stages of the search. Thus, using preliminary information can have limited flexibility and leads up to a loss of performance. This observation was proved by the study presented in [11], in which the use of a "dynamic" approach was established to be better than using a static one.

The intended contribution of this chapter is to present an efficient adaptive multiple crossover GA, which uses allocation rules on a mix of problem-specific and traditional crossovers. Best operators are given high application rates, and low rates are given to the worse operators. Based on the learning outcomes, Application rates are dynamically modified, to reflect the crossovers' behaviour at the current iteration. A roulette wheel concept is also used when selecting an operator, as an extension of the work by [161, 166]. However, in this study, the selection process is dynamic to provide a better reflection of the operator's performance, as the search progress. This study claims that using an active change of the application rates can enhance the GA efficiency when tackling WSRP scenarios. To this end, the specific objectives of this chapter are:

- To present an adaptive multiple crossover GA that aims to enhance the baseline GA efficiency.
- To analyse the impact of crossover operators collaboration on the quality of solutions as well as algorithm speed.

7.2 Adaptive Multiple-Crossover GA (AMCGA)

This section describes the proposed adaptive aspects of the GA as an extension of the proposed approach in Chapter 6. The proposal here is the integration of the adaptive control mechanism for genetic operator rates P_c and P_m while selecting a crossover operator. The reason for choosing adapting parameter rates in this study is because it

has provided the best results in comparison to other adaptive variations implemented for WSRP, as stated in Section 6.3.4.

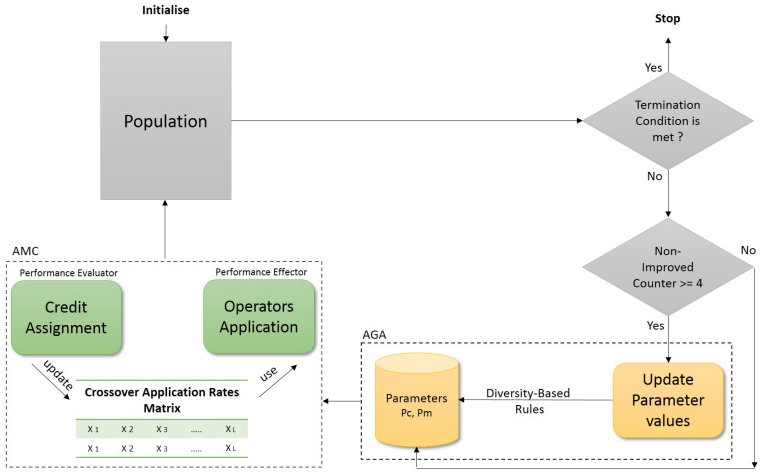


Figure 7.1: Adaptive multiple crossovers GA (AMCGA).

Figure 7.1 shows a flowchart of the proposed adaptive multiple crossovers genetic algorithm (AMCGA). This method is a combination of the adaptive genetic algorithm (AGA) and the adaptive multiple-crossover framework (AMC).

The AMCGA works as follows. First, an initial population P of M individuals (one-day plans) is created based on the indirect chromosome encoding that constructs feasible solutions. The representation is designed to include all assigned visits $|V|$; thus the chromosome length varies according to the problem size. At the start of each generation, $9M/20$ pairs of parent individuals are selected using binary tournaments. Then, a new population is created through crossover and mutation. All crossovers are applied as part of the adaptive multiple crossover framework (AMC), with probability P_c to each pair of parents to generate two offspring. Each offspring go through a flat-cost flip (FCF) mutation, with some probability P_m. The operator performance influences the information feedback process where the adjustment of a parameter value is done. At the end of each generation t, the population is updated using an elitism strategy. The $M/10$ best individuals from the current population are kept, and along with the $9M/20$ offspring individuals generated, the new population of M solutions is formed. After a number of generations, if there are no cost value improvements on the best so far

solutions, the GA is considered to have stagnated. Thus, the updates to parameter values occur in response to the feedback process that seeks to maintain a diverse population to improve the effectiveness of the search. Such feedback from the search is used to generate the next population. Details of the AMC are described next.

7.2.1 Multiple-Crossover Framework (AMC)

This method uses a set of crossovers $\chi = \{x_1, x_2 \dots x_L\}$ of a size L, as one operator that explores the search area simultaneously. For an effective collaboration, a mix of traditional and problem-specific operators are considered. The reason for using a diverse set of operators is to allow dominance of one crossover over the others. Additionally, the problem-specific operators have the advantage of using heuristic-based operators on WSRP instances, where they were established to perform better than traditional crossovers, as stated in Chapter 5.

The steps of applying the AMC are as follows. For each iteration, a σ number of crossover functions are applied, until the next generation is created. Initially, all crossovers are applied uniformly, to obtain sufficient information of their performance. There is at least $\frac{1}{L}$ chance for each operator to be selected. During the learning process (also denoted as a cycle of a size υ), the operators' performances are evaluated and recorded as *scores*. For each crossover, the accumulated scores are stored into the **crossover reward matrix (CRM)**, considered as a reward registry for that crossover in the current cycle. Scores are then transformed into **application rates** P_{x_i}, *where* P_{x_i} is the probability of applying a crossover x_i. These steps are repeated for each cycle, where application rates are dynamically adjusted as the search progress.

Better operators have a higher score value, and therefore a higher probability to be utilised more, and week operators are not left without a chance. For example, given a set χ of six crossovers operators, with application rates as follows: $x_1 = 23$ %, $x_2 = 4$ %, $x_3 = 11$ %, $x_4 = 34$ % , $x_5 = 17$ % and $x_6 = 12$ %. Crossover x_4 has the highest application rate at this cycle, henceforth x_4 has more chance to be implemented. At some point of the search, when x_4 performance changes for the worse, its application rate decreases. Application rates of other crossovers also vary accordingly, causing a shift in the search.

Each operator is implemented independently; however, all operators share the same cycle until it resets. Hence, AMC is considered as a mix between natural systems (recombination operators) and synthetic systems (autonomous agents). The *natural system* aspect is reflected by the collective performances of the crossovers, onto a population [47]. The *synthetic systems* aspect is reflected by sharing a cycle as part of the learning process that directs their application [125]. Details for crossovers performance measurements and CRM construction is explained next.

Performance Measurements

Scores are considered as performance indicators that enforce the rewarding mechanism, *where* $\forall x_i \in \chi$, a score is given at each iteration, where $i = 1, 2 \ldots, L$. To calculate the scores, performance measurements are used to model the behaviour of different operators in the current stage of the search.

Different types of measurements are used in the literature, such as the fitness-based measurement [89, 174], the distance-based measurement [26] and the combined measurement and the operator execution-time measurement [104]. In this study, the execution time can be relative to the problem size for WSRPs. Hence, the time is not absolute in real-world settings, and therefore this measurement was excluded.

Parameter settings are affected by the population diversity, as explained in Chapter 6. For example, if instances are relatively small, a large M is required as a result of the large degree of similarity between individuals. For a large instance, on the other hand, there is less chance to have similar solutions. Thus, a small M is sufficient. Therefore, distance-based measurement is considered here along with the fitness-based measurement. Performance measurements used in this chapter are as follows.

1. **Fitness-Based**, This method is selected to maximise the cumulative improvement, as a historical fitness record of the cost value of an offspring (o) and its parent (p), where $f(o) < f(p)$. When there are improvement on the overall cost value, the score increases by one. This value ensures the convergence by selecting crossovers with better offspring quality, that eventually improves the overall performance of a GA.

2. **Distance-Based**, This method is selected to ensure a level of diversity among generated solutions, by calculating the distribution of the pair-wise differences between an offspring and its parent. If the percentage is 60% or more, the score increases with respect to the *hamming distance* H between two individuals. This value is selected to prevent inbreeding among generations. The value of H is calculated as the number of assignments in which the corresponding genes are different. Figure 7.2 illustrates an example of the dissimilarities percentage calculations (presented in white cells) between a parent and an offspring.

3. **Hybrid Approach**, This method is selected to harness the power of the two above measurements. When there is an improvement in the fitness or there exist differences between parents and offspring, the score increases. This means that with every crossover evaluation, the score value increases gradually. If only one measurement has proved an operator efficiency, the score value increases by one. On the other hand, if both performance measurements have proved an operator efficiency simultaneously, the score value increases by two.

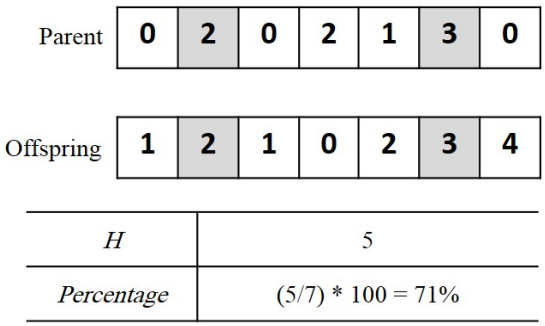


Figure 7.2: Dissimilarity percentage calculation between two individuals.

Each crossover is evaluated in isolation, after each application, with a performance measurement. The purpose of such separate evaluation is to obtain their relative score, under the evaluation criterion. In the event of one crossover is selected repeatedly, the score value increases gradually, based on the feedback from the online learning mechanism in AMC. Details for application rates calculations is described next.

Application Rates

Rather than using a large crossover rewards matrix (CRM), when recording the accumulated scores, a $2 \times L$ matrix is used to allow faster computation. i.e., the score value in $(1, i)$ is added to the value in $(2, i)$. The accumulated value is then transformed to an application rate $P_{x_i} = \frac{Score_i}{P_{Sum}}$, where $P_{Sum} = \sum_{i=1}^{L} P_{x_i}$.

For the next iteration, application rates are updated, until all cycles are completed. As a result, a gradual change in the P_{x_i} value exists, with no rapid increase or decrease in the overall application rates. This process is repeated, for each cycle. Application rates are always modified before the next cycle.

Figure 7.3 shows an example of the CRM construction process, for two crossovers x_1 and x_2, with a cycle size $\upsilon = 3$. For every cycle, two steps are executed. First, scores are retrieved for each crossover. In this case, the first cycle has a $Score_1 = 3$ and a $Score_2 = 4$. Next, the accumulated scores are calculated by adding up the scores at the position $(1, i)$ to the scores at the position $(2, i)$.

Once a score is shifted to position $(2, i)$, a value of 1 is stored at position $(1, i)$, to ensure the application of all crossover functions, regardless of a crossover performance. For example, if a crossover application has resulted in no improvement or even worse solution, it is still applied in the AMC. The active calculation of the application rates maintain a balance between the most effective operators, which provide good results, and week crossovers, which provide poor results.

7.3 Experimental Study and Results

These set of experiments have focused on applying the AMC on the baseline GA, to capture the benefits of this method in addition to enhancing the performance. The main GA components used in this chapter were the indirect chromosome representation and the flat-cost flip mutation (FCF). Additionally, a mix of six crossover operators was used as part of the AMC, including 1PX, 2PX, UX, HX, FCX, and PMFCX. Greedy crossovers were excluded from this study due to their excessive computational time, as explained in previous chapters. Even though they were relatively good for small instances, they were not necessarily the best operators for large problems.

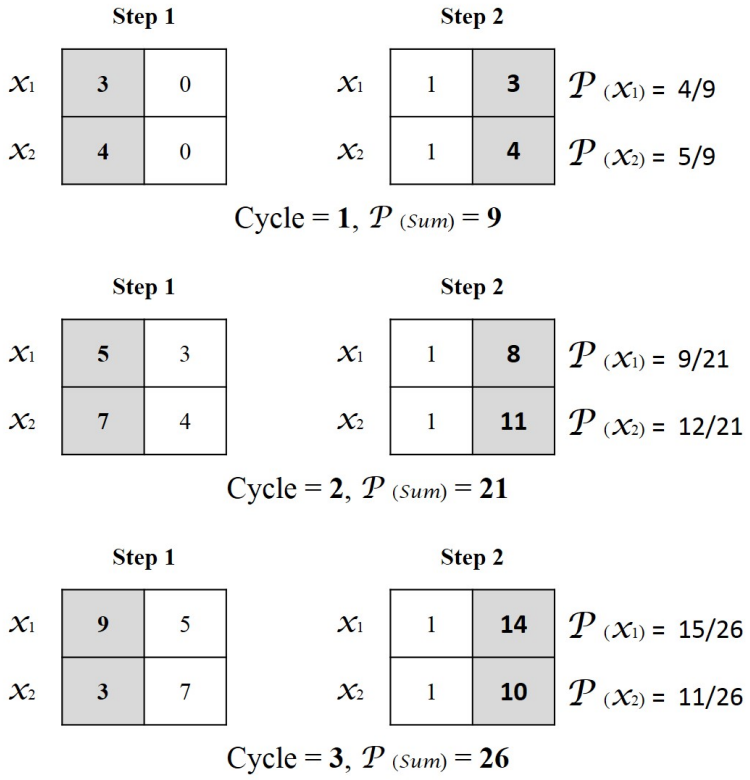


Figure 7.3: An illustration of application rates calculations, in an AMC, shown for 2 crossovers and 3 cycles.

Table 7.1: Parameter settings for AMC and AMCGA.

Parameter	A	B	C	D	E	F
P_m	50%	50%	30%	10%	10%	10%
P_C	50%	50%	50%	100%	100%	100%
M	500	500	500	250	250	250
σ	250	250	250	125	125	125

Parameter settings, used in chapter 5, were successful in providing good results. Each run in these experiments was executed with the same set of settings, shown in Table 7.1. Crossover rate P_c and mutation rate P_m, were used as initial values for the AMCGA. These values were then adjusted when using the adaptive parameter control method (AGA), which was a component of the AMCGA.

In order to obtain enough information of the search space, each WSRP problem set has a different number of crossovers to be applied during the learning process, denoted

as σ. The P_C value has an effect on the parameter value selected for σ. For example, in smaller instances, P_c value was 50%; therefore, fewer crossovers were applied. Hence, a larger σ value was required, to acquire enough knowledge regarding the crossovers' performances. In larger instances, P_C was 100%, this means that more crossovers were applied, and a large σ value was not necessary.

Results were grouped by the performance measurement applied, and denoted as follows. Fitness improvements (AMC_f), dissimilarities between individuals (AMC_d) and hybrid approach (AMC_h). The best performing method was selected to be included in the AMCGA, which combined the AMC with the AGA.

Different cycle settings were used here, where $\upsilon = \{5, 10, 15, 25, 40, 50\}$. The reason for selecting these values was due to preliminary experimentation. Testing revealed that a cycle of a size less than 5, had a slow learning process. Therefore, all operators were applied equally. A cycle of a size larger than 50 had a low accuracy of the learning factor, in which crucial information was lost. A crossover operator might be the best performer at the start of a cycle, yet it performance can change when the search progressed.

Each cycle was executed eight times (runs), i.e., for each instance, there were 8 $\times$ 6 (cycles) $\times$ 3 (methods) = 144 runs, seeded with a random initial population. The methods used to evaluate the results were as explained in Chapter 4.3. See Appendix D for all the results using the above parameter settings.

7.3.1 Performance of AMCs in Time

Each line in Figure 7.4 shows a change in the cost value overtime, for problem sets A to F. All three multiple crossovers methods were applied and plotted. Each line plots the cost values for AMC_f (black), the AMC_d (blue), and AMC_h (red). The lower the line, the better the cost value.

The AMC_f has spent more time on exploring the search space in all of the problem sets. As a result of the time required to calculate the scores according to the fitness improvements. However, AMC_f has converged earlier on problem sets A and F.

In problem set A, AMC_f has found the best results early in the search. Hence, this method converged faster. On the other hand, in problem set F, AMC_f has struggled in

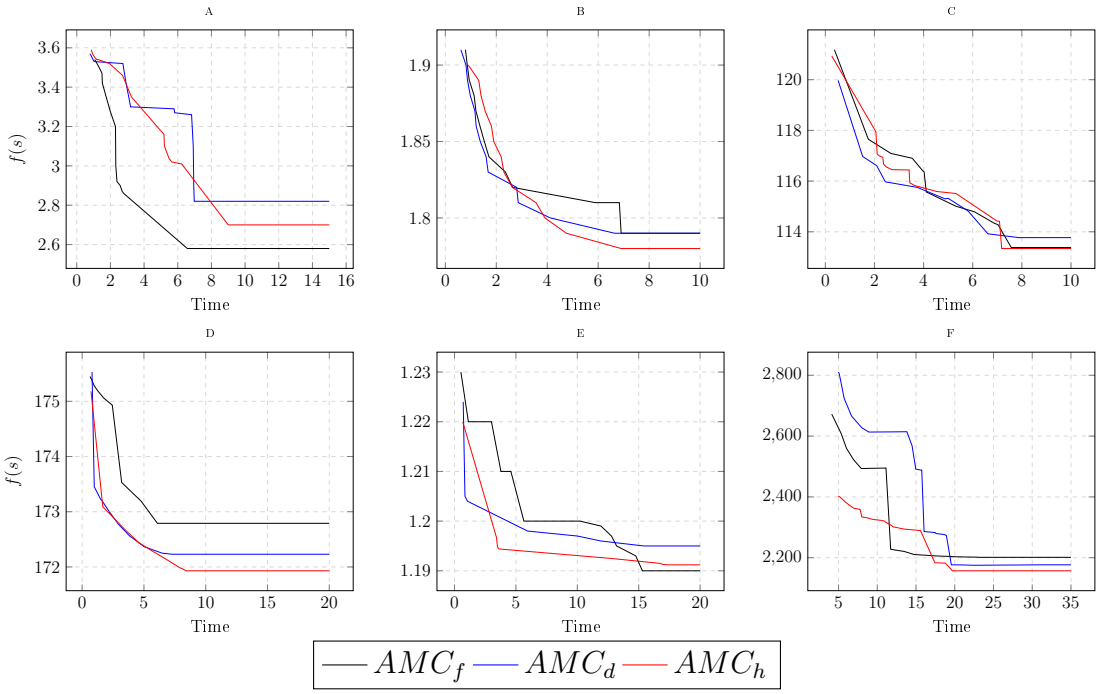


Figure 7.4: Change in cost values $f(s)$ over time (in seconds) shown for methods AMC_f, AMC_d and AMC_h.

obtaining good solutions, within the time limitations, due to the slow fitness calculations.

On the other hand, the AMC_d has obtained the results faster than the AMC_f, with worse results on problem sets A, C, and E. This was the outcome of the dissimilarity calculations, which was more sensitive to change than retrieving information of the fitness improvements. Hence, results were computed faster, which was produced in prematurely. The performance of the AMC_d is discussed later in Section 7.3.3.

Contradictory to AMC_d, AMC_h has a slower convergence, with better results on problem sets B, D, and F. Still, it has converged faster than AMC_f. This was the outcome of the combination of two performance measurements, which was calculated within the time limitations. The reason for this was the faster calculation of the accumulated scores, that rewarded a successful crossover by a value of 2 instead of 1, as the other performance measurements used when the scores were calculated. This method delayed the GA stagnation and converged to better results in the longer run.

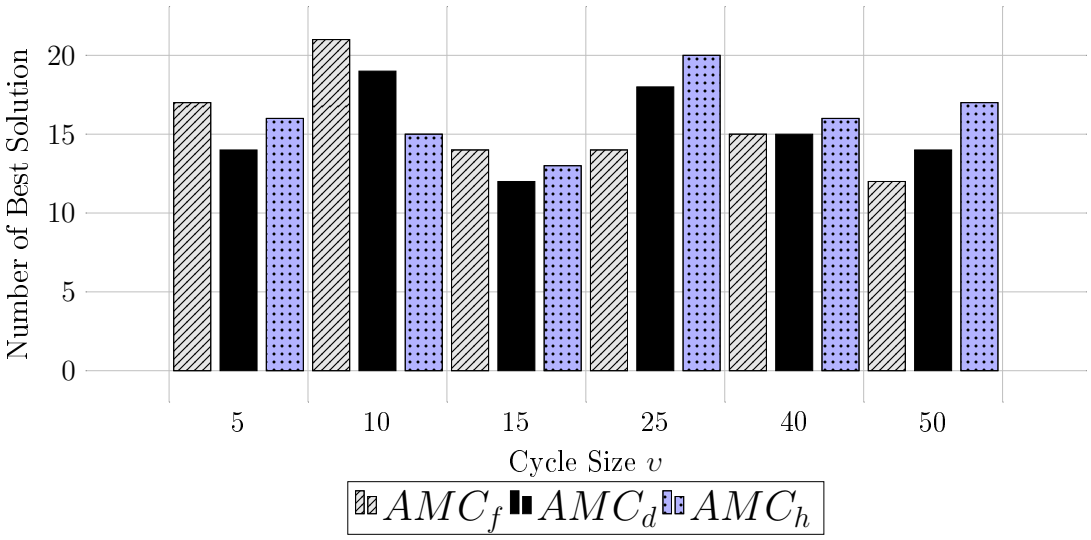


Figure 7.5: Total number of best solutions produced by different rewarding mechanisms (AMC_f, AMC_d and AMC_h).

7.3.2 Performance of Different Cycle Sizes

These experiments aims to identify the best cycle size (υ) for each WSRP problem set, with respect to the rewarding method applied.

Figure 7.5 shows the total number of best solutions for all problem instances. The x-axis represents a rewarding method applied with a cycle of a size υ. Methods shown in the plot were: AMC_f (grey bars with striped lines), AMC_d (black solid bars) and AMC_h (blue bars with dots). The higher the bar, the better, i.e., the more "best" values.

Clearly, there was no apparent difference between rewarding methods applied. All methods have provided a relatively similar number of best solutions with a slight increase on one method over the others. Still, AMC_f has the highest number of best solutions when υ was 10. Followed by AMC_h, when υ was 25.

The result was related to the scores given to an operator throughout the search, in which the accumulated score increases or decreases gradually. Causing a shift in the score value throughout the cycle, especially when a large υ value was used.

Interestingly, the highest number of best solutions was obtained while using AMC_f, when υ was 10. With one performance measurement, i.e., AMC_f and AMC_d, small changes in the score values occurred. This provided enough time for updating the application rates, and therefore these crossovers were used in the current population. Hence,

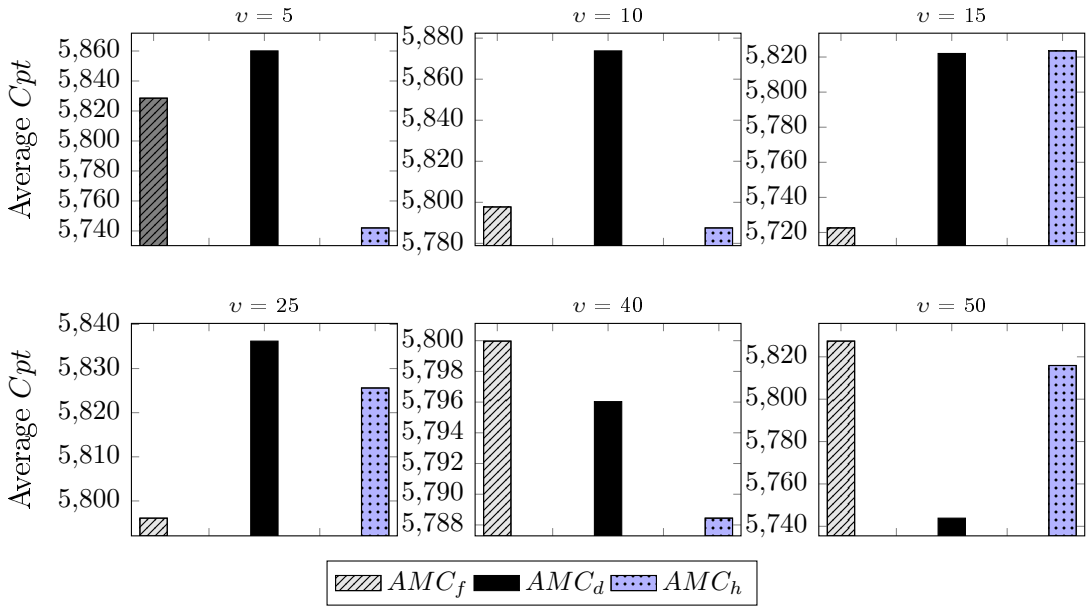


Figure 7.6: Average computation-time (in seconds) produced by AMC variations with different cycle sizes υ.

there was no need for a large cycle with long historical records, especially when one performance measurement was applied.

The AMC_h required a large υ value, as a result of the discrete change in the accumulated scores. Nevertheless, if υ was small, the time was not enough to reward all the operators. Thus, one operator dominated the algorithm, resulting in an uneven distribution of the application rates.

The larger the υ value, the vast increase in the score. The opposite occurred in small cycles, in which were ignored earlier in the search. The υ value can affect a crossover dominance in the AMC method over another.

Figure 7.6 illustrates the overall average computational times in seconds on problem sets A to F while using different υ values. Each sub-figure corresponded to a cycle size υ, and presented the average computational time used by AMC_f (grey bars with stripped lines), AMC_d (solid black bars) and AMC_h (blue bars with dots). The lower the bar, the better, i.e., the less computational time.

From the plots, it was apparent that different methods provided solutions in approximately similar times. The largest computational times for AMC_f, among all methods, was recorded when $\upsilon = 40$ and 50. It was more expensive to make large adjustments

in the scores, especially when smaller υ was used. On the contrary, more time existed when using a large υ values. This outcome provided a shift in the scores.

The fitness-based calculations were faster than the diversity-based calculations. When υ was equal 5 to 25, more time was spent on calculating the scores with AMC_d. This was the outcome of the diversity-based calculations, computed gene by gene, and executed V times for each individual. An enormous amount of time was required to calculate the scores accurately. However, when υ was equal to 40 or 50, the sensitivity of the distance-based calculation has computed the solutions in less time. AMC_h had similar computation times to AMC_d, with $\upsilon = 15$.

The next set of experiments was designed to use one cycle size per method, selected according to the highest total number of best solutions. Thus, $\upsilon = 10$ was chosen for AMC_f as well as for AMC_d, and $\upsilon = 25$ was selected for AMC_h.

7.3.3 Performance Comparison of Different AMCs

Table 7.2: Statistical results of the AMCs methods generated by a non-parametric Friedman's test combined with performances metrics

Method	Mean	Std. Deviation	Min	Max	Mean Rank	Dev %	# Best	Score
$\boldsymbol{AMC_f}$	287.07	699.78	1.18	3495.55	1.99	0.13%	0.50	0.65
$\boldsymbol{AMC_d}$	**286.74**	**699.12**	1.18	3496.73	**1.82**	**0.05%**	**0.64**	**0.73**
$\boldsymbol{AMC_h}$	287.23	700.10	1.18	3496.36	2.19	0.16%	0.33	0.46

Bold text refers to the best result.

The second set of experiments was designed to compare the proposed AMCs variations. To identify if there was a significant difference between the proposed AMCs variations or not, a two-way Friedman analysis was used as a non-parametric statistical test to establish statistical significance in EAs. In this thesis, an IBM SPSS 22 two-way analysis was used to compare the variances of seven related-samples, with a significant level of $\alpha = 0.05$ and 95% as a confidence interval.

Table 7.2 provides the results generated by the Friedman analysis on the 42 problem instances including the mean value, the standard deviation, the minimum cost-value, the maximum cost-value and the mean rank. Results presented in the mean rank column show the methods order based on the statistical analysis, where a low rank indicates the

best method while a high rank indicates the worse method ranked overall. All problem instances were used to set the sample size of one method as large as possible. This increases the probability of accepting or rejecting the null hypothesis. The best results are highlighted in **bold**.

Results in Table 7.2, had a significance level = 0.061, degrees of freedom = 2 and χ^2 = 5.605, indicating that there was no evidence of a statistically significant difference between all methods. Thus, three performance measurements matrices were used: **Dev%**, **Best** and **Score**, calculated as explained in Chapter 4.

Further statistical tests revealed that AMC_d had the lowest mean value, standard deviation value, mean rank value and Dev% value. This method also had the highest fraction of the number of best solutions with better score values, as stated in the table above. Using the distance-based measurement was more sensitive to the change of the diversity. Hence, this measurement has provided scores that have a higher level of accuracy in comparison to the other AMC methods.

Different methods applied to different instances can obtain different results, and henceforth each problem set benefited from each method accordingly. Thus, it can be argued that all variations were suitable for WSRP, while AMC_d was slightly better than the other proposed AMCs.

Effect of Adaptive Application Rates on AMCs

This section investigates the effectiveness of one crossover over the other, by identifying the most used operator in each method. All three methods were applied, with a set of crossovers, i.e., 1PX, 2PX, UX, HX, FCX, and PMFCX.

The x-axis in Figure 7.7 illustrates the average application rate P_{x_i} for all problem sets. Methods applied were AMC_f (grey bars with striped lines), AMC_d (solid black bars), and AMC_h (blue bars with dots).

Among all crossovers applied for AMC_f, the values were relatively similar with a slight increase the average application rates when FCX was used. This result further confirms the observations derived from Chapter 5, which identified the FCX as one of the best crossovers for WSRP problem sets. Due to FCX superior performance, it has

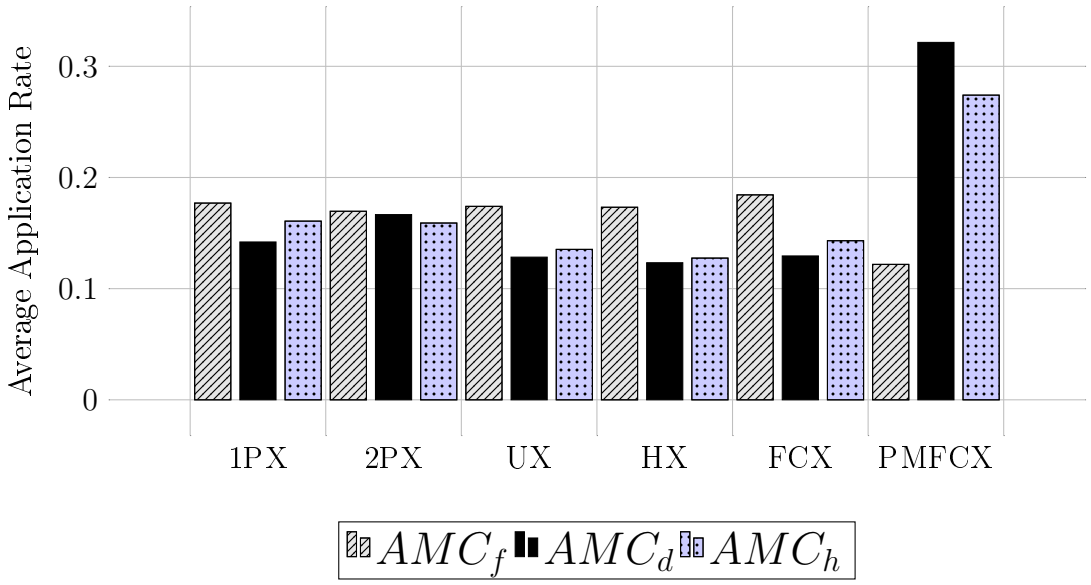


Figure 7.7: Average application rates values used under different rewarding mechanisms (AMC_f, AMC_d, and AMC_h).

been used more with AMC_f. Faster calculations were required when using AMC_d and AMC_h. Thus, the PMFCX crossover was utilised more, due to its fast calculations as a result of mixing heuristic approach with traditional PMX crossover method.

Figure 7.8 illustrates a group of box plots comparisons to show the overall patterns of response for application rates ranges, for each crossover operator, i.e., the bigger the box plot, the more diverse was the application rates values to the correspondent crossover.

Each sub-figure corresponds to a method applied, i.e. AMC_f, AMC_d and AMC_h. Each box-plot illustrates the average application rate range for each crossover and are shown in 1PX (blue), 2PX (red), UX (black), HX (green), FCX (grey) a PMFCX (brown). A change in the application rate value indicates that the crossovers distribution has corresponded to the current environment state.

When using traditional crossovers, poor quality solutions were provided, which were comparatively similar to their parents. Still, using them alongside the problem-specific method can ensure dissimilarities among the generated offspring.

The box plot for PMFCX was lower than all other crossovers when AMC_f was used. This manes that application rates were diverse when using the PMFCX and therefore updated more frequently. This was due to the feedback information obtained from the

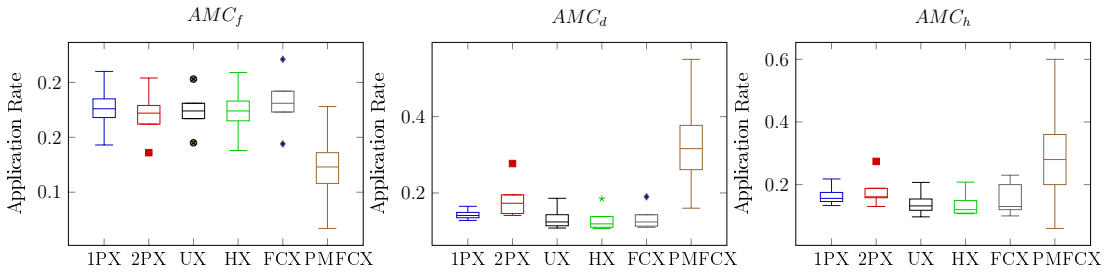


Figure 7.8: Box plots comparisons of the average application rates range shown for crossovers 1PX (blue), 2PX (red), UX (black), HX (green), FCX (grey) and PMFCX (brown).

learning process. The opposite occurs for AMC_d and AMC_h. This result supports the previous finding discussed in this section.

Effect of Using Online Learning on AMC

The third set of experiments compares the performance of each AMC variation, i.e., AMC_f, AMC_d and AMC_h with IGA presented in Chapter 6. The comparison also included a variation of AMC that excluded the reinforcement learning process, noted as AMC_r, with a uniform choice of operators. Table 7.3 shows the solution quality, noted as $f(S)$, and the time in seconds, noted as Cpt, in which the best solution was found. The best values were highlighted in **bold**.

As it can be seen from the table, all AMC variations outperformed the GA and AMC_r, in terms of solution quality and computational time, regardless of the performance measurements applied. Closer inspection of the results shows the percentage of best cost values overall were as follows. The GA 14.29%, AMC_r 30.95%, AMC_f 40.48%, AMC_d 47.48% and AMC_h 21.43%.

For each problem set, all AMCs methods applied have obtained good quality solutions. This was because of using a variety of operators, in which they have different performances. The AMCs were able to make small movements in various directions in the solution space, and thus a wider exploration was ensured [125]. By contrast, in IGA, the search was completed by one crossover, with less ability to extend the search in those regions, which were most promising.

The AMC_r has utilised different crossovers, at an arbitrary level, without any prior knowledge of the current search space. As a result, there were no advancements on the

Table 7.3: Cost-values $f(S)$ and computational-time Cpt (in seconds) for WSRP instances, produced by IGA, AMC_r, AMC_f, AMC_d and AMC_h.

		IGA		AMC_r		AMC_f		AMC_d		AMC_h	
		$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt	$f(S)$	Cpt
A	1	3.49	34.08	3.49	3.90	3.49	2.27	3.49	1.85	**3.49**	**1.65**
	2	2.96	396.77	2.49	5.07	2.54	2.34	2.49	2.62	**2.49**	**1.42**
	3	3.65	436.92	3.01	153.54	**3.02**	**14.83**	3.02	58.57	3.02	115.48
	4	1.42	229.01	1.42	41.76	**1.42**	**41.54**	1.42	67.69	1.42	66.38
	5	2.42	0.98	2.42	1.07	2.42	0.81	2.42	1.26	**2.42**	**0.42**
	6	3.61	159.83	3.55	4.64	**3.55**	**3.21**	3.55	8.30	3.55	8.93
	7	3.71	0.83	3.71	1.02	3.71	0.81	3.71	1.42	**3.71**	**0.32**
B	1	1.76	422.62	1.72	146.60	**1.72**	**141.90**	1.72	276.01	1.72	262.21
	2	1.75	78.57	1.75	0.64	1.75	0.71	1.75	1.22	**1.75**	**0.34**
	3	1.90	629.30	1.79	329.10	1.78	555.93	1.78	570.47	**1.78**	**435.03**
	4	2.08	29.51	2.08	4.77	**2.08**	**4.06**	2.08	21.25	2.08	7.28
	5	1.96	434.63	1.91	417.83	**1.90**	**490.40**	1.91	483.60	1.92	558.09
	6	1.71	343.74	1.64	105.39	1.65	80.54	**1.64**	**72.75**	1.65	137.81
	7	1.87	502.96	1.80	248.04	**1.79**	**57.24**	1.80	244.68	1.79	144.08
C	1	115.13	846.87	115.01	597.34	114.81	513.89	**114.75**	**566.20**	114.88	641.68
	2	3.15	0.46	3.15	0.98	3.15	0.67	3.15	0.86	**3.15**	**0.20**
	3	104.83	657.11	105.02	659.60	**104.65**	**675.85**	104.89	434.64	104.82	535.76
	4	11.15	2.41	11.15	2.65	11.15	1.84	**11.15**	**1.79**	11.15	2.25
	5	12.34	3.38	12.34	16.09	12.34	2.94	12.34	68.90	**12.34**	**2.58**
	6	180.98	591.22	180.98	367.50	**180.80**	**301.95**	180.80	431.25	180.86	377.21
	7	**4.30**	**0.00**	**4.30**	**0.00**	**4.30**	**0.00**	**4.30**	**0.00**	**4.30**	**0.00**
D	1	173.15	2989.34	171.39	3140.40	171.08	3408.46	**170.71**	**3334.63**	171.08	3255.77
	2	169.56	3502.67	167.50	3448.17	168.11	3046.96	**167.31**	**3405.58**	168.31	3450.06
	3	184.49	3502.43	180.93	3211.17	180.86	3361.38	**180.86**	**3102.37**	181.55	3390.17
	4	171.83	3484.81	168.68	3338.98	168.42	3323.28	168.37	3476.31	**167.86**	**3412.84**
	5	162.32	3589.68	161.93	3382.11	161.99	3266.39	**161.86**	**3276.84**	161.92	3113.17
	6	178.80	3446.80	178.16	3268.95	178.35	3209.79	**178.16**	**3353.26**	178.28	3278.63
	7	178.85	3423.81	178.65	3263.43	178.71	3367.56	178.65	3347.14	**178.59**	**3271.47**
E	1	3.75	3314.33	1.18	3420.64	1.18	3366.72	1.18	3404.99	**1.18**	**3206.71**
	2	3.86	3570.60	1.21	3394.69	**1.21**	**3311.93**	1.21	3399.96	1.21	3431.21
	3	5.18	3371.76	1.22	3345.45	**1.21**	**3428.17**	1.22	3460.71	1.22	3395.24
	4	3.27	3584.67	1.30	3419.12	1.29	3512.18	**1.29**	**3493.56**	1.30	3432.41
	5	5.15	3585.41	**2.24**	**3287.96**	2.24	3469.36	2.25	3424.19	2.25	3450.09
	6	3.56	3559.72	1.30	3406.16	**1.30**	**3233.54**	1.30	3356.92	1.30	3377.66
	7	3.92	3574.02	**1.73**	**3339.36**	1.73	3486.11	1.73	3452.58	1.73	3375.31
F	1	2257.51	28710.28	2034.37	27834.65	2035.17	27760.40	**2022.40**	**28294.63**	2039.30	27768.47
	2	2306.05	28672.43	2084.22	27286.12	2084.84	27671.16	**2083.97**	**27978.79**	2084.47	27944.11
	3	850.24	28700.42	**608.07**	**27635.80**	608.32	27767.11	608.13	27867.36	608.63	27280.61
	4	1546.96	28733.22	1329.08	26819.62	1328.77	27863.52	1329.58	28029.66	**1328.33**	**27308.04**
	5	453.99	28415.05	**196.57**	**26846.99**	197.57	26783.72	197.20	28222.74	198.51	28380.85
	6	855.31	28707.11	625.58	27613.17	624.96	27835.21	**624.96**	**27996.64**	625.77	27519.31
	7	3980.44	28660.10	3495.86	27970.66	**3495.55**	**28138.01**	3496.73	27705.17	3496.36	28333.90

Bold text refers to the best result.

overall performances of these crossovers. Therefore, other AMC variations were considered better for some of WSRP problem instances. Nevertheless, AMC_r has obtained the best results on E_5, E_7, F_3 and F_5. This was because time was spent on finding a solution rather than using performance measures, which slows the process of generating solutions.

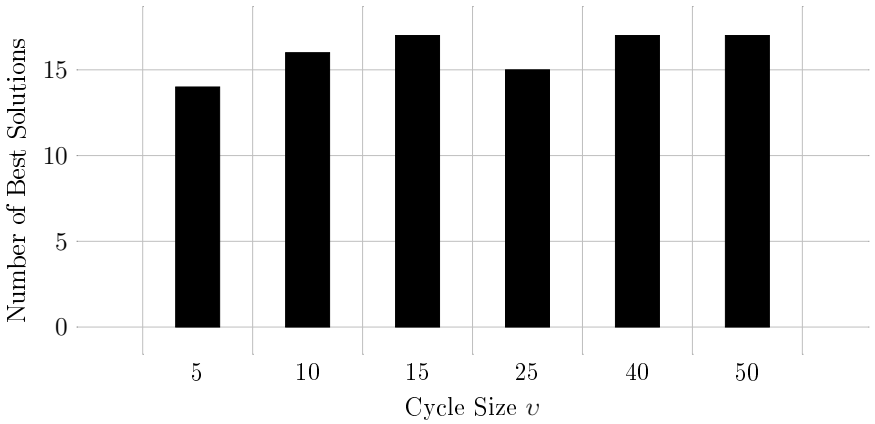


Figure 7.9: Total number of best solutions while using AMCGA, produced by different cycle sizes.

In regards to computational time, AMCs proved to improve the algorithm efficiency by providing better results in less computational time. For AMC_r, AMC_f, AMC_d and AMC_h, the average Cpt in seconds were 6069.04, 5756.7, 5797.7, 5873.8 and 5825.6 respectively. In IGA, one crossover was applied at each iteration. Hence, the algorithm required more time to improve the solution. On the other hand, the AMCs enforces the performances of different crossovers onto one population.Therefore, the proposed AMCs obtained solutions in fewer iterations than the IGA. In AMC_r, the average Cpt was the lowest among all compared methods. This was due to the exclusion of the feedback process. On the other hand, the average Cpt for AMC_f was less than AMC_d and AMC_h, as an outcome of computing the dissimilarities between the parent and the offspring.

7.3.4 Using AMC as a Component of AMCGA

A major advantage of AMCs was that they had a considerable influence on improving the efficiency of the GA, especially AMC_d. Thus, this method was selected to be integrated with AMCGA. The following experiments aim to evaluate the validity of the AMCGA by providing further insights into its performance.

Effect of using different cycle sizes in AMCGA

In this section, the aim was to investigate the effect of the using different cycle sizes in AMCGA on the quality of solutions in addition to the computational time. Figure 7.9 shows the total number of the best solutions generated by AMCGA. Bars on the

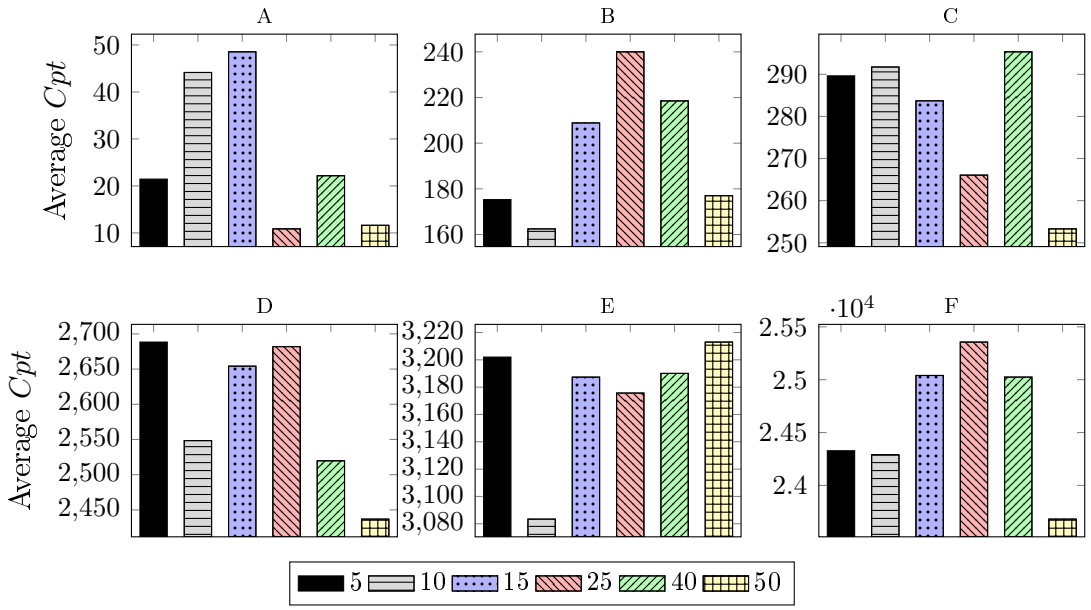


Figure 7.10: Average computational-time (in seconds) produced by AMCGA.

x-axis represents an AMCGA method applied with a cycle size υ. The higher the bar, the better, i.e., the more "best" values.

All methods were relatively similar in providing a high number of best solutions, in more than one cycle size. Thus, it can be claimed that using AGA made the performance of the AMC more stable. The number of best solutions was the highest when υ was 15, 40 and 50 with a value of 17 best solutions. Followed by $\upsilon = 10$, with 16 solutions, and $\upsilon = 25$, with 15 solutions. The values for the number of best solutions when $\upsilon = 5$ was the lowest. The time was insufficient to score all operators. Thus, a large cycle size, i.e., 50, provided better learning outcomes within the time by retrieving the needed information.

Each sub figure in Figure 7.10 corresponds to a problem set, to illustrate the overall average of the computational time (in seconds) used by a cycle size. Black solid bars when $\upsilon = 5$, grey bars with stripped lines when $\upsilon = 10$, blue bars with dots when $\upsilon = 15$, red bars with right inclined lines when $\upsilon = 25$, green bars with left inclined lines when $\upsilon = 40$ and yellow bars with a grid when $\upsilon = 50$. The lower the bar, the better, i.e., the less computational time.

On average, the lowest computational times for problem set A was when $\upsilon = 25$ However, all cycle sizes have obtained the same result in less than 50 seconds, which

was relatively acceptable. Followed by $\upsilon = 50$, that also obtained the lowest average computational times for problem sets C, D, and F. The lowest average computational times for problem sets B and E were when $\upsilon = 10$ with a difference between 100 to 200 seconds to the other sizes. Regarding Figure 7.9, this cycle size also obtained the second highest number of best solutions.

The larger the cycle size, the more scores were calculated for an operator in addition to updating the operator rates. As a result, the population evolving into better, fitter solutions. The interchange between the two adaptive aspects was vastly exploited in AMCGA. Results indicate that integrating the AMCGA with AMC_d of a cycle of size 50 provided the best results in less time, especially for difficult problem sets. The results for a $\upsilon = 10$ and $\upsilon = 50$ was investigated further in the next section.

Results of AMCGA vs. AMC_d and AGA

Table 7.4: Cost-values $f(S)$ and computational-time Cpt (in seconds) for WSRP instances, produced by AGA, AMC_d, $AMCGA_{10}$ and $AMCGA_{50}$ method.

		AGA		AMC_d		$AMCGA_{10}$		$AMCGA_{50}$				AGA		AMC_d		$AMCGA_{10}$		$AMCGA_{50}$	
		f(S)	*Cpt*	*f (S)*	*Cpt*	*f (S)*	*Cpt*	*f (S)*	*Cpt*			*f(S)*	*Cpt*	*f (S)*	*Cpt*	*f (S)*	*Cpt*	*f (S)*	*Cpt*
	1	3.49	14.45	3.49	1.85	3.49	2.81	**3.49**	**1.78**		**1**	170.77	2978.51	170.71	3334.63	**170.23**	**2533.55**	170.29	2545.39
	2	2.49	100.69	2.49	2.62	2.49	4.49	**2.49**	**1.27**		**2**	166.87	3228.52	167.31	3405.58	166.58	2109.05	**166.26**	**2512.36**
	3	3.08	11.50	3.02	58.57	3.02	221.92	**3.02**	**27.63**		**3**	180.60	3258.30	180.86	3102.37	179.81	2490.46	**179.75**	**2370.22**
A	**4**	1.42	247.07	1.42	67.69	1.42	51.27	**1.42**	**42.71**	**D**	**4**	167.93	3465.12	168.37	3476.31	166.64	2382.08	**166.39**	**2511.47**
	5	2.42	1.11	2.42	1.26	2.42	0.87	**2.42**	**0.38**		**5**	161.73	3367.04	161.86	3276.84	161.50	2503.26	**161.37**	**2532.9**
	6	3.56	130.22	3.55	8.30	3.55	26.59	**3.55**	**7.05**		**6**	178.28	3200.96	178.16	3353.26	178.11	2765.96	**177.99**	**2368.7**
	7	3.71	0.72	3.71	1.42	3.71	1.03	**3.71**	**0.51**		**7**	**178.40**	**2489.11**	178.65	3347.14	178.41	3054.21	178.41	2217.13
	1	1.75	84.00	1.72	276.01	**1.72**	**160.06**	1.73	146.45		**1**	1.18	3322.13	1.18	3404.99	1.17	3390.71	**1.17**	**3285.83**
	2	1.75	0.56	1.75	1.22	1.75	1.03	**1.75**	**0.29**		**2**	1.21	3368.60	1.21	3399.96	**1.20**	**3303.48**	1.20	3311.01
	3	1.82	280.35	1.78	570.47	**1.77**	**327.03**	1.79	468.52		**3**	1.21	3246.99	1.22	3460.71	1.21	3181.93	**1.21**	**3137.97**
B	**4**	2.08	46.19	2.08	21.25	2.08	5.12	**2.08**	**3.17**	**E**	**4**	1.29	3492.21	1.29	3493.56	**1.29**	**2932.00**	1.29	3123.67
	5	1.93	98.85	1.91	483.60	1.92	459.26	**1.91**	**396.11**		**5**	2.24	3442.02	2.25	3424.19	**2.24**	**2921.37**	2.24	3378.24
	6	1.66	190.74	**1.64**	**72.75**	1.65	97.23	1.64	146.05		**6**	1.30	3283.59	1.30	3356.92	**1.30**	**2539.72**	1.30	2975.27
	7	1.81	154.78	1.80	244.68	1.80	87.54	**1.80**	**78.43**		**7**	1.73	3473.85	1.73	3452.58	1.72	3314.44	**1.72**	**3278.99**
	1	115.07	703.71	**114.75**	**566.20**	114.87	644.23	114.94	640.35		**1**	2122.60	27647.63	2022.40	28294.63	2036.82	23117.21	**2019.72**	**22709.3**
	2	3.15	0.46	3.15	0.86	3.15	0.71	**3.15**	**0.30**		**2**	2082.79	26318.27	2083.97	27978.79	2081.99	23765.59	**2081.74**	**22516.9**
	3	105.08	677.45	104.89	434.64	104.75	804.69	**104.82**	**681.10**		**3**	607.26	27722.40	608.13	27867.36	605.14	22220.34	**604.71**	**22658.5**
C	**4**	11.15	2.40	**11.15**	**1.79**	11.15	3.14	11.15	2.64	**F**	**4**	1328.51	27229.31	1329.58	28029.66	1326.28	24782.11	**1325.90**	**23459**
	5	12.60	33.70	12.34	68.90	12.34	2.40	**12.34**	**2.35**		**5**	196.08	26934.63	197.20	28222.74	193.59	25631.57	**192.78**	**25895.8**
	6	180.99	589.30	**180.80**	**431.25**	180.92	586.95	180.99	446.39		**6**	624.77	25367.39	624.96	27996.64	622.41	26415.45	**622.34**	**23680**
	7	**4.30**	**0.00**	**4.30**	**0.00**	**4.30**	**0.00**	**4.30**	**0.00**		**7**	3495.99	27225.77	3496.73	27705.17	**3492.87**	**24083.07**	3493.18	24846.2

Bold text refers to the best result.

These experiments aim to compare the performance of AMCGA against diversity-based measurement AMC variations AMC_d and the diversity-based adaptive operators' rate control GA (noted as AGA). Results in Table 7.4 were shown for AGA, AMC_d, AMCGA when $\upsilon = 10$ (noted as $AMCGA_{10}$) and AMCGA when $\upsilon = 50$ (noted as $AMCGA_{50}$).

From the table, it can be seen that AMCGA has outperformed AGA and AMC_d,

especially $AMCGA_{50}$ that provided the best results on 59.52% of all solutions. Followed by $AMCGA_{10}$ that provided the best results on 42.86% of all solutions. These results indicate the power of combining two adaptive elements that work together. On average, the computational times in seconds were 5653.11, 5873.79, 5069.67 and 4962.10 for AGA, AMC_d, $AMCGA_{10}$ and $AMCGA_{50}$ respectively. Another advantage of AMCGAs was the rapid decrease in computational time, which was previously reduced by each method separately, in comparison to the IGA. Thus, using the combined method was more cost-effective than using one method individually for WSRP.

7.3.5 Results of AMCGA vs.WSRPs Solutions Methods

This section provides a comparison of the best-performing methods proposed in this thesis,i.e., $AMCGA_{10}$ and $AMCGA_{50}$, against three existing WSRP applications: MIP solver [78], MIP with decomposition [79] and VNS algorithm [127].

Table 7.5 shows the solution quality, noted as $f(S)$, and the computational time in seconds, noted as *Cpt*, in which the best solution was found. The best values are highlighted in **bold**.

For smaller problem sets, the proposed methods were quite competitive, matching the best-known results for many of those instances. The VNS seems to provide better overall results with 42.86% of all best solutions. However, the $AMCGA_{10}$ and $AMCGA_{50}$ outperformed the MIP with decomposition method with 21.43% and 42.86% of all best solutions respectively.

The average computational times were calculated for methods that provided solutions for all instances. i.e., MIP with decomposition $AMCGA_10$ and $AMCGA_50$. The recorded times were 4964.26, 5069.67 and 4962.10 respectively.

The $AMCGA_{10}$ has the largest computational time among all methods. On the contrary, the $AMCGA_{50}$ has the lowest computation time among all methods. These findings provide implications that the use of a large cycle size provided better GA performance, especially cost-wise. A noticeable effect of using adaptive aspects was the reduction of the computation time, which was also enhanced when using AMC. On the other hand, the MIP with decomposition method has obtained a low average computa-

Table 7.5: Cost-values $f(S)$ and computational-time Cpt (in seconds) for WSRP instances, produced by all WSRP solution methods on 42 instances.

		Optimal		MIP DECOMP.		VNS		$AMCGA_{10}$		$AMCGA_{50}$				Optimal		MIP DECOMP.		VNS		$AMCGA_{10}$		$AMCGA_{50}$	
		f (s)	*Cpt*	*f (s)*	*Cpt*	*f (s)*	*Cpt*	*f(S)*	*Cpt*	***f*(S)**	**Cptn**			***f (s)***	***Cpt***	***f (s)***	***Cpt***	***f (s)***	***Cpt***	***f(S)***	***Cpt***	***f*(S)**	**Cptn**
	01	3.49	7.00	5.65	3.71	3.49	2.99	3.49	2.81	**3.49**	**1.78**		**01**	n/a	n/a	496.4	1060.0	**170.3**	**3036.2**	170.23	2533.55	170.29	2545.39
	02	2.49	8.00	4.53	3.58	**2.49**	**0.33**	2.49	4.49	2.49	1.27		**02**	n/a	n/a	372.9	1192.1	**163.9**	**2840.8**	166.58	2109.05	166.26	2512.36
	03	**3.00**	**14.00**	10.65	3.70	3.00	29.30	3.02	221.92	3.02	27.63		**03**	n/a	n/a	3213.3	1209.0	**178.2**	**3172.8**	179.81	2490.46	179.75	2370.22
A	**04**	**1.42**	**5.00**	3.09	2.88	1.42	6.87	1.42	51.27	1.42	42.71	**D**	**04**	n/a	n/a	418.9	3005.1	167.1	3313.8	166.64	2382.08	**166.39**	**2511.47**
	05	2.42	1.00	3.54	1.77	**2.42**	**0.11**	2.42	0.87	2.42	0.38		**05**	n/a	n/a	243.9	1306.7	**161.1**	**2818.5**	161.50	2503.26	161.37	2532.90
	06	3.55	5.00	3.74	2.42	3.55	29.47	3.55	26.59	**3.55**	**7.05**		**06**	n/a	n/a	1411.3	1222.0	**177.4**	**2996.0**	178.11	2765.96	177.99	2368.70
	07	3.71	1.00	4.81	1.64	**3.71**	**0.07**	3.71	1.03	3.71	0.51		**07**	n/a	n/a	753.3	1361.9	177.9	2930.4	178.41	3054.21	**178.41**	**2217.13**
	01	**1.70**	**21.00**	1.79	8.07	1.70	156.45	1.72	160.06	1.73	146.45		**01**	n/a	n/a	33.0	8408.0	n/a	n/a	1.17	3390.71	**1.17**	**3285.834**
	02	1.75	2.00	1.89	4.29	**1.75**	**0.00**	1.75	1.03	1.75	0.29		**02**	n/a	n/a	26.0	12448.4	n/a	n/a	**1.20**	**3303.48**	1.20	3311.008
	03	1.72	6003.00	2.06	32.86	**1.72**	**473.21**	1.77	327.03	1.79	468.52		**03**	n/a	n/a	29.0	20746.6	n/a	n/a	1.21	3181.93	**1.21**	**3137.972**
B	**04**	2.07	25.00	2.21	15.25	**2.07**	**9.03**	2.08	5.12	2.08	3.17	**E**	**04**	n/a	n/a	28.5	15190.5	n/a	n/a	**1.29**	**2932.00**	1.29	3123.67
	05	**1.82**	**585.00**	4.74	25.35	1.83	135.33	1.92	459.26	1.91	396.11		**05**	n/a	n/a	270.1	32619.2	n/a	n/a	**2.24**	**2921.37**	2.24	3378.24
	06	**1.62**	**184.00**	2.52	24.11	1.62	194.29	1.65	97.23	1.64	146.05		**06**	n/a	n/a	24.6	24212.1	n/a	n/a	**1.30**	**2539.72**	1.30	2975.266
	07	**1.79**	**300.00**	4.06	23.64	1.79	304.96	1.80	87.54	1.80	78.43		**07**	n/a	n/a	427.8	51057.3	n/a	n/a	1.72	3314.44	**1.72**	**3278.994**
	01	n/a	n/a	904.98	211.64	**114.21**	**301.32**	114.87	644.23	114.94	640.35		**01**	n/a	n/a	64305.1	3446.4	n/a	n/a	2036.82	23117.21	**2019.72**	**22709.26**
	02	3.15	6.00	3.61	0.57	**3.15**	**0.00**	3.15	0.71	3.15	0.30		**02**	n/a	n/a	73291.2	1111.0	n/a	n/a	2081.99	23765.59	**2081.74**	**22516.85**
	03	n/a	n/a	1186.27	26.33	**103.52**	**550.29**	104.75	804.69	104.82	681.10		**03**	n/a	n/a	115235.2	4555.1	n/a	n/a	605.14	22220.34	**604.71**	**22658.5**
C	**04**	11.15	90.00	81.25	3.09	**11.15**	**0.91**	11.15	3.14	11.15	2.64	**F**	**04**	n/a	n/a	102994.2	4219.2	n/a	n/a	1326.28	24782.11	**1325.90**	**23458.95**
	05	12.34	55.00	68.94	1.05	**12.34**	**0.34**	12.34	2.40	12.34	2.35		**05**	n/a	n/a	101438.2	6156.5	n/a	n/a	193.59	25631.57	**192.78**	**25895.77**
	06	n/a	n/a	3102.26	47.05	**140.44**	**323.52**	180.92	586.95	180.99	446.39		**06**	n/a	n/a	76007.1	9695.6	n/a	n/a	622.41	26415.45	**622.34**	**23680.04**
	07	4.30	1.00	5.29	0.24	4.30	0.02	**4.30**	**0.00**	**4.30**	**0.00**		**07**	n/a	n/a	176540.6	3832.8	n/a	n/a	**3492.87**	**24083.07**	3493.18	24846.23

Bold text refers to the best result.

tion time. However, the results obtained were poor in comparison to the other methods.

So far, the proposed methods were the only algorithms that provided results for all instances. Hence, adaptive GAs were able to solve real-world and highly constrained optimisation problems. Interestingly, the best cost values were obtained by diversity based methods. This further proves the significance of maintaining a diverse population in enhancing the GA performance when tackling WSRP. Even though VNS had obtained better results than the proposed methods in this thesis, when results were available, this study aided in better understanding of the usability of GAs under the combined settings. This was the first time that a study on the design components of a GA, that has been used to solve WSRP.

7.4 Summary and Final Remarks

Using synergy among the operators provided better results than using one operator at a certain stage of the search [89]. This concept was used in this chapter by proposing an adaptive multiple crossover framework (AMC) to be integrated in adaptive GA that tackles WSRPs. The aim is to improve the GA efficiency.

In this chapter, six different crossover operators, well known and cost-based, were selected as part of the AMC method. The crossovers were rewarded with scores during the learning process, which affected their application rates during the run, and were later adjusted with values that reflect the search environment.

Rewards were given to a crossover according to its effectiveness at the current stage. To evaluate a crossover, three performance measurements were utilised to provide a score for each operator. The crossover ability to provide an offspring with fitness improvements, to generate an offspring that is different from their parents or to provide fitness improvements and diverse solutions at the same time.

Based on the performance measurements, three variations of the AMC were implemented (AMC_f, AMC_d, AMC_h), with six different cycle sizes on 42 different instances of WSRP problems. The cycle size, which provided the highest number of best solutions for one method, was selected in a more in-depth analysis in this chapter. Further analysis has found that all the methods applied have different performances, with AMC_d having

the best score, the highest number of best solutions with the highest computational time among all methods.

In general, the diversity-based measurement required more time than fitness-based measurement. However, it provided better quality solutions. The combination of the diversity and the fitness measurements have obtained less computational times than using separate methods. This result is related to the scoring process used. If only one measurement is used, a small increase in the score occurs. On the other hand, if both performance measurements are used a large increase in the score occurs. Thus, the scores in AMC_h were higher than AMC_f and AMC_d, henceforth, less time is required.

Another important finding was that the highest average value of application rates among all methods was obtained by FCX and PMFCX crossovers. The results showed that PMFCX crossover was the most utilised crossover in all methods. This observation complements those of earlier studies in this thesis by presenting the strength of problem-specific crossovers when tackling WSRPs.

Results were compared with the indirect GA (non-adaptive) and a random uniform variation of the AMC (no learning). The AMCs had outperformed the stated methods, especially when diversity measurement was applied.

The key strength of this study is the combinations of adaptive settings and multiple crossover method that resulted in better quality solutions in less computational time. These findings have major implications that using a full adaptive GA is the best method for the combined scheduling and routing problem settings.

The contribution of this study has been to confirm that the combined performances of all the operators were better than applying one operator in isolation. Using the best performing crossovers, as suggested by the offline tuning, along with traditional crossovers have delayed convergence and therefore encouraged the search.

Chapter 8

Conclusions and Future Work

8.1 Introduction

This chapter concludes the thesis by summarising the work done, reflecting on the research accomplished in this thesis and identifying interesting areas of research, arising from the thoughts presented in this thesis.

Different design approaches of genetic algorithm (GA), are presented in this thesis, which can tackle real-world instances of Workforce Scheduling and Routing Problems (WSRP). The WSRP involves the scheduling and routing of a workforce to visit customers at different locations to complete a set of tasks or activities. Therefore, WRSP is a combination of two problems, personnel scheduling and routing.

Generating good solutions for WSRP instances is very critical in real-world problem settings. Recent evidence by [17, 78] suggests that the problem degree of difficulty grows with an increase in the number of assignment in each instance and henceforth the problem size. The larger the problem, the more time is required to obtain reasonable solutions. Thus, more efficient and faster methods are needed to solve WSRP instances.

GAs have been shown to be effective approaches to find solutions for problems combining scheduling and routing, where exact methods are less effective, e.g., [30, 118]. Thus, this thesis has investigated the usability of GA methods to help obtain feasible solutions. However, in those works and other established evolutionary algorithms (EAs) for WSRPs [67, 93], similar evolutionary components were reused from WSRP-related

problems, i.e., resource scheduling and vehicle routing problem, such as representation types and traditional operators without any attempts for tuning the parameter values.

Most studies of different WSRP solution methods had suggested that the combined nature of the problem required more insights on the implementation. For example, greedy heuristics [17], an initialisation heuristic in a particle swarm optimisation (PSO) [5], problem-specific recombination and mutation operators in a mematic algorithm (MA) [70] and constrictive heuristics in variable neighbourhood search (VNS) [127].

This thesis has undertaken the design of problem-specific evolutionary components, customised for WSRP, which can be included in different EAs. This work also investigated various methods to increase the efficiency of the proposed components that can also improve the GA performance. As a result, different GAs were developed that used common objectives and constraints to preform significantly sound comparisons, including those related to workers' skill qualifications, time/area availability and time conflicts.

The same GA process was used on 42 different instances of WSRP problems for methods proposed in Chapter 5 to Chapter 7, by changing some elements to differentiate one method from its predecessor, i.e., repair heuristics, special operators and adaptive parameter control framework. However, the study presented on Chapter 4 has utilised two variants of simple GAs to identify the relative qualities and shortcomings of the proposed genetic operators on WSRPs rather than proposing a competitive algorithm. Thus, the design was restricted to utilising basic evolutionary procedures without any improvements. More details are stated in the next section.

8.2 Review of Contributions

In this section, a revision of the main contributions of this thesis is provided.

8.2.1 Selecting genetic operators to maximise preference satisfaction in WSRP

Reducing the number of soft constraint violations was one of the major challenges when applying GAs to WSRP. Therefore, in Chapter 4, the performances of various generic and problem-specific genetic operators were investigated for their ability to re-

duce the number of constraint violations. Thirty-four scenarios of the home health care scheduling (HHC) were tackled in this study by two well-known GAs: the steady state genetic algorithm (SSGA) and the generational genetic algorithm (GenGA). The study identified the best combinations of operators for WSRP, provided by a performance comparison between a set of operators in two well-known GAs. The study also investigated the effect of different parameter settings to better understand the operators' performances in a simple GA for WSRP.

The random initialisation has resulted in infeasible chromosomes. Thus, two repair heuristics were used as constraints handling methods to improve the quality of results. Not using repair operators resulted in many constraint violations, regardless of the combination of genetic operators used. Hence, genetic operators were co-dependent on the use of repair heuristics that use problem domain knowledge. Contributions of Chapter 4 were as follows.

- A time conflict reduction heuristic (TCR), was used to improve the quality of the initial solution. The GA converged faster to better solutions when using TCR than when using completely random initial solutions. Attempting to repair one constraint while other constraints were discarded has resulted in poor solutions quality in comparison to different methods used for WSRP.

- A worker suitability repair operator (WSR), was used to reduce any violation of constraints in two stages of the GAs, after the TCR and then again after the mutation operator in the optimisation process. The WSR operator succeeded in improving the quality of the solutions, regardless of the combination of genetic operators used. Though there was an apparent reduction in the violations, some still existed.

- Two cost-based search operators. A cost-based uniform crossover (CBUX), where an offspring inherits available workers in a specified position from parents. A cost-based mutation (CBM), seeks to insert workers assigned to visits that were among those considered as *preferred workers* for that visit. Despite the fact that the proposed operators were neither the best nor the next best in all WSRP scenario, the

study presented some insight into how to design better performance-wise operators similar to the ones presented in Chapter 5.

In addition to the proposed problem-specific operators, existing well-known generic operators, three mutation operators and seven crossover operators, which have performed well on related problems, were studied to select the best operators for the problem. From the experimental results, existing operators performed the best under WSRP, when straightforward GAs were used. Therefore, a simple GA does not require special or tailored operators for this type of problems. However, the solutions were far from optimal (shown in Table 2.5) indicating the need for a more efficient algorithm.

The tendency for constraints violations to exist in the initial solutions, when using a random method, can be controlled by using constraints handling methods, which can construct initial feasible individuals. Thus, an indirect representation scheme was proposed in the next chapter to ensure the feasibility of the initial population.

8.2.2 A Baseline GA for WSRP

Chapter 5 presented the design of problem-specific elements on a GA to tackle 42 different instances of WSRP and evaluated their offline tuning. An indirect chromosome encoding scheme was designed to produce feasible initial solutions only.

Different operators were applied in this chapter. Some of these operators have proven their efficiency in Chapter 4 and others were specially designed for the problem and solution representation considered in the chapter. Four crossovers and two mutation operators were taken from the literature, four crossover operators and two mutation operators were designed for WSRP context.

Using methods such as greedy heuristics in a crossover or a gene flip in a mutation has been applied in the literature previously. However, in Chapter 5, these methods were used in the design of the proposed operators with respect to the WSRP and the indirect representation scheme. Contributions of Chapter 5 were as follows.

- An indirect chromosome encoding scheme, designed to produce feasible solutions. The results showed that the proposed encoding was effective in maintaining the feasibility of solutions and helped the GA to perform well depending on different

problem sizes, indicating the merit to use constrictive heuristics to build feasible initial solutions for WSRP. Rather than using randomly generated individuals that require time to be repaired.

- Four cost-based crossover operators, including cost-based greedy crossover (CGX), partially matched greedy crossover (PMGreedyX), flat-costs crossover (FCX) and partially matched flat-costs crossover (PMFCX). For smaller instances, greedy crossovers, such as CGX and PMGreedy, obtained the best results. However, for larger instances, the greedy operators were computationally expensive. Thus, FCX crossover provided the best results in faster computational-times.

- Two mutation operators, flat-cost flip (FCF) and zero-flip. Among the mutation operators implemented, the FCF made a significant difference by improving the performance of the GA. It has introduced more diversity to the population by inserting new workers to an individual regardless of the overall cost.

Comprehensive computational experiments were conducted using various settings of crossover rate, mutation rate, and population size, for each combination of one crossover operator and one mutation operator with one population size. Results in Chapter 5 were compared to published results to better understand how to enhance the GA performance.

The comparison showed that the GA was capable of finding some best-known results for the larger problem instances and matching the best-known results for some of the smaller ones. Hence, the proposed baseline GA with problem-specific components was an effective evolutionary approach to solve the highly-constrained WSRP scenarios considered. However, the parameter-tuning was impractical and computationally expensive. Thus, the next research has aimed to improve the performance of this baseline GA.

8.2.3 Diversity-based Adaptive GA

Chapter 6 identified elements in the baseline GA design, which affected the degradation of the performance in later generations. Thus, adaptive parameter control method was integrated into the baseline GA. Adjusting parameter values have shifted the search

into more promising areas. This study aimed to investigate the effects of using adaptive parameter tuning techniques in enhancing the performance of the GA for WSRP.

Three aspects were considered: an adaptation of mutation/crossover parameters, dynamic population re-sizing, and a random crossover exchange. Parameters were changed to values that reflected the individual distribution and differences in a population, i.e., the population diversity.

Six GA variations were triggered by stagnation, i.e., no improvements on the cost values. Including adaptive mutation/crossover rates AGA, adaptive rates with dynamic population re-sizing AGADP, dynamic population re-sizing only GADP, adaptive rates with random crossover alterations RCAGA, random crossover with dynamic population re-sizing RCGADP, adaptive rates with dynamic population re-sizing and random crossover alterations RCAGADP. Extensive computational experiments were conducted on these variations. Results obtained were compared to the baseline GA. Contributions of Chapter 6 were as follows.

- Adaptive parameters rates control methods, extended from the literature, were applied. This method proved to be a key strength of the study as it provided better results for WSRP scenarios, especially in problem sets D and F, by avoiding premature convergence. Thus, it can be claimed that this study has established the efficiency of using adaptive rates for real-world problems.

- Dynamic population re-sizing was designed specifically for the baseline GA and can be integrated into other EAs. The size was changed dynamically during the run when there was no improvement, with respect to the step size, while always preserving 10% of the best-found solutions. Two methods of population re-sizing were used: population increasing and shrinking. Dynamic population re-sizing achieved the highest levels of diversity in A, B and C problem sets with fitness landscape change throughout the generations. This method aided in obtaining good solutions quickly in smaller instances. However, its performance was limited when used on problem sets D, E, and F.

The random crossover exchange method proposed in the chapter has selected an

arbitrary crossover, if there were no improvements, for the last four generations. The algorithm benefited from the performance of a group of crossover functions used rather than using one crossover throughout the run. This method allowed the crossover to continue to find better solutions and exhibited good fitness results, especially in A, B, C and E instance sets. In contrast, random crossover exchange was proven to be computationally expensive for D and F problem sets and was unable to converge to better cost values while it was statically worse than AGA, GADP, and AGADP.

Different adaptive variations were suited for different WSRP scenarios and all methods had outperformed the baseline GA. The highest number of best solutions was obtained when using AGA, especially in large problems with faster computation-time than the baseline GA. On the other hand, dynamic population re-sizing methods have succeeded in obtaining a diverse set of solutions, but with worse solutions than AGA. This was because of the extensive time required for computing the adjustments.

Adaptive population size was updated more often than the operator rates, due to the method sensitivity to change according to the population diversity, which varied from one generation to the other. It would be interesting to assess the effects of less change in the population size on the computational times and henceforth the solutions quality.

More research work was needed, especially towards the improvement of the crossover selection, to advance the algorithm efficiency by achieving a good balance between having a diverse set of solutions and a crossover selection.

8.2.4 Adaptive Multiple-Crossovers GA

Chapter 7 proposed a novel adaptive multiple crossover GA framework based on a learning model that was customised to tackle the WSRP scenarios. The idea was to adaptively distribute the application of the crossover function P_c over all the operators, giving high application rates to the best performers while still using the worse ones. The crossovers application rates were distributed based on the learning outcomes in regards to the environment changes. The goal of this study was to extend the diversity-based adaptive GA presented previously by improving the operator selection efficiency. The contributions of Chapter 5 included the following.

- Adaptive multiple crossover scheme (AMC), with a fixed number of crossovers, was used during the search. Crossovers were rewarded with scores during the learning process, which affected their application rates during the run. They were later adjusted with values that reflected the search environment. Rewards values were given to a crossover according on its effectiveness at the current stage. To evaluate a crossover, performance measurements were utilised to provide a score for each operator, i.e., the crossover ability to produce dissimilar offspring than their parents with fitness improvements. This method provided better and more diverse solutions.

- The crossover application rates distribution matrix (CRDM) recorded the scores during each cycle, and then transformed it as an application rate. The obtained experimental results showed that the combined performances of all the operators were better than using one operator in isolation. The results showed that PMFCX crossover was the most utilised crossover in all methods indicating the benefit of using well designed problem-specific operators. Applying good performance crossovers with traditional crossovers have avoided premature convergence and therefore encouraged the search further.

- Based on the performance measurements, three different variations of the AMC were implemented with different cycle sizes. The fitness-based calculations AMC_f have been proven to be faster than the diversity-based calculations AMC_d, especially when using larger cycle sizes. The cycle-size which provided the highest number of best solutions for each method was selected for more examinations to apply an in-depth analysis. Even though there was no statistically significant difference among all methods, diversity-based calculations AMC_d was the best in comparison to other methods. The combinations between the diversity-based measurement and the fitness-based measurement AMC_h have obtained better computation-times than the separate methods as the scores were larger and henceforth the rewarding scheme was faster.

- A diversity-based adaptive multiple crossovers GA ($AMCGA$). This method in-

tegrates the adaptive rates adjustments framework (AGA) presented in Chapter 6 into the best performing multiple crossover framework, in this case, AMC_d that provided the best results. The AGA was selected due to its effectiveness to alter the rates quickly while providing the good results.

Extensive computational experiments were conducted using the different proposed variations of AMC. Results obtained were compared to the baseline GA and a randomly uniform variation of the AMC with no learning process. It was suggested that different AMC methods applied to different datasets could obtain different results and henceforth each problem set benefited from each method accordingly.

This study has shown that AMCs have a considerable influence on the performance of the GA. This was because of the interactions between multiple operators using a learning scheme that has provided better results than using a uniform distribution.

Methods that used diversity-based performance measurements have been proven to be the best AMC method for WSRP, indicating the importance of biased recombination operators that highlight the significance of distances between parents and offspring.

Thus, this method was included in a full adaptive GA that combined adaptive settings control and multiple crossover framework that resulted in better quality solution in less computational time. The findings of this thesis could be used to help to understand the importance of adaptive components in EAs when tackling the combined scheduling and routing problem settings.

Further work needs to be carried out to validate the features of the set of operators used in this study, such as the number of operators and the type of operators. Previous chapters have claimed that greedy operators were better for small WSRP instances. However, they were excluded from the study because they were time-consuming. Thus, this study must be repeated using greedy operators to show that the result can be improved, given a much higher computation-time. Thereupon, another factor should be included as a performance measurement: the time required to generate an offspring by an operator. A higher score will be given for operators with less computational-time.

8.3 Future Work

Further work could usefully explore how to visualise the solutions in a graphical user interface (GUI). A GUI in this field would be of great help by enabling users to play a central role in the optimisation loop, by performing actions at an interaction solution levels. A user can select the solution elements, such as the number of visits, number of workers and priority of objectives. It can also show a 2-dimensional view of the WSRP solution. Additionally, a user might have the option to select a crossover operator over another or select a set of operators (crossovers or mutations) to be included in an AMC.

The proposed GAs harnesses the power of the genetic components to solve the WSRP instances considered. Even though the results have proven the efficiency of the GA method to tackle WSRP, some elements of the proposed methods can be improved. This section identifies some areas of interest for future work arising from this thesis.

Further research concerning WSRP

1. The work in this thesis used a weighted sum for the objective function that prioritised avoiding conflicted assignments over the preferences or the operational costs. Most of the objective functions used in related problems to the WSRP have focused on measuring their plan by a multi-objective function to satisfy hard constraints [30, 118]. Given that, different objective-function calculations need to be investigated. Also, adaptive penalty function can be used by extracting information from the current population, such as the level of violation of each constraint and each individual feasibility [87].

2. Mainly, soft constraints such as work breaks and regulations are not considered in this thesis. According to experimentation, satisfying soft constraints can be challenging in WSRP context. Therefore, fuzzy sets are suggested to be investigated to derive the combined satisfaction degrees of constraints.

   Fuzzy sets introduces gradualism in the constraint satisfaction in the same manner as the researched study by [126] on timetabling. Fuzzy IF-THEN rules are defined to derive the satisfaction degree of each of the constraint. The satisfaction degree

of planning visits with longer time periods is derived using the linguistic variables time duration of a visit and time availability. Duration of a visit is described by linguistic terms small, medium, and large, while time-window is assessed as early, middle, and late with respect to its position in the weekly planning period. Similarly, the linguistic variables distance between two visits is used to reason the satisfaction degree of the proximity of plan. This seems to be a sound direction towards progressing this research.

3. Real-world setting requires scheduling a weekly plan for workers. A weekly plan was proposed to be included at a later stage of the research program. Such step did not occur, and the entire research program was spent using a daily variation of the WSRP. A future study investigating GA performance on the weekly plan would be very interesting.

4. In the current research, more constraints should have been included such as teaming, repeated visitations, and shift coverage constraints, e.g., consecutive working hours regulations per week and days off.

5. Real-time data denotes that data might change due to changes in the environment [110]. One method that could be investigated is calculating distances according to the transportation type in a real-time. Transportation vehicles will have different speeds, may take different roads and distances will change according to the transportation type.

Further research concerning the Solution Methods

1. A major limitation of this study is the large number of objective function evaluations required for this problems. A further study could utilise parallel function evaluations that can reduce the computation-time by having independent performances [155].

2. An issue may arise on whether the adaptive method can further improve the performances of more competitive EAs when used for WSRP scenarios. An interesting examination would be to apply and compare this variation on different

evolutionary algorithms to examine the exploitation powers in the context of a dynamic/adaptive mechanism.

3. The interactions between different parameter settings are more difficult to evaluate, especially when feedback-based adaptation is used. Therefore, self-adaptation might be a better alternative to be used when combining different forms of control as parameter settings to be evolved in addition to the solutions [47].

4. The results have indicated that more time is required when using dynamic-population re-sizing in the adaptive GA. Thus, it will be interesting to assess the effects of less adaptive adjustments in this method on the computational-time of the adaptive GA and the quality of the generated results.

5. An investigation of a wider range of cycle and memory sizes should be carried out. The reason for the selected parameters has been noted in Chapter 7, however more evidence of the suitability of these sizes is needed.

6. So far, crossovers operators were only used as part of the AMC method. Another way of creating a multiple operator GA is to use different mutation methods even though FCF has obtained the best results with the baseline GA. Combining this with results in Chapter 7 could also provide interesting results.

7. Another potential extension of this study is to compare the results against previous methods of adaptive operator selections to investigate the efficiency of using AMC in comparison to existing methods in the literature. This can provide some insights into the strength of this method.

Bibliography

[1] O. Abdoun and J. Abouchabaka. A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem. *CoRR*, abs/1203.3097, 2012.

[2] D. H. Ackley. Introduction. In *A Connectionist Machine for Genetic Hillclimbing*, pages 1–28. Springer US, Boston, MA, 1987.

[3] U. Aickelin and K. Dowsland. Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem. *arXiv preprint arXiv:0802.2001*, 2008.

[4] U. Aickelin and K. A. Dowsland. An indirect genetic algorithm for a nurse-scheduling problem. *Computers & Operations Research*, 31(5):761 – 778, 2004.

[5] C. Akjiratikarl, P. Yenradee, and P. R. Drake. PSO-based algorithm for home care worker scheduling in the UK. *Computers & Industrial Engineering*, 53(4):559 – 583, 2007.

[6] H. Allaoua, S. Borne, L. LÃľtocart, and R. W. Calvo. A matheuristic approach for solving a home health care problem. *Electronic Notes in Discrete Mathematics*, 41:471 – 478, 2013.

[7] C. Artigues. The resource-constrained project scheduling problem. In *Resource-constrained project scheduling: models, algorithms, extensions and applications*, pages 19–35. John Wiley & Sons, 2013.

[8] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256, 2002.

[9] J. E. Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of an International Conference on Genetic Algorithms and their applications*, pages 101–111. Hillsdale, New Jersey, 1985.

[10] T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006.

[11] J. Belluz, M. Gaudesi, G. Squillero, and A. Tonda. Operator selection using improved dynamic multi-armed bandit. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 1311–1317, New York, NY, USA, 2015. ACM.

[12] J. Berger, M. Salois, and R. Begin. A hybrid genetic algorithm for the vehicle routing problem with time windows. In *Advances in Artificial Intelligence*, pages 114–127. Springer, 1998.

[13] S. Bertels and T. Fahle. A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. *Computers & Operations Research*, 33(10):2866–2890, 2006.

[14] D. Bredstrom and M. Ronnqvist. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research*, 191(1):19 – 31, 2008.

[15] K. Bryan and Y. Shibberu. Penalty functions and constrained optimization. *Dept. of Mathematics, Rose-Hulman Institute of Technology. http:// www. rosehulman. edu/˜ bryan/lottamath/penalty. pdf*, 2005.

[16] A. E. Carter and C. T. Ragsdale. A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European journal of operational research*, 175(1):246–257, 2006.

[17] J. A. Castillo-Salazar, D. Landa-Silva, and R. Qu. A greedy heuristic for workforce scheduling and routing with time-dependent activities constraints. In *International*

Conference on Operations Research and Enterprise Systems (ICORES), pages 367–375, Lisbon, Portugal, 2015. Scipress.

[18] J. A. Castillo-Salazar, D. Landa-Silva, and R. Qu. Workforce scheduling and routing problems: literature survey and computational study. *Annals of Operations Research*, 239(1):39–67, 2016.

[19] Y. Chang and L. Chen. Solve the vehicle routing problem with time windows via a genetic algorithm. *Discrete and continuous dynamical systems supplement*, pages 240–249, 2007.

[20] S. Chatterjee, C. Carrera, and L. A. Lynch. Genetic algorithms and traveling salesman problems. *European journal of operational research*, 93(3):490–5102, 1996.

[21] E. Cheng and J. L. Rich. A home health care routing and scheduling problem. *Houston, Texas*, 1998.

[22] P. C. Chu and J. E. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of heuristics*, 42(1):63–86, 1998.

[23] V. Cicirello and S. Smith. Modeling ga performance for control parameter optimization. In *GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference*, July 2000.

[24] V. A. Cicirello. Non-wrapping order crossover: An order preserving crossover operator that respects absolute position. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 1125–1132, New York, NY, USA, 2006. ACM.

[25] C. A. C. Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11âĂŞ12):1245 – 1287, 2002.

[26] P. Consoli and X. Yao. Diversity-driven selection of multiple crossover operators for the capacitated arc routing problem. In C. Blum and G. Ochoa, editors, *Pro-*

ceedings of the 14th European Conference on Evolutionary Computation in Combinatorial Optimisation, EvoCOP'14, pages 97–108. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[27] C. Contreras-Bolton and V. Parada. Automatic combination of operators in a genetic algorithm to solve the traveling salesman problem. *PLOS ONE*, 10(9): 1–25, 09 2015.

[28] C. Contreras-Bolton, G. Gatica, C. R. Barra, and V. Parada. A multi-operator genetic algorithm for the generalized minimum spanning tree problem. *Expert Systems with Applications*, 50:1 – 8, 2016.

[29] P. Cowling, G. Kendall, and L. Han. An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 2, pages 1185–1190. IEEE, 2002.

[30] P. Cowling, N. Colledge, K. Dahal, and S. Remde. The trade-off between diversity and quality for multi-objective workforce scheduling. In *Proceedings of the 6th European Conference on Evolutionary Computation in Combinatorial Optimization*, EvoCOP'06, pages 13–24, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[31] M. Črepinšek, S.-H. Liu, and M. Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.*, 45(3):35:1–35:33, July 2013.

[32] E. D. de Jong, R. A. Watson, and J. B. Pollack. Reducing bloat and promoting diversity using multi-objective methods. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 11–18, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.

[33] K. De Jong, D. B. Fogel, and H.-P. Schwefel. A history of evolutionary computation. *Evolutionary Computation I: Basic Algorithms and Operators (T. Baeck, D. Fogel, and T. Michalewicz, eds.)*, pages 40–51, 2000.

[34] K. A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, Ann Arbor, MI, USA, 1975.

[35] K. Deep and H. Mebrathu. Combined mutation operators of genetic algorithm for the travelling salesman problem. *International Journal of Combinatorial Optimization Problems and Informatics*, 2(3):2, 2011.

[36] Y. Deng, Y. Liu, and D. Zhou. An improved genetic algorithm with initial population strategy for symmetric tsp. *Mathematical Problems in Engineering*, 2015: 212794, 2015.

[37] G. Desaulniers, O. B. Madsen, and S. Ropke. The vehicle routing problem with time windows. In *The vehicle routing problem*, pages 119–152. Siam, 2014.

[38] M. K. Dhodhi, I. Ahmad, A. Yatama, and I. Ahmad. An integrated technique for task matching and scheduling onto distributed heterogeneous computing systems. *Journal of Parallel and Distributed Computing*, 62(9):1338 – 1361, 2002.

[39] A. Dohn, M. S. Rasmussen, T. Justesen, and J. Larsen. The home care crew scheduling problem. In *Proceedings of the 1st international conference on applied operational research (ICAORâĂŹ08), Yerevan, Armenia. Lecture Notes in Management Science*, volume 1, pages 1–8, 2008.

[40] A. Dohn, E. Kolind, and J. Clausen. The manpower allocation problem with time windows and job-teaming constraints: A branch-and-price approach. *Computers & Operations Research*, 36(4):1145 – 1157, 2009.

[41] A. Dohn, M. S. Rasmussen, and J. Larsen. The vehicle routing problem with time windows and temporal dependencies. *Networks*, 58(4):273–289, 2011.

[42] H. Dong, J. He, H. Huang, and W. Hou. Evolutionary programming using a mixed mutation strategy. *Information Sciences*, 177(1):312–327, 2007.

[43] M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli, and K. V. Price, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[44] M. Dorigo and L. M. Gambardella. Ant colonies for the travelling salesman problem. *BioSystems*, 43(2):73–81, 1997.

[45] M. Drozdowski. Scheduling multiprocessor tasks âĂŤ an overview. *European Journal of Operational Research*, 94(2):215 – 230, 1996. ISSN 0377-2217.

[46] A. Eiben and S. Smit. Parameter tuning for configuring and analysing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19 – 31, 2011.

[47] A. E. Eiben, M. Horvath, W. Kowalczyk, and M. C. Schut. Reinforcement learning for online control of evolutionary algorithms. In *Proceedings of the 4th International Workshop on Engineering Self-Organising Systems*, pages 151–160. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[48] S. Elaoud, J. Teghem, and T. Loukil. Multiple crossover genetic algorithm for the multiobjective traveling salesman problem. *Electronic Notes in Discrete Mathematics*, 36:939 – 946, 2010.

[49] L. J. Eshelman. The CHC adaptive search algorithm : How to have safe search when engaging in nontraditional genetic recombination. *Foundations of Genetic Algorithms*, pages 265–283, 1991.

[50] L. J. Eshelman. Genetic algorithms. In T. Back, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, 1st edition, 1997.

[51] L. J. Eshelman. Genetic algorithms. *Evolutionary Computation I: Basic Algorithms and Operators (T. Baeck, D. Fogel, and T. Michalewicz, eds.)*, 1:64–80, 2000.

[52] E. Falkenauer and S. Bouffouix. A genetic algorithm for job shop. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 824–829. IEEE, 1991.

[53] Á. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Analyzing bandit-based adaptive operator selection mechanisms. *Annals of Mathematics and Artificial Intelligence*, 60(1):25–64, 2010.

[54] C. Fikar and P. Hirsch. A matheuristic for routing real-world home service transport systems facilitating walking. *Journal of Cleaner Production*, 105:300 – 310, 2015.

[55] M. L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124, 1981.

[56] D. B. Fogel. A parallel processing approach to a multiple travelling salesman problem using evolutionary programming. In *Proceedings of the fourth annual symposium on parallel processing*, pages 318–326, 1990.

[57] D. B. Fogel. Principles of evolutionary processes. In T. Back, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages 23–26. IOP Publishing Ltd., Bristol, UK, 1997.

[58] S. García, D. Molina, M. Lozano, and F. Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. *Journal of Heuristics*, 15(6):617, 2008.

[59] G. Ghiani, G. Laporte, and R. Musmanno. Introducing logistics systems. In *Introduction to Logistics Systems Planning and Control*, pages 1–23. John Wiley & Sons, Ltd, 2005.

[60] B. M. Ginley, J. Maher, C. O'Riordan, and F. Morgan. Maintaining healthy population diversity using adaptive crossover, mutation, and selection. *IEEE Transactions on Evolutionary Computation*, 15(5):692–714, Oct 2011.

[61] D. Goldberg. Genetic algorithms in optimization, search and machine learning. *Addison Wesley, New York. Eiben AE, Smith JE (2003) Introduction to Evolutionary Computing. Springer. Jacq J, Roux C (1995) Registration of non-segmented images using a genetic algorithm. Lecture notes in computer science*, 905:205–211, 1989.

[62] D. Goldberg, K. Deb, and B. Korb. Messy genetic algorithms: motivation, analysis, and first results. *Complex Systems*, 3(5):493–530, 1989.

[63] D. E. Goldberg. Goals and elements of ga design. In *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, pages 51–57. Springer, 2002.

[64] D. E. Goldberg and R. Lingle, Jr. Alleleslociand the traveling salesman problem. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 154–159, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.

[65] J. J. Grefenstette, R. Gopal, B. J. Rosmaita, and D. V. Gucht. Genetic algorithms for the traveling salesman problem. In *Proceedings of the 1st international conference on genetic algorithms*, pages 160–168. L. Erlbaum Associates Inc., 1985.

[66] M. Gröbner and P. Wilke. Optimizing employee schedules by a hybrid genetic algorithm. In *Applications of Evolutionary Computing*, pages 463–472. Springer, 2001.

[67] M. Gúnther and V. Nissen. A comparison of neighbourhood topologies for staff scheduling with particle swarm optimisation. In B. Mertsching, M. Hund, and Z. Aziz, editors, *KI 2009: Advances in Artificial Intelligence*, volume 5803 of *Lecture Notes in Computer Science*, pages 185–192. Springer, 2009.

[68] P. Hansen and N. Mladenović. Variable neighborhood search. In E. K. Burke and G. Kendall, editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 211–238. Springer US, Boston, MA, 2005.

[69] S. Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, 45(7):733–750, 1998.

[70] G. Hiermann, M. Prandtstetter, A. Rendl, J. Puchinger, and G. R. Raidl. Metaheuristics for solving a multimodal home-healthcare scheduling problem. *Central European Journal of Operations Research*, 23(1):89–113, 2015.

[71] J. H. Holland. Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence. *Ann Arbor: University of Michigan Press*, 1, 1975.

[72] J. H. Holland. Adaptation of codings and representations. In *Adaptation in Natural and Artificial Systems:An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, pages 141–158. MIT Press, 1992.

[73] A. Kapsalis, V. J. Rayward-Smith, and G. D. Smith. Solving the graphical steiner tree problem using genetic algorithms. *The Journal of the Operational Research Society*, 44(4):397–406, 1993.

[74] S. Karakatič and V. Podgorelec. A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing*, 27:519–532, 2015.

[75] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948, Nov 1995.

[76] K. Kotecha, G. Sanghani, and N. Gambhava. Genetic algorithm for airline crew scheduling problem using cost-based uniform crossover. In S. Manandhar, J. Austin, U. B. Desai, Y. Oyanagi, and A. K. Talukder, editors, *AACC*, volume 3285 of *Lecture Notes in Computer Science*, pages 84–91, Kathmandu, Nepal, 2004. Springer.

[77] A. J. Kulkarni and K. Tai. Probability collectives: a multi-agent approach for solving combinatorial optimization problems. *Applied Soft Computing*, 10(3):759–771, 2010.

[78] W. Laesanklang and D. Landa-Silva. Decomposition techniques with mixed integer programming and heuristics for home healthcare planning. *Annals of Operations Research*, pages 1–35, October 2016.

[79] W. Laesanklang, R. Pinheiro, H. Algethami, and D. Landa-Silva. Extended decomposition for mixed integer programming to solve a workforce scheduling and routing problem. In D. de Werra, G. H. Parlier, and B. Vitoriano, editors, *4th International Conference on Operations Research and Enterprise Systems (ICORES)*, volume 577 of *Communications in Computer and Information Science*, pages 191–211. Springer International Publishing, Lisbon, Portugal, December 2015.

[80] W. Laesanklang, D. Landa-Silva, and J. A. Castillo-Salazar. Mixed integer programming with decomposition for workforce scheduling and routing with time-dependent activities constraints. In *Proceedings of the 5th International Conference on Operations Research and Enterprise Systems (ICORES)*, pages 23–27, Rome, Italy, 2016.

[81] D. Landa-Silva. *Metaheuristic and multiobjective approaches for space allocation.* PhD thesis, University of Nottingham, 2003.

[82] D. Landa-Silva and K. N. Le. A simple evolutionary algorithm with self-adaptation for multi-objective nurse scheduling. In *Adaptive and Multilevel Metaheuristics*, pages 133–155. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[83] P. Larrañaga, C. Kuijpers, R. Murga, I. Inza, and S. Dizdarevic. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13(2):129–170, 1999.

[84] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13(2):129–170, 1999.

[85] E. Lawler. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* Wiley, New York, 1985.

[86] Z.-J. Lee. A hybrid algorithm applied to travelling salesman problem. In *Networking, Sensing and Control, 2004 IEEE International Conference on*, volume 1, pages 237–242. IEEE, 2004.

[87] A. C. Lemonge, H. J. Barbosa, and H. S. Bernardino. Variants of an adaptive penalty scheme for steady-state genetic algorithms in engineering optimization. *Engineering Computations*, 32(8):2182–2215, 2015.

[88] J. K. Lenstra and A. Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

[89] K. Li, . Fialho, S. Kwong, and Q. Zhang. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 18(1):114–130, Feb 2014.

[90] Y. Liang and K.-S. Leung. Genetic algorithm with adaptive elitist-population strategies for multimodal function optimization. *Applied Soft Computing*, 11(2): 2017 – 2034, 2011. The Impact of Soft Computing for the Progress of Artificial Intelligence.

[91] C. Lin. An adaptive genetic algorithm based on population diversity strategy. In *Genetic and Evolutionary Computing, 2009. WGEC '09. 3rd International Conference on*, pages 93–96, Oct 2009.

[92] S. Lin. Computer solutions of the traveling salesman problem. *The Bell System Technical Journal*, 44(10):2245–2269, Dec 1965.

[93] R. Liu, X. Xie, V. Augusto, and C. Rodriguez. Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. *European Journal of Operational Research*, 230(3):475 – 486, 2013.

[94] F. G. Lobo and C. F. Lima. Adaptive population sizing schemes in genetic algorithms. In *Parameter Setting in Evolutionary Algorithms*, pages 185–204. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[95] S. J. Louis and G. Li. Case injected genetic algorithms for traveling salesman problems. *Information Sciences*, 122(2):201–225, 2000.

[96] J. Lu, N. Fang, D. Shao, and C. Liu. An improved immune-genetic algorithm for the traveling salesman problem. In *Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 4, pages 297–301. IEEE, 2007.

[97] H. Ma, D. Simon, M. Fei, and Z. Chen. On the equivalences and differences of evolutionary algorithms. *Engineering Applications of Artificial Intelligence*, 2013.

[98] Z. S. Ma and A. W. Krings. Dynamic populations in genetic algorithms. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1807–1811. ACM, 2008.

[99] B. Maenhout and M. Vanhoucke. Comparison and hybridization of crossover operators for the nurse scheduling problem. *Annals of Operations Research*, 159(1): 333–353, 2008.

[100] K. F. Man, K. S. Tang, and S. Kwong. Intrinsic characteristics. In *Genetic Algorithms: Concepts and Designs*, pages 45–63. Springer London, London, 1999.

[101] D. Mankowska, F. Meisel, and C. Bierwirth. The home health care routing and scheduling problem with interdependent services. *Health Care Management Science*, 17(1):15–30, 2014.

[102] A. K. M. Masum, M. Shahjalal, F. Faruque, and I. Sarker. Solving the vehicle routing problem using genetic algorithm. *International Journal of Advanced Computer Science and Applications*, 2(7):126–131, 2011.

[103] D. Mattfield. Job shop scheduling. In *Evolutionary Search and the Job-Shop*, pages 7–27. Physica-Verlag HD, 1996.

[104] J. Maturana and F. Saubion. A compass to guide genetic algorithms. In G. Rudolph, T. Jansen, N. Beume, S. Lucas, and C. Poloni, editors, *Proceedings*

of the 10th International Conference on Parallel Problem Solving from Nature, PPSN'10, pages 256–265. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[105] L. Min and W. Cheng. A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines. *Artificial Intelligence in Engineering*, 13 (4):399 – 403, 1999.

[106] M. Misir, K. Verbeeck, P. D. Causmaecker, and G. V. Berghe. Hyper-heuristics with a dynamic heuristic set for the home care scheduling problem. In *IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010.

[107] M. Misir, P. Smet, K. Verbeeck, and G. Vanden Berghe. Security personnel routing and rostering: A hyper-heuristic approach. In *Proceedings of the 3rd International Conference on Applied Operational Research*, volume 3, pages 193–205. Tadbir, 2011.

[108] M. Mısır, P. Smet, and G. Vanden Berghe. An analysis of generalised heuristics for vehicle routing and personnel rostering problems. *Journal of the Operational Research Society*, 66(5):858–870, 05 2015.

[109] M. Mitchell. Genetic algorithms: An overview. In *An introduction to genetic algorithms*, pages 2–24. MIT press, Cambridge, MA, USA, 1998.

[110] D. Montana, M. Brinn, S. Moore, and G. Bidwell. Genetic algorithms for complex, real-time scheduling. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, volume 3, pages 2213–2218. IEEE, 1998.

[111] R. W. Morrison and K. A. De Jong. Measurement of population diversity. In *Artificial Evolution: 5th International Conference, Evolution Artificielle, EA 2001 Le Creusot, France, October 29–31, 2001 Selected Papers*, pages 31–41. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[112] P. Moscato et al. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826:1989, 1989.

[113] C. Muller, E. H. Magill, P. Prosser, and D. G. Smith. Distributed genetic algorithms for resource allocation. In *Scheduling of Production Processes*, pages 70–78. Ellis Horwood, 1993.

[114] T. Murata and H. Ishibuchi. Performance evaluation of genetic algorithms for flowshop scheduling problems. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 812–817. IEEE, 1994.

[115] T. Murata, H. Ishibuchi, and H. Tanaka. Genetic algorithms for flowshop scheduling problems. *Computers & Industrial Engineering*, 30(4):1061 – 1071, 1996.

[116] N. Musliu. Heuristic methods for automatic rotating workforce scheduling. *International Journal of Computational Intelligence Research*, 2(4):309–326, 2006.

[117] M. Mutingi and C. Mbohwa. Home health-care worker scheduling: A group genetic algorithm approach. In *Proceedings of the World Congress on Engineering*, volume 1, pages 3–5, London, UK, 2013.

[118] M. Mutingi and C. Mbohwa. Health-care staff scheduling in a fuzzy environment: A fuzzy genetic algorithm approach. In *Conference Proceedings (DFC Quality and Operations Management)*. International Conference on Industrial Engineering and Operations Management, 2014.

[119] I. Oliver, D. Smith, and J. R. Holland. A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 224–230. L. Erlbaum Associates Inc., 1987.

[120] F. A. Omara and M. M. Arafa. Genetic algorithms for task scheduling problem. *Journal of Parallel and Distributed Computing*, 70(1):13 – 22, 2010.

[121] Y.-S. Ong, M.-H. Lim, N. Zhu, and K.-W. Wong. Classification of adaptive memetic algorithms: a comparative study. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(1):141–152, Feb 2006.

[122] E. Osaba, R. Carballedo, F. Diaz, and A. Perallos. Analysis of the suitability of using blind crossover operators in genetic algorithms for solving routing problems. In *Applied Computational Intelligence and Informatics (SACI), 2013 IEEE 8th International Symposium on*, pages 17–22. IEEE, 2013.

[123] E. Osaba, E. Onieva, R. Carballedo, F. Diaz, and A. Perallos. An adaptive multi-crossover population algorithm for solving routing problems. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2013)*, pages 113–124. Springer, 2014.

[124] Y. Pan, W. X. Zhang, T. Y. Gao, Q. Y. Ma, and D. J. Xue. An adaptive genetic algorithm for the flexible job-shop scheduling problem. In *2011 IEEE International Conference on Computer Science and Automation Engineering*, volume 4, pages 405–409. IEEE, June 2011.

[125] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.

[126] S. Petrovic, V. Patel, and Y. Yang. Examination timetabling with fuzzy constraints. In *Practice and Theory of Automated Timetabling V*, pages 313–333. Springer, 2005.

[127] R. L. Pinheiro, D. Landa-Silva, and J. Atkin. A variable neighbourhood search for the workforce scheduling and routing problem. In *Advances in Nature and Biologically Inspired Computing*, pages 247–259. Springer, Pietermaritzburg, South Africa, 2016.

[128] R. L. Pinheiro, D. L. Silva, R. Qu, E. Yanaga, and A. A. Constantino. Towards an efficient api for optimisation problems data. In *18th International Conference on Enterprise Information Systems (ICEIS 2016)*, pages 89–98, April 2016.

[129] J.-Y. Potvin. Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, 63(3):337–370, 1996.

[130] J.-Y. Potvin. Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, 63(3):337–370, 1996.

[131] J.-Y. Potvin and S. Bengio. The vehicle routing problem with time windows part ii: genetic search. *INFORMS journal on Computing*, 8(2):165–172, 1996.

[132] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985 – 2002, 2004.

[133] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985 – 2002, 2004.

[134] K. Puljić and R. Manger. Comparison of eight evolutionary crossover operators for the vehicle routing problem. *Mathematical Communications*, 18(2):359–375, 2013.

[135] N. J. Radcliffe. Theoretical foundation and properties of evolutionary computation, introduction. In T. Back, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, 1st edition, 1997.

[136] M. S. Rasmussen, T. Justesen, A. Dohn, and J. Larsen. The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies. *European Journal of Operational Research*, 219(3):598 – 610, 2012.

[137] S. S. Ray, S. Bandyopadhyay, and S. K. Pal. New operators of genetic algorithms for traveling salesman problem. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 497–500. IEEE, 2004.

[138] C. R. Reeves. A genetic algorithm for flowshop sequencing. *Computers & Operations Research*, 22(1):5 – 13, 1995.

[139] S. Remde, P. Cowling, K. Dahal, and N. Colledge. Exact/heuristic hybrids using rvns and hyperheuristics for workforce scheduling. In J. Cotta, Carlosand van Hemert, editor, *Proceedings of the 7th European Conference on Evolutionary Computation in Combinatorial Optimization*, EvoCOP'07, pages 188–197. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[140] S. Remde, P. Cowling, K. Dahal, and N. Colledge. Evolution of fitness functions to improve heuristic performance. In *Learning and Intelligent Optimization*, pages 206–219. Springer, 2008.

[141] K.-D. Rest and P. Hirsch. Daily scheduling of home health care services using time-dependent public transport. *Flexible Services and Manufacturing Journal*, 28 (3):495–525, 2016.

[142] F. Rothlauf. Representations for genetic and evolutionary algorithms. *Studies in Fuzziness and Soft Computing*, 104:9–32, 2003.

[143] L. Sakalauskas and G. Felinskas. Optimization of resource constrained project schedules by genetic algorithm based on the job priority list. *Information Technology And Control*, 35(4), 2015.

[144] K. Sastry, D. Goldberg, and G. Kendall. Genetic algorithms. In E. K. Burke and G. Kendall, editors, *Search methodologies: introductory tutorials in optimization and decision support techniques*, pages 97–125. Springer US, 2005.

[145] J. E. Smith. Self-adaptation in evolutionary algorithms for combinatorial optimisation. In C. Cotta, M. Sevaux, and K. Sörensen, editors, *Adaptive and Multilevel Metaheuristics*, pages 31–57. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[146] J. E. Smith. Self-adaptation in evolutionary algorithms for combinatorial optimisation. In *Adaptive and Multilevel Metaheuristics*, pages 31–57. Springer, 2008.

[147] R. E. Smith and E. Smuda. Adaptively resizing populations: Algorithm, analysis, and first results. *Complex Systems*, 9:47 – 72, 1993.

[148] L. V. Snyder and M. S. Daskin. A random-key genetic algorithm for the generalized traveling salesman problem. *European Journal of Operational Research*, 174(1):38 – 53, 2006.

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

[150] K. Sörensen and M. Sevaux. Ma| pm: memetic algorithms with population management. *Computers & Operations Research*, 33(5):1214 – 1225, 2006.

[151] W. M. Spears. Adapting crossover in evolutionary algorithms. In *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pages 367–384. MIT Press, 1995.

[152] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

[153] T. Starkweather, S. McDaniel, K. E. Mathias, L. D. Whitley, and C. Whitley. A comparison of genetic sequencing operators. In *ICGA*, pages 69–76. Citeseer, 1991.

[154] R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11 (4):341–359, Dec. 1997.

[155] D. Sudholt. Parallel evolutionary algorithms. In J. Kacprzyk and W. Pedrycz, editors, *Springer Handbook of Computational Intelligence*, pages 929–959. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[156] G. Syswerda. Uniform crossover in genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 2–9, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[157] G. Syswerda. Schedule optimization using genetic algorithms. In L. Davis, editor, *Handbook of Genetic Algorithms*, pages 332–349. Van Nostrand Reinhold, New York, NY, 1991.

[158] K. Tan, L. Lee, and K. Ou. Artificial intelligence heuristics in solving vehicle routing problems with time window constraints. *Engineering Applications of Artificial Intelligence*, 14(6):825–837, 2001.

[159] K. Tan, L. Lee, Q. Zhu, and K. Ou. Heuristic methods for vehicle routing problem with time windows. *Artificial Intelligence in Engineering*, 15(3):281 – 295, 2001.

[160] H. Terashima-Marín, P. Ross, and M. Valenzuela-Rendon. Application of the hardness theory when solving the timetabling problem with genetic algorithms. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1. IEEE, 1999.

[161] D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, GECCO '05, pages 1539–1546, New York, NY, USA, 2005. ACM.

[162] D. Thierens and D. E. Goldberg. Mixing in genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 38–47, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[163] P. Toth and D. Vigo. An overview of vehicle routing problems. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, pages 1–26. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. ISBN 0-89871-498-2.

[164] H.-K. Tsai, J.-M. Yang, Y.-F. Tsai, and C.-Y. Kao. An evolutionary algorithm for large traveling salesman problems. *Trans. Sys. Man Cyber. Part B*, 34(4): 1718–1729, Aug. 2004.

[165] E. Tsang and C. Voudouris. Fast local search and guided local search and their application to british telecom's workforce scheduling problem. *Operations Research Letters*, 20(3):119–127, 1997.

[166] A. L. Tuson. Adapting operator probabilities in genetic algorithms. Technical report, Master's thesis, Evolutionary Computation Group, Dept. of Artificial Intelligence, Edinburgh University, 1995.

[167] T. Urlings, R. Ruiz, and F. S. Serifoglu. Genetic algorithms with different representation schemes for complex hybrid flexible flow line problems. *International Journal of Metaheuristics*, 1(1):30–54, 2010.

[168] V. Valls, F. Ballestin, and S. Quintanilla. A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 185(2):495–508, 2008.

[169] S. van Rijn, M. Emmerich, E. Reehuis, and T. Bäck. Optimizing highly constrained truck loadings using a self-adaptive genetic algorithm. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 227–234, May 2015.

[170] F. Vavak and T. C. Fogarty. Comparison of steady state and generational genetic algorithms for use in nonstationary environments. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 192–195. IEEE, 1996.

[171] S. Wang and M. Liu. A genetic algorithm for two-stage no-wait hybrid flow shop scheduling problem. *Computers & Operations Research*, 40(4):1064–1075, 2013.

[172] W. Wang. Improved genetic algorithm for vehicle routing problem with time windows. In *Intelligent Computing and Cognitive Informatics (ICICCI), 2010 International Conference on*, pages 203–206. IEEE, 2010.

[173] F. Werner. Genetic algorithms for shop scheduling problems: a survey. *Preprint*, 31(11), 2011.

[174] J. M. Whitacre, T. Q. Pham, and R. A. Sarker. Use of statistical outlier detection method in adaptive evolutionary algorithms. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, pages 1345–1352, New York, NY, USA, 2006. ACM.

[175] D. Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[176] D. Whitley. Permutations. In T. Back, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages 139–149. IOP Publishing Ltd., Bristol, UK, 1997.

[177] Y.-Y. Wong, K.-H. Lee, K.-S. Leung, and C.-W. Ho. A novel approach in parameter adaptation and diversity maintenance for genetic algorithms. *Soft Computing*, 7 (8):506–515, 2003.

[178] S.-H. Woo, S.-B. Yang, S.-D. Kim, and T.-D. Han. Task scheduling in distributed computing systems with a genetic algorithm. In *High Performance Computing on the Information Superhighway, 1997. HPC Asia '97*, pages 301–305, Apr 1997.

[179] S. Yalçındağ, A. Matta, E. Şahin, and J. G. Shanthikumar. The patient assignment problem in home health care: using a data-driven method to estimate the travel times of care givers. *Flexible Services and Manufacturing Journal*, 28(1):304–335, 2016.

[180] G. Zhang, L. Gao, and Y. Shi. An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*, 38(4):3563–3573, 2011.

[181] K. Q. Zhu. A new genetic algorithm for vrptw. In *Proceedings of the International Conference on Artificial Intelligence*, page 311264. Citeseer, 2000.

Appendices

Appendix A

Acronyms and Abbreviations

WSRP	Workforce scheduling and routing problem.
HHC	Home health care problems.
TCR	Time conflict reduction.
WSR	Worker suitability repair.
CBUX	Cost-based uniform crossover.
CBM	Cost-nased mutation.
CGX	Cost greddy crossover.
PMGreedyX	Partially-matched greedy crossover.
FCF	Flat-costs crossover.
PMFCX	Partially-matched flat-costs crossover.
FCF	Flat-costs mutation.
ADGA	Adaptive diversity-based genetic algorithm.
AMC	Adaptive multiple crossover.
CRDM	Crossovers application-rates distribution matrix.
AMCGA	Diversity-based adaptive multiple crossovers genetic algorithm.

Appendix B

Selecting Genetic Operators Results

Chapter 4 proposes an in-depth study on selecting genetic operators to maximise customers' and workers' preference constraints. Experiments were conducted to assess the performance of the proposed GA on the 35 real-world WSRP scenarios from Table 2.4. Performance matrices are used to assess the quality of each mutation and crossover combination in two variations of well-known GAs. This section provides all results using the parameter settings shown in Table 4.1.

Table B.1: Results of performance comparisons of different combinations of operators SSGA and GenGA

SSGA

	RSM				IM				SM				CBM			
		scheduling operators				scheduling operators				scheduling operators				scheduling operators		
	1PX	2PX	UX	HX	1PX	2PX	UX	HX	1PX	2PX	UX	HX	1PX	2PX	UX	HX
Deviation %	0.017419	0.014585	0.018467	0.034114	0.020505	0.011716	0.012628	0.032335	0.016543	0.014768	0.014072	0.040609	0.012067	0.018128	0.014784	0.025248
# Best	0.285714	0.2	0.228571	0.285714	0.285714	0.314286	0.342857	0.057143	0.314286	0.285714	0.371429	0.028571	0.457143	0.085714	0.171429	0.285714
Score	0.590476	0.533333	0.47619	0.4	0.485714	0.590476	0.619048	0.304762	0.504762	0.628571	0.638095	0.228571	0.590476	0.485714	0.504762	0.419048
		Routing operators				Routing operators				Routing operators				Routing operators		
	PMX	OX	CX		PMX	OX	CX		PMX	OX	CX		PMX	OX	CX	
Deviation %	0.009159	0.004298	0.053046		0.004638	0.007451	0.031776		0.009677	0.007402	0.051728		0.004639	0.005635	0.031951	
# Best	0.628571	0.4	0.971429		0.571429	0.571429	0.857143		0.542857	0.514286	0.942857		0.514286	0.571429	0.914286	
Score	0.657143	0.785714	0.057143		0.614286	0.614286	0.271429		0.685714	0.714286	0.1		0.714286	0.657143	0.128571	

GenGA

	RSM				IM				SM				CBM			
		scheduling operators				scheduling operators				scheduling operators				scheduling operators		
	1PX	2PX	UX	HX	1PX	2PX	UX	HX	1PX	2PX	UX	HX	1PX	2PX	UX	HX
Deviation %	0.021412	0.016044	0.026369	0.036287	0.030135	0.035595	0.039224	0.047724	0.026921	0.02978	0.026937	0.029345	0.037875	0.039218	0.028366	0.046093
# Best	0.428571	0.371429	0.285714	0.342857	0.257143	0.514286	0.228571	0.171429	0.314286	0.314286	0.314286	0.457143	0.371429	0.285714	0.228571	0.285714
Score	0.609524	0.638095	0.561905	0.466667	0.590476	0.6	0.533333	0.428571	0.561905	0.561905	0.571429	0.552381	0.571429	0.47619	0.619048	0.428571
		Routing operators				Routing operators				Routing operators				Routing operators		
	PMX	OX	CX		PMX	OX	CX		PMX	OX	CX		PMX	OX	CX	
Deviation %	0.032929	0.100133	0.27163		0.036973	0.129859	0.250533		0.027282	0.145778	0.252232		0.047022	0.086445	0.207628	
# Best	0.4	0.542857	0.885714		0.371429	0.742857	0.828571		0.285714	0.8	0.885714		0.371429	0.685714	0.914286	
Score	0.757143	0.642857	0.2		0.771429	0.542857	0.228571		0.828571	0.471429	0.228571		0.742857	0.557143	0.214286	

Table B.2: The normalised results for problem set A generated by SSGA* and SSGA

SSGA*

			1PX			2PX			UX			PMX			OX			CX			HX			CBUX		
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
	0.001	0.553	0.460	0.311	0.251	0.477	0.382	0.263	0.266	0.551	0.411	0.413	0.323	0.521	0.464	0.303	0.227	0.578	0.394	0.296	0.250	0.532	0.575	0.527	0.650	0.526	0.412	0.322	0.251	0.594	0.523	0.468	0.451
	0.010	0.243	0.153	0.106	0.101	0.239	0.146	0.078	0.085	0.219	0.169	0.110	0.082	0.256	0.148	0.108	0.068	0.276	0.142	0.117	0.095	0.283	0.290	0.282	0.671	0.174	0.181	0.113	0.097	0.300	0.235	0.233	0.218
RSM	0.100	0.248	0.133	0.066	0.065	0.228	0.133	0.096	0.078	0.282	0.126	0.060	0.048	0.318	0.141	0.111	0.095	0.231	0.144	0.098	0.063	0.252	0.239	0.177	0.690	0.177	0.107	0.090	0.072	0.242	0.203	0.183	0.167
	0.300	0.220	0.127	0.075	0.048	0.213	0.157	0.052	0.057	0.211	0.138	0.081	0.051	0.244	0.165	0.078	0.055	0.224	0.130	0.102	0.083	0.267	0.211	0.187	0.651	0.220	0.094	0.082	0.052	0.245	0.208	0.154	0.154
	0.001	0.667	0.582	0.405	0.361	0.615	0.581	0.449	0.335	0.639	0.546	0.401	0.390	0.733	0.498	0.404	0.296	0.616	0.578	0.437	0.338	0.653	0.660	0.583	0.666	0.675	0.585	0.471	0.327	0.620	0.670	0.584	0.560
IM	0.010	0.437	0.375	0.293	0.184	0.454	0.338	0.290	0.242	0.434	0.370	0.236	0.198	0.440	0.309	0.186	0.190	0.428	0.358	0.301	0.192	0.483	0.416	0.393	0.601	0.410	0.308	0.291	0.235	0.495	0.456	0.360	0.335
	0.100	0.331	0.234	0.160	0.118	0.407	0.232	0.157	0.139	0.334	0.252	0.177	0.147	0.320	0.195	0.132	0.156	0.352	0.227	0.205	0.140	0.337	0.313	0.319	0.661	0.289	0.223	0.201	0.158	0.335	0.301	0.262	0.274
	0.300	0.279	0.224	0.145	0.115	0.291	0.238	0.185	0.102	0.294	0.199	0.135	0.105	0.357	0.246	0.110	0.070	0.309	0.233	0.172	0.127	0.334	0.305	0.278	0.637	0.293	0.261	0.149	0.148	0.347	0.286	0.273	0.253
	0.001	0.550	0.459	0.361	0.256	0.543	0.460	0.321	0.244	0.564	0.426	0.418	0.284	0.566	0.389	0.340	0.222	0.565	0.465	0.313	0.273	0.593	0.553	0.569	0.654	0.544	0.411	0.342	0.292	0.574	0.531	0.470	0.428
	0.010	0.261	0.150	0.096	0.069	0.276	0.140	0.160	0.128	0.235	0.131	0.125	0.086	0.285	0.161	0.092	0.105	0.301	0.171	0.090	0.089	0.275	0.260	0.268	0.630	0.254	0.207	0.157	0.108	0.316	0.231	0.192	0.196
SM	0.100	0.210	0.117	0.055	0.037	0.171	0.079	0.070	0.029	0.185	0.062	0.057	0.038	0.273	0.116	0.052	0.039	0.212	0.120	0.051	0.041	0.270	0.210	0.185	0.622	0.191	0.094	0.041	0.069	0.271	0.182	0.153	0.138
	0.300	0.205	0.106	0.057	0.044	0.183	0.127	0.060	0.020	0.225	0.108	0.063	0.051	0.195	0.132	0.064	0.045	0.199	0.105	0.058	0.065	0.245	0.208	0.185	0.665	0.183	0.076	0.051	0.042	0.220	0.189	0.140	0.150
	0.001	0.616	0.511	0.421	0.298	0.573	0.534	0.390	0.295	0.725	0.575	0.413	0.389	0.614	0.499	0.346	0.278	0.576	0.521	0.373	0.337	0.626	0.562	0.621	0.660	0.667	0.532	0.463	0.386	0.605	0.526	0.504	0.471
CBM	0.010	0.447	0.423	0.326	0.240	0.418	0.368	0.299	0.261	0.408	0.375	0.322	0.249	0.460	0.328	0.288	0.222	0.420	0.387	0.319	0.252	0.415	0.425	0.355	0.609	0.413	0.382	0.347	0.273	0.418	0.438	0.344	0.327
	0.100	0.353	0.401	0.304	0.278	0.372	0.387	0.334	0.213	0.405	0.349	0.252	0.283	0.410	0.351	0.261	0.177	0.424	0.377	0.324	0.257	0.452	0.389	0.299	0.640	0.498	0.374	0.286	0.254	0.383	0.361	0.360	0.314
	0.300	0.411	0.363	0.312	0.221	0.399	0.383	0.371	0.279	0.435	0.390	0.336	0.242	0.375	0.296	0.265	0.210	0.410	0.373	0.264	0.291	0.390	0.447	0.313	0.615	0.391	0.442	0.326	0.288	0.374	0.432	0.368	0.331

SSGA

			1PX			2PX			UX			PMX			OX			CX			HX			CBUX		
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
	0.001	0.804	0.646	0.524	0.505	0.823	0.618	0.575	0.521	0.776	0.642	0.605	0.572	0.804	0.676	0.563	0.546	0.790	0.644	0.615	0.571	0.813	0.826	0.865	0.895	0.759	0.659	0.590	0.623	0.808	0.792	0.741	0.729
	0.010	0.487	0.321	0.268	0.223	0.497	0.334	0.256	0.211	0.496	0.327	0.263	0.239	0.550	0.404	0.245	0.219	0.561	0.349	0.307	0.222	0.594	0.606	0.631	0.905	0.461	0.350	0.255	0.285	0.649	0.540	0.531	0.503
RSM	0.100	0.469	0.260	0.141	0.183	0.446	0.202	0.081	0.153	0.455	0.219	0.151	0.155	0.515	0.284	0.230	0.122	0.505	0.256	0.176	0.121	0.567	0.558	0.510	0.898	0.424	0.219	0.189	0.172	0.519	0.480	0.468	0.411
	0.300	0.499	0.287	0.149	0.135	0.497	0.257	0.163	0.164	0.485	0.285	0.168	0.181	0.502	0.300	0.164	0.112	0.509	0.332	0.194	0.159	0.559	0.516	0.558	0.905	0.462	0.261	0.187	0.114	0.561	0.488	0.422	0.461
	0.001	0.939	0.801	0.701	0.593	0.899	0.764	0.684	0.627	0.883	0.875	0.736	0.677	0.869	0.826	0.668	0.647	0.897	0.814	0.753	0.655	0.936	0.885	0.891	0.908	0.895	0.806	0.728	0.698	0.919	0.858	0.832	0.803
	0.010	0.763	0.619	0.566	0.467	0.725	0.659	0.539	0.495	0.732	0.599	0.558	0.489	0.775	0.654	0.516	0.457	0.791	0.643	0.559	0.523	0.800	0.786	0.766	0.916	0.711	0.629	0.562	0.529	0.787	0.744	0.682	0.687
IM	0.100	0.643	0.541	0.411	0.355	0.644	0.524	0.375	0.392	0.593	0.505	0.406	0.400	0.645	0.544	0.369	0.364	0.695	0.528	0.443	0.362	0.710	0.713	0.683	0.896	0.615	0.514	0.451	0.411	0.715	0.695	0.595	0.614
	0.300	0.635	0.502	0.355	0.320	0.630	0.429	0.372	0.330	0.645	0.471	0.365	0.340	0.633	0.514	0.348	0.326	0.636	0.523	0.384	0.368	0.636	0.674	0.646	0.910	0.611	0.469	0.415	0.357	0.677	0.653	0.603	0.585
	0.001	0.837	0.777	0.639	0.553	0.839	0.716	0.663	0.600	0.841	0.728	0.659	0.638	0.835	0.697	0.625	0.620	0.820	0.770	0.692	0.659	0.850	0.853	0.891	0.899	0.855	0.745	0.729	0.622	0.873	0.825	0.777	0.767
	0.010	0.541	0.411	0.300	0.299	0.601	0.367	0.280	0.274	0.617	0.429	0.327	0.287	0.604	0.388	0.313	0.290	0.604	0.385	0.357	0.300	0.647	0.628	0.682	0.910	0.556	0.391	0.337	0.312	0.642	0.598	0.586	0.571
SM	0.100	0.452	0.191	0.144	0.106	0.425	0.220	0.165	0.087	0.440	0.215	0.139	0.151	0.479	0.236	0.154	0.068	0.459	0.223	0.126	0.132	0.558	0.548	0.529	0.900	0.445	0.200	0.133	0.101	0.542	0.459	0.463	0.443
	0.300	0.423	0.205	0.115	0.078	0.412	0.168	0.130	0.081	0.406	0.198	0.103	0.066	0.447	0.209	0.105	0.104	0.487	0.212	0.083	0.075	0.566	0.539	0.577	0.917	0.448	0.137	0.104	0.111	0.548	0.438	0.424	0.403
	0.001	0.762	0.739	0.614	0.566	0.735	0.743	0.651	0.578	0.786	0.732	0.663	0.663	0.754	0.737	0.623	0.603	0.768	0.729	0.600	0.639	0.764	0.733	0.794	0.927	0.753	0.749	0.696	0.615	0.775	0.747	0.751	0.728
	0.010	0.656	0.596	0.508	0.484	0.603	0.594	0.528	0.485	0.629	0.603	0.574	0.509	0.633	0.586	0.505	0.487	0.633	0.616	0.565	0.526	0.667	0.634	0.619	0.896	0.632	0.601	0.567	0.511	0.634	0.613	0.572	0.583
CBM	0.100	0.600	0.574	0.546	0.461	0.611	0.577	0.548	0.439	0.618	0.586	0.573	0.525	0.642	0.611	0.514	0.486	0.649	0.595	0.546	0.507	0.591	0.595	0.607	0.928	0.623	0.617	0.558	0.522	0.614	0.599	0.590	0.574
	0.300	0.610	0.587	0.525	0.485	0.599	0.596	0.550	0.460	0.602	0.593	0.555	0.530	0.622	0.608	0.521	0.455	0.608	0.592	0.552	0.537	0.620	0.606	0.599	0.900	0.603	0.589	0.529	0.548	0.619	0.599	0.599	0.543

Table B.3: The normalised results for problem set B generated by SSGA* and SSGA

SSGA*

		1PX				2PX				UX				PMX				OX				CX				HX				CBUX			
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
	0.001	0.330	0.322	0.219	0.257	0.328	0.311	0.214	0.219	0.329	0.288	0.233	0.222	0.367	0.300	0.228	0.177	0.365	0.263	0.275	0.220	0.345	0.273	0.418	0.336	0.341	0.304	0.245	0.206	0.400	0.355	0.338	0.412
	0.010	0.222	0.131	0.121	0.105	0.201	0.091	0.094	0.051	0.226	0.096	0.073	0.059	0.265	0.110	0.084	0.053	0.255	0.146	0.094	0.087	0.290	0.252	0.275	0.405	0.180	0.139	0.052	0.104	0.255	0.277	0.272	0.286
RSM	0.100	0.215	0.097	0.036	0.044	0.234	0.109	0.045	0.037	0.205	0.090	0.044	0.037	0.223	0.114	0.048	0.037	0.232	0.100	0.057	0.031	0.281	0.235	0.219	0.309	0.186	0.059	0.037	0.024	0.254	0.226	0.215	0.239
	0.300	0.246	0.105	0.046	0.036	0.215	0.085	0.064	0.037	0.228	0.092	0.034	0.028	0.236	0.124	0.045	0.039	0.232	0.110	0.018	0.032	0.269	0.256	0.217	0.460	0.193	0.082	0.047	0.022	0.226	0.242	0.217	0.194
	0.001	0.492	0.339	0.286	0.225	0.363	0.398	0.373	0.241	0.407	0.404	0.348	0.310	0.460	0.433	0.374	0.226	0.396	0.429	0.242	0.246	0.487	0.334	0.344	0.369	0.368	0.306	0.321	0.322	0.454	0.379	0.472	0.391
IM	0.010	0.314	0.333	0.211	0.213	0.324	0.263	0.181	0.282	0.327	0.291	0.248	0.136	0.335	0.257	0.219	0.169	0.331	0.295	0.222	0.314	0.435	0.353	0.283	0.420	0.350	0.254	0.174	0.277	0.349	0.363	0.352	0.302
	0.100	0.275	0.196	0.158	0.102	0.318	0.191	0.143	0.145	0.263	0.217	0.112	0.102	0.281	0.225	0.110	0.090	0.341	0.191	0.120	0.133	0.284	0.264	0.278	0.397	0.229	0.181	0.154	0.143	0.283	0.355	0.273	0.294
	0.300	0.265	0.183	0.136	0.122	0.263	0.160	0.135	0.125	0.258	0.178	0.179	0.144	0.283	0.236	0.123	0.116	0.294	0.222	0.126	0.078	0.344	0.254	0.263	0.400	0.270	0.177	0.097	0.119	0.297	0.263	0.246	0.245
	0.001	0.341	0.311	0.235	0.221	0.402	0.293	0.177	0.268	0.345	0.302	0.320	0.251	0.364	0.308	0.276	0.264	0.324	0.337	0.280	0.293	0.504	0.380	0.399	0.462	0.374	0.241	0.230	0.206	0.437	0.375	0.383	0.357
	0.010	0.248	0.177	0.086	0.066	0.203	0.132	0.086	0.058	0.203	0.139	0.094	0.054	0.215	0.148	0.092	0.100	0.284	0.140	0.106	0.097	0.269	0.249	0.345	0.340	0.239	0.137	0.135	0.066	0.269	0.281	0.264	0.232
SM	0.100	0.153	0.067	0.031	0.043	0.200	0.061	0.070	0.029	0.193	0.084	0.054	0.025	0.236	0.082	0.038	0.031	0.203	0.090	0.047	0.026	0.227	0.237	0.190	0.396	0.143	0.075	0.042	0.021	0.239	0.240	0.203	0.226
	0.300	0.204	0.075	0.038	0.022	0.184	0.079	0.026	0.029	0.195	0.092	0.031	0.026	0.203	0.095	0.033	0.022	0.192	0.087	0.022	0.041	0.249	0.224	0.198	0.346	0.183	0.064	0.035	0.026	0.233	0.224	0.201	0.195
	0.001	0.354	0.250	0.221	0.233	0.324	0.309	0.234	0.265	0.326	0.235	0.218	0.184	0.322	0.269	0.284	0.242	0.295	0.283	0.283	0.259	0.326	0.288	0.467	0.465	0.341	0.244	0.278	0.308	0.309	0.401	0.352	0.318
CBM	0.010	0.230	0.155	0.165	0.212	0.143	0.191	0.128	0.111	0.165	0.131	0.105	0.101	0.155	0.182	0.128	0.099	0.139	0.147	0.161	0.109	0.204	0.135	0.171	0.650	0.128	0.192	0.146	0.127	0.198	0.147	0.140	0.139
	0.100	0.121	0.158	0.091	0.077	0.145	0.134	0.099	0.091	0.115	0.127	0.117	0.076	0.138	0.141	0.083	0.063	0.124	0.138	0.133	0.093	0.119	0.143	0.110	0.376	0.113	0.160	0.196	0.122	0.147	0.114	0.141	0.150
	0.300	0.141	0.101	0.140	0.085	0.146	0.113	0.124	0.151	0.121	0.158	0.136	0.082	0.126	0.137	0.091	0.096	0.125	0.137	0.106	0.113	0.125	0.108	0.181	0.525	0.103	0.110	0.100	0.152	0.123	0.174	0.121	0.118

SSGA

		1PX				2PX				UX				PMX				OX				CX				HX				CBUX			
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
	0.001	0.830	0.741	0.636	0.620	0.856	0.747	0.735	0.651	0.825	0.734	0.725	0.701	0.859	0.761	0.670	0.659	0.860	0.774	0.723	0.667	0.885	0.833	0.874	0.897	0.835	0.795	0.724	0.726	0.881	0.868	0.883	0.837
	0.010	0.546	0.422	0.346	0.386	0.587	0.384	0.368	0.312	0.575	0.372	0.342	0.320	0.604	0.419	0.293	0.300	0.629	0.455	0.377	0.360	0.655	0.656	0.689	0.881	0.537	0.368	0.343	0.344	0.682	0.682	0.645	0.658
RSM	0.100	0.405	0.166	0.146	0.106	0.439	0.174	0.162	0.112	0.420	0.175	0.093	0.104	0.468	0.210	0.111	0.083	0.444	0.183	0.122	0.102	0.548	0.566	0.533	0.903	0.335	0.161	0.116	0.103	0.508	0.479	0.456	0.443
	0.300	0.384	0.131	0.094	0.064	0.431	0.178	0.092	0.090	0.403	0.141	0.067	0.057	0.439	0.212	0.081	0.058	0.445	0.176	0.085	0.060	0.526	0.549	0.530	0.898	0.357	0.113	0.119	0.058	0.519	0.471	0.454	0.410
	0.001	0.953	0.871	0.780	0.710	0.924	0.852	0.794	0.741	0.911	0.865	0.817	0.787	0.922	0.889	0.789	0.760	0.939	0.886	0.816	0.766	0.949	0.916	0.889	0.901	0.949	0.858	0.809	0.801	0.943	0.914	0.947	0.913
	0.010	0.843	0.752	0.662	0.590	0.839	0.733	0.622	0.629	0.851	0.669	0.653	0.619	0.843	0.693	0.617	0.602	0.833	0.765	0.642	0.649	0.847	0.840	0.809	0.902	0.847	0.714	0.657	0.643	0.856	0.845	0.838	0.847
IM	0.100	0.754	0.634	0.452	0.467	0.727	0.603	0.533	0.482	0.719	0.595	0.521	0.516	0.754	0.579	0.421	0.439	0.749	0.626	0.535	0.465	0.774	0.757	0.746	0.891	0.700	0.540	0.557	0.484	0.789	0.758	0.761	0.711
	0.300	0.725	0.590	0.450	0.403	0.722	0.545	0.419	0.393	0.671	0.525	0.467	0.429	0.626	0.511	0.377	0.324	0.696	0.581	0.480	0.434	0.739	0.712	0.723	0.896	0.682	0.529	0.438	0.447	0.731	0.725	0.682	0.682
	0.001	0.886	0.803	0.686	0.659	0.859	0.808	0.716	0.685	0.884	0.826	0.793	0.731	0.894	0.813	0.717	0.683	0.891	0.811	0.780	0.698	0.880	0.865	0.888	0.897	0.911	0.816	0.769	0.750	0.931	0.890	0.878	0.901
	0.010	0.626	0.474	0.415	0.405	0.653	0.485	0.376	0.373	0.590	0.480	0.415	0.377	0.650	0.458	0.408	0.353	0.641	0.513	0.416	0.382	0.710	0.673	0.764	0.889	0.601	0.462	0.435	0.417	0.711	0.733	0.636	0.663
SM	0.100	0.444	0.195	0.152	0.131	0.414	0.210	0.163	0.162	0.426	0.175	0.149	0.145	0.435	0.224	0.159	0.120	0.525	0.207	0.171	0.117	0.572	0.527	0.575	0.905	0.388	0.193	0.155	0.148	0.527	0.517	0.486	0.452
	0.300	0.367	0.134	0.094	0.085	0.390	0.140	0.070	0.076	0.396	0.161	0.110	0.066	0.454	0.209	0.084	0.060	0.440	0.145	0.058	0.089	0.545	0.536	0.519	0.905	0.390	0.132	0.092	0.072	0.535	0.467	0.432	0.412
	0.001	0.796	0.751	0.692	0.671	0.813	0.779	0.661	0.647	0.796	0.760	0.740	0.735	0.751	0.740	0.712	0.665	0.803	0.773	0.744	0.694	0.785	0.798	0.815	0.906	0.810	0.775	0.730	0.693	0.797	0.781	0.795	0.762
	0.010	0.484	0.448	0.421	0.385	0.478	0.448	0.397	0.392	0.462	0.463	0.409	0.421	0.468	0.410	0.362	0.379	0.474	0.462	0.420	0.426	0.452	0.484	0.531	0.883	0.475	0.484	0.408	0.440	0.467	0.448	0.476	0.479
CBM	0.100	0.355	0.325	0.305	0.280	0.353	0.334	0.319	0.299	0.347	0.328	0.326	0.321	0.380	0.350	0.277	0.251	0.357	0.336	0.320	0.321	0.361	0.331	0.360	0.897	0.354	0.340	0.322	0.305	0.327	0.365	0.323	0.343
	0.300	0.357	0.333	0.315	0.287	0.363	0.331	0.308	0.284	0.357	0.340	0.304	0.327	0.307	0.341	0.307	0.283	0.327	0.373	0.323	0.320	0.337	0.339	0.334	0.900	0.320	0.321	0.359	0.309	0.363	0.343	0.327	0.338

Table B.4: The normalised results for problem set C generated by SSGA* and SSGA

SSGA*

			1PX			2PX			UX			PMX			OX			CX			HX			CBUX		
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
	0.001	0.848	0.811	0.700	0.685	0.853	0.779	0.641	0.601	0.862	0.771	0.650	0.582	0.849	0.736	0.596	0.606	0.834	0.816	0.686	0.649	0.826	0.744	0.677	0.704	0.878	0.769	0.632	0.597	0.902	0.835	0.747	0.767
RSM	0.010	0.676	0.623	0.568	0.541	0.671	0.613	0.476	0.484	0.683	0.647	0.446	0.404	0.651	0.643	0.487	0.430	0.677	0.621	0.544	0.468	0.636	0.531	0.559	0.713	0.651	0.560	0.473	0.368	0.708	0.647	0.625	0.585
	0.100	0.404	0.442	0.320	0.329	0.422	0.341	0.318	0.225	0.428	0.373	0.316	0.216	0.484	0.393	0.304	0.195	0.503	0.431	0.354	0.261	0.420	0.342	0.372	0.721	0.466	0.362	0.309	0.178	0.445	0.409	0.416	0.404
	0.300	0.362	0.309	0.283	0.244	0.341	0.298	0.251	0.202	0.319	0.269	0.177	0.192	0.383	0.290	0.213	0.148	0.359	0.334	0.256	0.181	0.297	0.266	0.278	0.689	0.353	0.296	0.262	0.147	0.350	0.346	0.366	0.321
	0.001	0.884	0.859	0.785	0.715	0.923	0.795	0.720	0.637	0.912	0.816	0.699	0.607	0.904	0.824	0.667	0.617	0.884	0.843	0.723	0.647	0.836	0.734	0.735	0.723	0.903	0.841	0.677	0.577	0.884	0.876	0.870	0.859
IM	0.010	0.841	0.734	0.687	0.649	0.782	0.733	0.620	0.565	0.808	0.732	0.604	0.548	0.766	0.738	0.556	0.528	0.753	0.782	0.615	0.588	0.764	0.674	0.652	0.724	0.792	0.687	0.607	0.530	0.793	0.770	0.750	0.724
	0.100	0.683	0.641	0.573	0.510	0.690	0.610	0.522	0.436	0.700	0.584	0.559	0.418	0.644	0.614	0.517	0.443	0.699	0.642	0.516	0.511	0.644	0.577	0.546	0.702	0.707	0.612	0.495	0.425	0.686	0.707	0.633	0.615
	0.300	0.578	0.582	0.547	0.515	0.650	0.568	0.507	0.435	0.630	0.590	0.500	0.415	0.644	0.573	0.455	0.414	0.627	0.594	0.531	0.475	0.572	0.519	0.505	0.684	0.608	0.562	0.470	0.441	0.608	0.604	0.634	0.561
	0.001	0.841	0.755	0.695	0.648	0.829	0.795	0.634	0.609	0.826	0.781	0.618	0.596	0.829	0.796	0.628	0.515	0.831	0.797	0.695	0.649	0.760	0.711	0.672	0.695	0.820	0.780	0.635	0.571	0.866	0.817	0.771	0.752
	0.010	0.641	0.600	0.553	0.487	0.623	0.562	0.480	0.443	0.586	0.559	0.490	0.465	0.617	0.587	0.458	0.373	0.647	0.570	0.514	0.421	0.586	0.533	0.554	0.708	0.645	0.579	0.447	0.407	0.674	0.626	0.622	0.580
SM	0.100	0.386	0.351	0.310	0.281	0.441	0.362	0.307	0.275	0.425	0.359	0.321	0.221	0.426	0.372	0.234	0.140	0.428	0.388	0.248	0.240	0.423	0.358	0.362	0.706	0.411	0.339	0.300	0.172	0.421	0.421	0.395	0.343
	0.300	0.313	0.286	0.264	0.243	0.320	0.283	0.263	0.147	0.369	0.247	0.249	0.152	0.339	0.295	0.201	0.077	0.326	0.262	0.236	0.201	0.330	0.268	0.257	0.717	0.352	0.279	0.219	0.169	0.378	0.346	0.303	0.278
	0.001	0.877	0.837	0.699	0.700	0.890	0.804	0.678	0.670	0.883	0.812	0.681	0.599	0.893	0.800	0.654	0.546	0.893	0.863	0.703	0.664	0.842	0.732	0.714	0.696	0.870	0.761	0.701	0.593	0.911	0.879	0.842	0.763
CBM	0.010	0.853	0.802	0.728	0.678	0.844	0.812	0.699	0.592	0.859	0.803	0.636	0.625	0.910	0.799	0.633	0.532	0.880	0.840	0.711	0.682	0.826	0.722	0.661	0.699	0.845	0.812	0.658	0.570	0.874	0.858	0.807	0.755
	0.100	0.860	0.797	0.711	0.672	0.875	0.810	0.655	0.608	0.847	0.810	0.692	0.590	0.860	0.801	0.625	0.538	0.860	0.795	0.697	0.636	0.809	0.733	0.669	0.694	0.849	0.803	0.681	0.550	0.847	0.834	0.779	0.764
	0.300	0.851	0.831	0.725	0.649	0.855	0.807	0.672	0.637	0.825	0.815	0.660	0.578	0.898	0.802	0.624	0.565	0.856	0.836	0.731	0.652	0.797	0.749	0.680	0.693	0.853	0.822	0.693	0.566	0.843	0.835	0.763	0.800

SSGA

			1PX			2PX			UX			PMX			OX			CX			HX			CBUX		
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
	0.001	0.643	0.556	0.470	0.269	0.746	0.471	0.343	0.251	0.639	0.539	0.440	0.419	0.748	0.416	0.313	0.165	0.563	0.578	0.304	0.175	0.632	0.746	0.736	0.664	0.628	0.544	0.437	0.331	0.597	0.600	0.633	0.541
RSM	0.010	0.524	0.481	0.364	0.305	0.645	0.467	0.352	0.262	0.644	0.495	0.294	0.239	0.570	0.453	0.239	0.131	0.588	0.508	0.294	0.151	0.657	0.578	0.710	0.712	0.565	0.459	0.320	0.241	0.555	0.430	0.511	0.395
	0.100	0.570	0.479	0.354	0.268	0.558	0.476	0.344	0.206	0.526	0.450	0.244	0.236	0.539	0.415	0.238	0.180	0.480	0.466	0.215	0.193	0.512	0.521	0.619	0.751	0.557	0.464	0.234	0.239	0.498	0.501	0.405	0.386
	0.300	0.482	0.501	0.366	0.299	0.517	0.409	0.253	0.225	0.470	0.447	0.260	0.184	0.496	0.429	0.259	0.122	0.580	0.438	0.235	0.157	0.545	0.431	0.562	0.710	0.489	0.444	0.316	0.261	0.571	0.506	0.344	0.461
	0.001	0.660	0.519	0.443	0.357	0.606	0.559	0.377	0.330	0.662	0.578	0.467	0.418	0.641	0.431	0.311	0.236	0.731	0.463	0.346	0.192	0.757	0.713	0.744	0.722	0.663	0.575	0.425	0.383	0.668	0.648	0.597	0.560
IM	0.010	0.620	0.536	0.415	0.421	0.680	0.522	0.436	0.277	0.685	0.622	0.394	0.418	0.562	0.476	0.285	0.159	0.641	0.484	0.283	0.191	0.710	0.717	0.736	0.801	0.575	0.601	0.428	0.413	0.722	0.594	0.495	0.383
	0.100	0.617	0.523	0.434	0.373	0.637	0.593	0.444	0.261	0.609	0.455	0.458	0.343	0.618	0.428	0.351	0.185	0.573	0.476	0.313	0.209	0.635	0.632	0.728	0.684	0.660	0.518	0.443	0.474	0.623	0.614	0.539	0.594
	0.300	0.633	0.502	0.455	0.399	0.632	0.624	0.454	0.371	0.680	0.622	0.409	0.402	0.740	0.548	0.287	0.166	0.615	0.615	0.319	0.218	0.656	0.603	0.586	0.724	0.654	0.600	0.431	0.386	0.618	0.544	0.516	0.473
	0.001	0.677	0.520	0.442	0.381	0.664	0.553	0.429	0.347	0.666	0.504	0.448	0.425	0.701	0.512	0.318	0.213	0.687	0.466	0.303	0.222	0.679	0.691	0.721	0.715	0.685	0.589	0.443	0.479	0.749	0.668	0.484	0.548
	0.010	0.660	0.559	0.424	0.358	0.672	0.519	0.403	0.234	0.615	0.496	0.397	0.296	0.552	0.450	0.305	0.172	0.675	0.445	0.322	0.187	0.678	0.653	0.734	0.697	0.685	0.552	0.378	0.334	0.602	0.671	0.522	0.447
SM	0.100	0.508	0.484	0.379	0.224	0.670	0.511	0.359	0.247	0.598	0.456	0.301	0.199	0.570	0.404	0.167	0.145	0.532	0.462	0.318	0.155	0.574	0.599	0.682	0.665	0.605	0.517	0.272	0.243	0.545	0.513	0.490	0.408
	0.300	0.606	0.517	0.370	0.343	0.622	0.536	0.301	0.269	0.520	0.475	0.308	0.245	0.628	0.445	0.236	0.157	0.556	0.445	0.276	0.197	0.607	0.588	0.633	0.788	0.605	0.534	0.307	0.258	0.576	0.553	0.369	0.439
	0.001	0.664	0.607	0.427	0.443	0.650	0.607	0.395	0.419	0.661	0.555	0.494	0.374	0.586	0.500	0.298	0.153	0.618	0.507	0.277	0.204	0.750	0.671	0.674	0.627	0.618	0.541	0.407	0.332	0.692	0.613	0.512	0.457
	0.010	0.600	0.530	0.365	0.330	0.650	0.571	0.388	0.238	0.742	0.497	0.384	0.326	0.601	0.474	0.231	0.178	0.726	0.585	0.279	0.197	0.660	0.598	0.651	0.789	0.678	0.489	0.364	0.316	0.600	0.618	0.483	0.445
CBM	0.100	0.549	0.514	0.354	0.307	0.609	0.504	0.275	0.380	0.551	0.529	0.342	0.259	0.579	0.390	0.248	0.222	0.591	0.483	0.243	0.153	0.609	0.535	0.716	0.731	0.550	0.521	0.334	0.284	0.602	0.551	0.416	0.422
	0.300	0.483	0.576	0.323	0.328	0.494	0.571	0.403	0.315	0.648	0.444	0.363	0.199	0.539	0.455	0.261	0.199	0.630	0.465	0.284	0.179	0.545	0.560	0.756	0.680	0.624	0.463	0.329	0.306	0.459	0.578	0.479	0.444

Table B.5: The normalised results for problem set D generated by SSGA* and SSGA

		SSGA*																													
		1PX			2PX			UX			PMX			OX			CX			HX			CBUX								
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
RSM	0.001	0.507	0.519	0.499	0.512	0.512	0.504	0.496	0.521	0.526	0.521	0.515	0.526	0.520	0.511	0.516	0.520	0.516	0.524	0.516	0.507	0.525	0.514	0.511	0.535	0.517	0.508	0.515	0.530	0.522	0.534	0.511	0.521
	0.010	0.496	0.496	0.469	0.439	0.488	0.493	0.488	0.495	0.512	0.516	0.512	0.489	0.488	0.500	0.503	0.478	0.484	0.503	0.478	0.496	0.526	0.513	0.519	0.529	0.500	0.497	0.484	0.489	0.496	0.499	0.521	0.490
	0.100	0.385	0.355	0.334	0.340	0.367	0.380	0.354	0.343	0.391	0.366	0.361	0.360	0.392	0.367	0.340	0.343	0.407	0.367	0.363	0.365	0.387	0.403	0.452	0.530	0.382	0.345	0.365	0.350	0.384	0.379	0.386	0.393
	0.300	0.257	0.244	0.178	0.197	0.258	0.231	0.214	0.229	0.228	0.236	0.204	0.216	0.232	0.216	0.213	0.197	0.240	0.226	0.210	0.220	0.232	0.262	0.339	0.538	0.231	0.229	0.200	0.206	0.224	0.238	0.247	0.235
	0.001	0.516	0.512	0.503	0.499	0.526	0.518	0.521	0.504	0.509	0.527	0.514	0.503	0.531	0.513	0.512	0.522	0.525	0.536	0.522	0.517	0.533	0.557	0.519	0.522	0.525	0.520	0.512	0.517	0.536	0.533	0.520	0.538
IM	0.010	0.529	0.515	0.504	0.510	0.517	0.521	0.509	0.521	0.526	0.510	0.542	0.512	0.540	0.542	0.522	0.534	0.520	0.524	0.523	0.513	0.523	0.532	0.510	0.524	0.519	0.512	0.516	0.500	0.535	0.509	0.528	0.530
	0.100	0.518	0.512	0.502	0.482	0.502	0.527	0.521	0.510	0.517	0.505	0.498	0.510	0.516	0.502	0.505	0.506	0.503	0.495	0.514	0.492	0.532	0.510	0.524	0.517	0.519	0.527	0.525	0.514	0.492	0.502	0.531	0.509
	0.300	0.502	0.513	0.478	0.496	0.498	0.496	0.503	0.493	0.512	0.502	0.493	0.497	0.521	0.501	0.492	0.481	0.505	0.506	0.499	0.492	0.502	0.493	0.519	0.520	0.515	0.492	0.497	0.489	0.499	0.503	0.503	0.498
	0.001	0.509	0.516	0.524	0.497	0.526	0.517	0.517	0.500	0.519	0.505	0.523	0.510	0.514	0.529	0.520	0.515	0.511	0.524	0.526	0.520	0.517	0.524	0.525	0.519	0.528	0.516	0.518	0.529	0.511	0.530	0.509	0.522
	0.010	0.519	0.489	0.477	0.476	0.503	0.502	0.478	0.485	0.509	0.489	0.491	0.481	0.480	0.514	0.490	0.497	0.503	0.486	0.492	0.458	0.502	0.489	0.510	0.521	0.513	0.493	0.474	0.483	0.487	0.502	0.496	0.517
SM	0.100	0.408	0.364	0.353	0.352	0.413	0.363	0.348	0.329	0.405	0.367	0.331	0.331	0.406	0.361	0.347	0.340	0.407	0.387	0.327	0.332	0.400	0.407	0.467	0.536	0.401	0.358	0.330	0.331	0.399	0.393	0.385	0.411
	0.300	0.257	0.227	0.186	0.186	0.272	0.233	0.209	0.198	0.269	0.230	0.188	0.181	0.266	0.258	0.219	0.191	0.262	0.253	0.185	0.173	0.260	0.293	0.366	0.515	0.275	0.231	0.189	0.179	0.249	0.242	0.255	0.263
	0.001	0.524	0.505	0.511	0.506	0.511	0.502	0.500	0.520	0.520	0.539	0.516	0.520	0.517	0.525	0.517	0.532	0.929	0.934	0.935	0.243	0.923	0.923	0.938	0.931	0.915	0.928	0.912	0.907	0.889	0.919	0.914	0.933
CBM	0.010	0.511	0.500	0.489	0.485	0.487	0.497	0.503	0.488	0.506	0.512	0.506	0.496	0.495	0.492	0.487	0.509	0.919	0.919	0.912	0.130	0.909	0.926	0.954	0.912	0.904	0.920	0.929	0.903	0.896	0.936	0.925	0.935
	0.100	0.391	0.410	0.397	0.378	0.413	0.411	0.416	0.406	0.412	0.395	0.391	0.409	0.406	0.382	0.369	0.355	0.913	0.915	0.923	0.805	0.928	0.948	0.927	0.928	0.897	0.931	0.895	0.959	0.910	0.926	0.919	0.919
	0.300	0.280	0.275	0.271	0.287	0.291	0.292	0.272	0.293	0.304	0.273	0.265	0.275	0.289	0.270	0.281	0.262	0.945	0.958	0.911	0.913	0.921	0.950	0.928	0.920	0.924	0.913	0.907	0.948	0.940	0.924	0.927	0.931

		SSGA																													
		1PX			2PX			UX			PMX			OX			CX			HX			CBUX								
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
RSM	0.001	0.215	0.175	0.247	0.163	0.154	0.159	0.416	0.306	0.123	0.241	0.228	0.302	0.301	0.244	0.193	0.208	0.402	0.224	0.270	0.247	0.204	0.180	0.232	0.283	0.197	0.119	0.204	0.306	0.245	0.223	0.262	0.287
	0.010	0.230	0.250	0.177	0.315	0.169	0.223	0.197	0.320	0.283	0.266	0.274	0.246	0.176	0.193	0.288	0.284	0.374	0.203	0.260	0.237	0.239	0.207	0.240	0.247	0.234	0.284	0.229	0.273	0.217	0.208	0.208	0.308
	0.100	0.205	0.198	0.228	0.200	0.207	0.220	0.145	0.291	0.258	0.174	0.323	0.299	0.245	0.226	0.247	0.320	0.266	0.308	0.213	0.199	0.214	0.190	0.246	0.342	0.219	0.274	0.208	0.308	0.239	0.258	0.252	0.270
	0.300	0.247	0.240	0.240	0.171	0.190	0.251	0.228	0.241	0.144	0.256	0.193	0.224	0.177	0.272	0.298	0.214	0.200	0.151	0.195	0.248	0.293	0.261	0.279	0.278	0.179	0.250	0.212	0.267	0.163	0.314	0.300	0.227
	0.001	0.285	0.252	0.222	0.224	0.118	0.201	0.253	0.158	0.177	0.171	0.323	0.266	0.269	0.271	0.301	0.236	0.162	0.199	0.252	0.261	0.247	0.295	0.185	0.330	0.239	0.222	0.264	0.372	0.323	0.212	0.205	0.166
IM	0.010	0.222	0.290	0.303	0.210	0.393	0.177	0.293	0.341	0.249	0.218	0.198	0.323	0.298	0.194	0.342	0.387	0.258	0.215	0.285	0.310	0.199	0.235	0.227	0.204	0.166	0.228	0.251	0.260	0.240	0.215	0.222	0.264
	0.100	0.168	0.260	0.255	0.263	0.161	0.159	0.229	0.220	0.186	0.197	0.131	0.230	0.252	0.138	0.197	0.415	0.247	0.226	0.136	0.325	0.140	0.270	0.256	0.256	0.200	0.218	0.301	0.425	0.283	0.230	0.191	0.343
	0.300	0.293	0.298	0.158	0.209	0.192	0.348	0.249	0.424	0.260	0.252	0.211	0.214	0.256	0.186	0.288	0.212	0.164	0.227	0.180	0.318	0.198	0.241	0.299	0.283	0.214	0.242	0.282	0.214	0.229	0.237	0.209	0.233
	0.001	0.263	0.119	0.200	0.209	0.131	0.186	0.220	0.234	0.149	0.224	0.310	0.277	0.201	0.213	0.265	0.242	0.203	0.216	0.239	0.158	0.140	0.248	0.242	0.290	0.212	0.204	0.190	0.248	0.195	0.211	0.152	0.205
	0.010	0.262	0.229	0.287	0.309	0.177	0.221	0.207	0.226	0.181	0.221	0.200	0.312	0.143	0.333	0.220	0.278	0.169	0.225	0.225	0.266	0.164	0.217	0.205	0.217	0.166	0.163	0.286	0.307	0.212	0.346	0.348	0.315
SM	0.100	0.215	0.172	0.272	0.252	0.165	0.142	0.266	0.292	0.164	0.270	0.294	0.176	0.238	0.197	0.307	0.202	0.170	0.277	0.151	0.265	0.262	0.168	0.188	0.276	0.152	0.220	0.308	0.256	0.197	0.188	0.242	0.202
	0.300	0.228	0.253	0.215	0.213	0.240	0.224	0.229	0.155	0.172	0.162	0.229	0.278	0.129	0.217	0.188	0.255	0.125	0.256	0.184	0.210	0.188	0.159	0.196	0.292	0.280	0.310	0.350	0.287	0.166	0.249	0.193	0.196
	0.001	0.279	0.245	0.376	0.250	0.213	0.202	0.279	0.417	0.258	0.267	0.265	0.219	0.148	0.162	0.276	0.256	0.207	0.261	0.206	0.215	0.167	0.202	0.285	0.335	0.152	0.157	0.214	0.215	0.342	0.325	0.322	0.385
CBM	0.010	0.274	0.154	0.275	0.260	0.274	0.230	0.239	0.362	0.179	0.252	0.353	0.249	0.240	0.256	0.274	0.230	0.216	0.201	0.356	0.249	0.190	0.184	0.242	0.243	0.257	0.223	0.183	0.221	0.366	0.338	0.344	0.477
	0.100	0.151	0.229	0.197	0.317	0.169	0.219	0.243	0.341	0.235	0.181	0.286	0.215	0.208	0.175	0.204	0.219	0.266	0.230	0.247	0.179	0.165	0.181	0.328	0.274	0.257	0.223	0.160	0.228	0.427	0.319	0.391	0.379
	0.300	0.355	0.131	0.212	0.142	0.160	0.181	0.082	0.159	0.200	0.188	0.172	0.240	0.255	0.260	0.182	0.164	0.221	0.242	0.223	0.170	0.203	0.144	0.180	0.265	0.130	0.112	0.229	0.194	0.258	0.289	0.340	0.316

Table B.6: The normalised results for problem set E generated by SSGA* and SSGA

SSGA*

			1PX			2PX			UX			PMX			OX			CX			HX			CBUX		
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
	0.001	0.918	0.906	0.822	0.781	0.896	0.879	0.831	0.771	0.921	0.916	0.932	0.916	0.929	0.924	0.894	0.873	0.927	0.914	0.887	0.826	0.903	0.939	0.930	0.931	0.921	0.901	0.927	0.905	0.915	0.912	0.931	0.890
	0.010	0.876	0.854	0.759	0.760	0.845	0.836	0.776	0.740	0.872	0.837	0.881	0.864	0.901	0.858	0.824	0.787	0.875	0.858	0.840	0.791	0.902	0.868	0.907	0.911	0.868	0.846	0.856	0.892	0.884	0.866	0.879	0.868
RSM	0.100	0.628	0.590	0.503	0.501	0.610	0.563	0.551	0.499	0.629	0.600	0.565	0.566	0.643	0.598	0.517	0.483	0.633	0.587	0.551	0.528	0.658	0.636	0.787	0.927	0.608	0.562	0.522	0.600	0.600	0.603	0.583	0.626
	0.300	0.314	0.302	0.219	0.194	0.328	0.286	0.196	0.209	0.319	0.310	0.249	0.286	0.387	0.291	0.236	0.246	0.325	0.306	0.276	0.261	0.311	0.365	0.548	0.921	0.285	0.228	0.250	0.241	0.267	0.288	0.266	0.334
	0.001	0.920	0.875	0.846	0.795	0.908	0.912	0.848	0.784	0.922	0.898	0.886	0.922	0.923	0.931	0.892	0.856	0.929	0.938	0.898	0.860	0.913	0.928	0.915	0.912	0.877	0.905	0.906	0.939	0.901	0.904	0.930	0.925
	0.010	0.939	0.885	0.829	0.785	0.899	0.891	0.837	0.757	0.858	0.891	0.893	0.889	0.869	0.885	0.875	0.837	0.920	0.885	0.847	0.840	0.916	0.899	0.901	0.942	0.918	0.903	0.864	0.881	0.882	0.882	0.905	0.895
IM	0.100	0.847	0.798	0.752	0.729	0.848	0.818	0.771	0.709	0.846	0.836	0.827	0.845	0.819	0.809	0.793	0.731	0.818	0.808	0.792	0.769	0.822	0.879	0.870	0.909	0.833	0.807	0.816	0.813	0.809	0.807	0.849	0.836
	0.300	0.727	0.720	0.689	0.641	0.735	0.715	0.712	0.683	0.763	0.765	0.756	0.742	0.714	0.727	0.669	0.655	0.729	0.753	0.744	0.698	0.741	0.760	0.811	0.906	0.735	0.730	0.744	0.751	0.714	0.727	0.733	0.733
	0.001	0.900	0.872	0.838	0.742	0.918	0.898	0.833	0.784	0.933	0.912	0.900	0.933	0.909	0.896	0.851	0.859	0.900	0.901	0.868	0.819	0.916	0.918	0.902	0.908	0.919	0.904	0.922	0.925	0.933	0.907	0.891	0.904
	0.010	0.888	0.831	0.786	0.712	0.846	0.824	0.776	0.731	0.832	0.835	0.813	0.820	0.863	0.821	0.789	0.759	0.870	0.825	0.801	0.787	0.874	0.883	0.901	0.926	0.857	0.821	0.814	0.806	0.837	0.866	0.853	0.862
SM	0.100	0.541	0.496	0.419	0.369	0.555	0.515	0.416	0.351	0.570	0.501	0.446	0.447	0.513	0.460	0.385	0.373	0.518	0.483	0.435	0.383	0.534	0.540	0.711	0.908	0.508	0.393	0.390	0.363	0.503	0.464	0.464	0.487
	0.300	0.185	0.179	0.081	0.077	0.212	0.180	0.096	0.065	0.214	0.201	0.116	0.112	0.205	0.170	0.080	0.087	0.211	0.168	0.122	0.100	0.215	0.224	0.446	0.897	0.130	0.108	0.069	0.075	0.146	0.150	0.143	0.148
	0.001	0.895	0.894	0.811	0.798	0.921	0.908	0.854	0.753	0.895	0.910	0.921	0.910	0.912	0.902	0.861	0.868	0.905	0.914	0.872	0.850	0.911	0.922	0.921	0.940	0.917	0.889	0.929	0.876	0.912	0.903	0.909	0.920
CBM	0.010	0.881	0.856	0.811	0.754	0.860	0.858	0.804	0.761	0.864	0.869	0.863	0.858	0.857	0.889	0.806	0.803	0.877	0.900	0.811	0.803	0.883	0.883	0.919	0.909	0.857	0.875	0.860	0.851	0.886	0.882	0.905	0.889
	0.100	0.631	0.619	0.552	0.556	0.630	0.604	0.578	0.566	0.638	0.640	0.669	0.641	0.644	0.615	0.564	0.584	0.652	0.641	0.624	0.583	0.662	0.656	0.753	0.923	0.655	0.646	0.626	0.643	0.619	0.616	0.643	0.695
	0.300	0.351	0.327	0.351	0.293	0.366	0.335	0.329	0.305	0.395	0.366	0.355	0.363	0.384	0.366	0.343	0.290	0.402	0.385	0.363	0.362	0.386	0.399	0.573	0.924	0.418	0.390	0.366	0.366	0.273	0.317	0.364	0.386

SSGA

			1PX			2PX			UX			PMX			OX			CX			HX			CBUX		
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
	0.001	0.482	0.464	0.436	0.420	0.467	0.502	0.452	0.457	0.496	0.467	0.456	0.476	0.476	0.505	0.480	0.452	0.486	0.466	0.443	0.457	0.464	0.474	0.480	0.472	0.488	0.484	0.482	0.478	0.456	0.486	0.459	0.430
	0.010	0.449	0.471	0.380	0.424	0.484	0.465	0.429	0.454	0.435	0.488	0.483	0.490	0.486	0.465	0.464	0.447	0.462	0.461	0.476	0.449	0.438	0.485	0.459	0.459	0.447	0.443	0.456	0.456	0.501	0.495	0.462	0.461
RSM	0.100	0.407	0.391	0.321	0.290	0.357	0.400	0.345	0.310	0.401	0.379	0.346	0.331	0.420	0.376	0.301	0.338	0.392	0.376	0.326	0.358	0.397	0.447	0.410	0.496	0.414	0.355	0.371	0.341	0.425	0.414	0.385	0.400
	0.300	0.318	0.325	0.202	0.209	0.297	0.337	0.253	0.227	0.305	0.269	0.234	0.226	0.300	0.298	0.250	0.200	0.290	0.296	0.271	0.294	0.275	0.277	0.394	0.465	0.328	0.286	0.244	0.234	0.292	0.277	0.283	0.305
	0.001	0.488	0.461	0.443	0.450	0.496	0.491	0.463	0.482	0.496	0.475	0.486	0.483	0.449	0.489	0.501	0.447	0.470	0.482	0.426	0.434	0.449	0.504	0.471	0.465	0.466	0.504	0.486	0.453	0.506	0.486	0.457	0.467
	0.010	0.482	0.446	0.433	0.432	0.482	0.475	0.446	0.423	0.450	0.476	0.464	0.463	0.459	0.459	0.467	0.423	0.467	0.464	0.480	0.461	0.508	0.510	0.453	0.469	0.436	0.499	0.447	0.452	0.488	0.484	0.464	0.481
IM	0.100	0.473	0.463	0.440	0.427	0.500	0.469	0.478	0.452	0.505	0.479	0.462	0.474	0.447	0.473	0.442	0.448	0.506	0.489	0.467	0.441	0.467	0.455	0.489	0.496	0.455	0.476	0.436	0.469	0.471	0.485	0.487	0.472
	0.300	0.497	0.476	0.430	0.433	0.464	0.484	0.421	0.419	0.458	0.472	0.452	0.467	0.490	0.481	0.445	0.501	0.442	0.453	0.443	0.433	0.480	0.437	0.457	0.473	0.437	0.533	0.462	0.445	0.465	0.442	0.485	0.467
	0.001	0.481	0.484	0.426	0.481	0.475	0.486	0.453	0.403	0.474	0.454	0.497	0.448	0.467	0.506	0.472	0.444	0.448	0.458	0.479	0.462	0.482	0.482	0.482	0.439	0.469	0.473	0.477	0.480	0.441	0.471	0.471	0.463
	0.010	0.511	0.430	0.434	0.440	0.444	0.465	0.426	0.424	0.509	0.464	0.427	0.418	0.485	0.446	0.426	0.442	0.469	0.462	0.438	0.475	0.455	0.455	0.439	0.514	0.472	0.468	0.463	0.466	0.420	0.444	0.454	0.473	0.469
SM	0.100	0.418	0.384	0.338	0.335	0.404	0.398	0.356	0.574	0.414	0.342	0.315	0.354	0.423	0.377	0.340	0.329	0.384	0.403	0.346	0.388	0.414	0.404	0.434	0.455	0.368	0.389	0.357	0.410	0.405	0.409	0.407	0.371
	0.300	0.324	0.261	0.250	0.214	0.332	0.313	0.276	0.198	0.277	0.257	0.164	0.203	0.291	0.299	0.235	0.216	0.289	0.274	0.231	0.205	0.294	0.316	0.428	0.464	0.334	0.253	0.218	0.237	0.314	0.334	0.298	0.315
	0.001	0.466	0.484	0.475	0.430	0.498	0.487	0.460	0.414	0.426	0.504	0.451	0.476	0.588	0.615	0.607	0.607	0.724	0.685	0.718	0.751	0.604	0.561	0.549	0.605	0.480	0.476	0.462	0.505	0.475	0.462	0.499	0.472
	0.010	0.486	0.472	0.394	0.391	0.490	0.471	0.417	0.420	0.438	0.437	0.420	0.431	0.605	0.585	0.569	0.568	0.693	0.684	0.699	0.717	0.544	0.580	0.576	0.571	0.469	0.429	0.440	0.422	0.458	0.464	0.481	0.466
CBM	0.100	0.351	0.338	0.332	0.319	0.304	0.311	0.307	0.305	0.287	0.293	0.295	0.306	0.467	0.445	0.401	0.410	0.542	0.558	0.552	0.540	0.465	0.416	0.476	0.590	0.316	0.306	0.289	0.282	0.328	0.331	0.277	0.302
	0.300	0.130	0.119	0.159	0.094	0.118	0.106	0.120	0.155	0.134	0.139	0.119	0.197	0.293	0.273	0.291	0.246	0.412	0.426	0.421	0.389	0.263	0.277	0.451	0.494	0.158	0.141	0.132	0.119	0.012	0.166	0.145	0.177

Table B.7: The normalised results for problem set A generated by GenGA* and GenGA

		GenGA*																													
		1PX				2PX				UX				PMX				OX				CX				HX				CBUX			
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
RSM	0.001	0.452	0.409	0.198	0.079	0.413	0.305	0.303	0.141	0.363	0.318	0.147	0.050	0.459	0.468	0.237	0.148	0.346	0.295	0.195	0.084	0.449	0.406	0.676	0.694	0.437	0.364	0.174	0.067	0.312	0.328	0.341	0.221
	0.010	0.399	0.208	0.119	0.106	0.350	0.178	0.208	0.053	0.363	0.205	0.110	0.092	0.328	0.295	0.149	0.158	0.386	0.217	0.086	0.057	0.343	0.396	0.458	0.706	0.341	0.135	0.097	0.063	0.334	0.270	0.200	0.254
	0.100	0.241	0.084	0.071	0.026	0.212	0.103	0.058	0.056	0.276	0.059	0.057	0.004	0.218	0.241	0.068	0.077	0.229	0.099	0.096	0.044	0.347	0.339	0.300	0.750	0.087	0.064	0.077	0.082	0.271	0.137	0.145	0.147
	0.300	0.073	0.015	0.024	0.018	0.027	0.009	0.003	0.005	0.034	0.050	0.019	0.032	0.079	0.049	0.053	0.001	0.109	0.083	0.065	0.036	0.344	0.273	0.357	0.735	0.052	0.026	0.003	0.001	0.189	0.158	0.067	0.054
	0.001	0.524	0.580	0.274	0.202	0.544	0.475	0.385	0.230	0.540	0.576	0.250	0.190	0.521	0.515	0.418	0.280	0.520	0.468	0.295	0.132	0.494	0.566	0.629	0.675	0.558	0.462	0.248	0.165	0.502	0.627	0.477	0.431
IM	0.010	0.327	0.298	0.143	0.209	0.382	0.505	0.127	0.238	0.388	0.201	0.237	0.104	0.509	0.362	0.210	0.165	0.364	0.313	0.166	0.083	0.500	0.466	0.486	0.771	0.410	0.291	0.178	0.119	0.391	0.442	0.356	0.370
	0.100	0.175	0.048	0.059	0.296	0.155	0.113	0.068	0.285	0.117	0.105	0.041	0.378	0.233	0.099	0.019	0.351	0.171	0.062	0.031	0.374	0.566	0.478	0.600	0.260	0.144	0.085	0.076	0.432	0.383	0.375	0.224
	0.300	0.173	0.037	0.009	0.029	0.158	0.153	0.059	0.023	0.238	0.135	0.009	0.006	0.294	0.103	0.041	0.002	0.146	0.044	0.032	0.458	0.350	0.312	0.684	0.278	0.116	0.020	0.002	0.282	0.205	0.088	0.133
	0.001	0.453	0.242	0.227	0.094	0.302	0.206	0.098	0.163	0.380	0.400	0.177	0.102	0.422	0.311	0.229	0.154	0.420	0.322	0.161	0.123	0.551	0.310	0.535	0.701	0.362	0.295	0.154	0.122	0.506	0.304	0.244	0.335
	0.010	0.383	0.263	0.109	0.040	0.254	0.323	0.116	0.114	0.308	0.141	0.100	0.100	0.247	0.344	0.122	0.108	0.388	0.182	0.071	0.020	0.332	0.403	0.577	0.702	0.398	0.116	0.123	0.037	0.335	0.288	0.179	0.206
SM	0.100	0.132	0.081	0.042	0.019	0.195	0.147	0.038	0.031	0.340	0.090	0.049	0.019	0.278	0.086	0.045	0.080	0.258	0.080	0.034	0.021	0.298	0.342	0.304	0.610	0.087	0.062	0.060	0.026	0.210	0.202	0.116	0.135
	0.300	0.049	0.051	0.011	0.001	0.033	0.032	0.008	0.020	0.036	0.015	0.003	0.002	0.105	0.063	0.044	0.003	0.070	0.015	0.024	0.031	0.376	0.395	0.318	0.790	0.079	0.075	0.025	0.019	0.137	0.110	0.037	0.046
	0.001	0.481	0.541	0.398	0.149	0.479	0.557	0.233	0.271	0.358	0.659	0.247	0.167	0.701	0.433	0.316	0.100	0.593	0.482	0.429	0.253	0.468	0.583	0.629	0.623	0.471	0.509	0.573	0.094	0.505	0.490	0.445	0.408
CBM	0.010	0.603	0.512	0.368	0.246	0.507	0.392	0.260	0.188	0.431	0.420	0.254	0.154	0.460	0.581	0.293	0.176	0.534	0.606	0.389	0.178	0.479	0.374	0.489	0.728	0.581	0.530	0.322	0.238	0.422	0.642	0.323	0.367
	0.100	0.519	0.439	0.355	0.206	0.458	0.432	0.223	0.134	0.573	0.480	0.310	0.149	0.429	0.438	0.259	0.207	0.489	0.483	0.265	0.142	0.531	0.363	0.575	0.591	0.466	0.617	0.282	0.132	0.401	0.511	0.528	0.487
	0.300	0.445	0.316	0.129	0.158	0.526	0.519	0.252	0.179	0.531	0.461	0.187	0.082	0.400	0.403	0.158	0.069	0.597	0.401	0.184	0.071	0.428	0.390	0.367	0.737	0.447	0.419	0.225	0.036	0.644	0.523	0.383	0.306

		GenGA																													
		1PX				2PX				UX				PMX				OX				CX				HX				CBUX			
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
RSM	0.001	0.421	0.356	0.490	0.469	0.460	0.470	0.318	0.362	0.463	0.405	0.380	0.514	0.422	0.345	0.419	0.437	0.433	0.403	0.438	0.489	0.417	0.430	0.483	0.523	0.578	0.419	0.392	0.504	0.363	0.387	0.461	0.495
	0.010	0.389	0.477	0.460	0.424	0.400	0.312	0.456	0.313	0.403	0.324	0.462	0.454	0.373	0.508	0.402	0.458	0.272	0.453	0.545	0.341	0.558	0.299	0.357	0.280	0.387	0.454	0.621	0.435	0.535	0.464	0.387	0.310
	0.100	0.611	0.445	0.351	0.485	0.572	0.518	0.446	0.361	0.392	0.455	0.402	0.551	0.523	0.403	0.457	0.395	0.544	0.573	0.500	0.416	0.338	0.411	0.473	0.361	0.365	0.370	0.634	0.459	0.418	0.473	0.307	0.386
	0.300	0.343	0.357	0.488	0.379	0.404	0.371	0.462	0.391	0.332	0.305	0.390	0.362	0.447	0.450	0.422	0.355	0.444	0.455	0.377	0.377	0.344	0.379	0.334	0.451	0.467	0.510	0.411	0.414	0.426	0.439	0.308	0.457
	0.001	0.463	0.509	0.348	0.412	0.446	0.367	0.463	0.456	0.387	0.394	0.393	0.414	0.443	0.415	0.370	0.457	0.422	0.443	0.428	0.474	0.446	0.432	0.391	0.395	0.431	0.286	0.435	0.429	0.538	0.484	0.358	0.346
IM	0.010	0.560	0.527	0.448	0.403	0.470	0.272	0.446	0.369	0.442	0.392	0.358	0.419	0.451	0.398	0.360	0.392	0.445	0.451	0.423	0.349	0.505	0.326	0.353	0.473	0.354	0.444	0.387	0.389	0.374	0.431	0.444	0.428
	0.100	0.439	0.399	0.301	0.483	0.552	0.416	0.477	0.368	0.440	0.312	0.441	0.405	0.432	0.415	0.420	0.386	0.523	0.426	0.532	0.562	0.455	0.368	0.429	0.495	0.466	0.483	0.338	0.361	0.460	0.441	0.387	0.386
	0.300	0.396	0.464	0.281	0.426	0.465	0.397	0.380	0.421	0.396	0.302	0.330	0.390	0.504	0.421	0.395	0.364	0.390	0.457	0.366	0.517	0.385	0.397	0.311	0.410	0.409	0.383	0.282	0.449	0.567	0.506	0.465	0.478
	0.001	0.455	0.342	0.375	0.551	0.422	0.345	0.366	0.436	0.456	0.367	0.355	0.450	0.466	0.418	0.278	0.370	0.397	0.419	0.368	0.414	0.395	0.526	0.417	0.346	0.499	0.480	0.441	0.440	0.302	0.413	0.572	0.404
	0.010	0.482	0.373	0.421	0.350	0.415	0.371	0.435	0.526	0.418	0.417	0.391	0.498	0.373	0.378	0.357	0.365	0.378	0.394	0.520	0.376	0.341	0.476	0.445	0.503	0.423	0.349	0.427	0.417	0.468	0.448	0.399	0.441
SM	0.100	0.434	0.254	0.487	0.530	0.391	0.421	0.533	0.418	0.453	0.382	0.436	0.326	0.455	0.397	0.431	0.414	0.513	0.524	0.381	0.312	0.471	0.421	0.343	0.507	0.464	0.351	0.341	0.420	0.435	0.488	0.491	0.416
	0.300	0.290	0.349	0.332	0.414	0.420	0.311	0.521	0.348	0.376	0.553	0.382	0.276	0.543	0.315	0.499	0.355	0.443	0.390	0.527	0.530	0.434	0.501	0.469	0.390	0.437	0.340	0.400	0.346	0.407	0.425	0.232	0.405
	0.001	0.390	0.485	0.299	0.402	0.404	0.341	0.403	0.358	0.318	0.393	0.457	0.390	0.341	0.455	0.522	0.376	0.350	0.366	0.402	0.335	0.473	0.384	0.364	0.459	0.427	0.406	0.437	0.299	0.422	0.362	0.439	0.356
CBM	0.010	0.610	0.525	0.394	0.452	0.401	0.414	0.436	0.471	0.486	0.256	0.495	0.516	0.409	0.408	0.340	0.452	0.460	0.324	0.360	0.370	0.452	0.436	0.480	0.448	0.256	0.358	0.383	0.392	0.410	0.387	0.396	0.420
	0.100	0.449	0.423	0.404	0.389	0.284	0.360	0.353	0.413	0.537	0.329	0.364	0.459	0.381	0.465	0.370	0.356	0.521	0.337	0.449	0.375	0.351	0.357	0.349	0.572	0.460	0.379	0.537	0.359	0.451	0.363	0.416	0.482
	0.300	0.306	0.521	0.546	0.464	0.472	0.410	0.512	0.474	0.455	0.356	0.550	0.504	0.474	0.372	0.372	0.442	0.359	0.428	0.439	0.381	0.459	0.329	0.426	0.399	0.364	0.414	0.512	0.421	0.407	0.399	0.398	0.384

Table B.8: The normalised results for problem set B generated by GenGA* and GenGA

GenGA*

		1PX				2PX				UX				PMX				OX				CX				HX				CBUX			
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
	0.001	0.205	0.176	0.195	0.051	0.356	0.434	0.164	0.141	0.252	0.198	0.055	0.052	0.212	0.196	0.071	0.096	0.238	0.114	0.066	0.027	0.277	0.289	0.608	0.464	0.192	0.210	0.046	0.051	0.332	0.167	0.382	0.196
	0.010	0.208	0.124	0.076	0.027	0.209	0.141	0.102	0.080	0.202	0.143	0.048	0.043	0.257	0.166	0.052	0.041	0.257	0.190	0.022	0.010	0.254	0.216	0.303	0.787	0.215	0.122	0.032	0.004	0.232	0.199	0.192	0.226
RSM	0.100	0.181	0.077	0.024	0.032	0.242	0.070	0.050	0.003	0.141	0.062	0.006	0.014	0.226	0.070	0.026	0.054	0.201	0.049	0.011	0.001	0.260	0.199	0.155	0.433	0.084	0.041	0.009	0.009	0.199	0.190	0.156	0.196
	0.300	0.078	0.013	0.010	0.004	0.115	0.048	0.005	0.001	0.082	0.025	0.007	0.000	0.109	0.037	0.002	0.001	0.067	0.030	0.001	0.001	0.253	0.209	0.186	0.425	0.023	0.053	0.005	0.000	0.143	0.162	0.112	0.062
	0.001	0.594	0.536	0.357	0.176	0.295	0.493	0.197	0.111	0.669	0.474	0.108	0.106	0.579	0.424	0.560	0.106	0.453	0.746	0.243	0.085	0.322	0.700	0.446	0.591	0.603	0.296	0.310	0.107	0.474	0.325	0.390	0.465
	0.010	0.370	0.287	0.093	0.154	0.375	0.161	0.115	0.333	0.253	0.169	0.058	0.101	0.328	0.224	0.124	0.147	0.269	0.236	0.094	0.105	0.432	0.259	0.411	0.672	0.204	0.363	0.138	0.062	0.240	0.231	0.203	0.186
IM	0.100	0.235	0.101	0.022	0.019	0.228	0.075	0.066	0.071	0.236	0.110	0.073	0.014	0.244	0.140	0.140	0.042	0.295	0.153	0.056	0.011	0.278	0.215	0.232	0.354	0.320	0.184	0.068	0.032	0.265	0.208	0.227	0.168
	0.300	0.118	0.049	0.060	0.002	0.137	0.095	0.031	0.017	0.092	0.074	0.004	0.007	0.101	0.100	0.017	0.014	0.182	0.041	0.006	0.002	0.237	0.186	0.228	0.450	0.153	0.121	0.010	0.006	0.239	0.228	0.220	0.131
	0.001	0.257	0.192	0.106	0.139	0.264	0.457	0.062	0.119	0.244	0.205	0.128	0.090	0.276	0.244	0.237	0.098	0.296	0.171	0.088	0.061	0.246	0.225	0.363	0.889	0.264	0.168	0.282	0.049	0.307	0.424	0.230	0.371
	0.010	0.248	0.135	0.073	0.027	0.189	0.161	0.077	0.076	0.229	0.119	0.025	0.051	0.243	0.234	0.147	0.058	0.252	0.189	0.060	0.019	0.213	0.233	0.238	0.460	0.229	0.131	0.049	0.014	0.237	0.260	0.224	0.215
SM	0.100	0.222	0.082	0.013	0.002	0.119	0.043	0.040	0.009	0.183	0.066	0.008	0.006	0.161	0.103	0.013	0.016	0.185	0.064	0.016	0.005	0.210	0.249	0.204	0.574	0.159	0.060	0.057	0.001	0.173	0.205	0.155	0.178
	0.300	0.066	0.041	0.002	0.001	0.051	0.052	0.001	0.004	0.054	0.014	0.003	0.001	0.123	0.031	0.002	0.005	0.072	0.010	0.004	0.001	0.271	0.253	0.214	0.416	0.079	0.065	0.006	0.001	0.138	0.150	0.072	0.053
	0.001	0.472	0.217	0.406	0.211	0.252	0.375	0.135	0.155	0.412	0.443	0.345	0.136	0.307	0.282	0.234	0.383	0.239	0.399	0.107	0.091	0.193	0.242	0.457	0.519	0.420	0.428	0.263	0.054	0.472	0.583	0.374	0.220
CBM	0.010	0.396	0.149	0.098	0.242	0.348	0.483	0.270	0.154	0.360	0.336	0.362	0.056	0.149	0.273	0.128	0.074	0.224	0.315	0.168	0.080	0.224	0.361	0.362	0.475	0.429	0.379	0.091	0.093	0.471	0.312	0.160	0.298
	0.100	0.218	0.206	0.239	0.111	0.147	0.346	0.230	0.079	0.462	0.331	0.183	0.067	0.368	0.303	0.302	0.042	0.160	0.427	0.094	0.061	0.210	0.179	0.309	0.717	0.162	0.344	0.218	0.096	0.202	0.445	0.286	0.199
	0.300	0.299	0.220	0.224	0.038	0.278	0.137	0.161	0.057	0.181	0.179	0.079	0.037	0.195	0.101	0.064	0.030	0.165	0.194	0.078	0.045	0.372	0.182	0.270	0.664	0.274	0.310	0.208	0.013	0.314	0.156	0.162	0.309

GenGA

		1PX				2PX				UX				PMX				OX				CX				HX				CBUX			
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
	0.001	0.504	0.443	0.375	0.409	0.480	0.462	0.482	0.440	0.597	0.488	0.407	0.466	0.450	0.442	0.377	0.356	0.502	0.443	0.441	0.451	0.515	0.439	0.536	0.483	0.531	0.366	0.496	0.478	0.495	0.520	0.452	0.381
	0.010	0.487	0.365	0.466	0.411	0.546	0.447	0.374	0.530	0.436	0.477	0.432	0.503	0.595	0.492	0.454	0.529	0.401	0.415	0.511	0.403	0.427	0.441	0.475	0.413	0.419	0.428	0.402	0.513	0.497	0.454	0.506	0.407
RSM	0.100	0.462	0.502	0.423	0.460	0.344	0.420	0.362	0.418	0.472	0.500	0.480	0.376	0.388	0.535	0.469	0.382	0.431	0.358	0.444	0.433	0.582	0.410	0.423	0.428	0.404	0.427	0.426	0.422	0.510	0.425	0.399	0.422
	0.300	0.472	0.487	0.438	0.520	0.334	0.452	0.490	0.529	0.369	0.473	0.453	0.519	0.435	0.516	0.539	0.480	0.415	0.425	0.498	0.478	0.472	0.533	0.369	0.426	0.477	0.436	0.428	0.483	0.579	0.478	0.497	0.496
	0.001	0.538	0.439	0.555	0.514	0.443	0.418	0.526	0.355	0.361	0.516	0.420	0.374	0.512	0.527	0.388	0.458	0.540	0.298	0.520	0.465	0.504	0.399	0.365	0.499	0.451	0.397	0.356	0.485	0.459	0.418	0.473	0.415
	0.010	0.362	0.527	0.391	0.416	0.443	0.451	0.527	0.441	0.435	0.499	0.441	0.548	0.413	0.480	0.494	0.420	0.500	0.427	0.393	0.472	0.462	0.453	0.450	0.439	0.441	0.451	0.479	0.500	0.475	0.383	0.489	0.419
IM	0.100	0.463	0.473	0.549	0.362	0.407	0.453	0.475	0.382	0.435	0.463	0.437	0.415	0.481	0.569	0.408	0.469	0.512	0.509	0.360	0.444	0.544	0.388	0.439	0.442	0.555	0.488	0.514	0.459	0.557	0.461	0.431	0.464
	0.300	0.447	0.491	0.498	0.419	0.434	0.478	0.501	0.438	0.440	0.334	0.481	0.487	0.457	0.514	0.628	0.489	0.477	0.480	0.482	0.492	0.466	0.423	0.443	0.462	0.423	0.557	0.520	0.410	0.479	0.464	0.367	0.390
	0.001	0.506	0.515	0.498	0.515	0.515	0.408	0.359	0.567	0.368	0.430	0.419	0.597	0.593	0.454	0.517	0.558	0.509	0.509	0.503	0.437	0.564	0.447	0.576	0.465	0.460	0.543	0.481	0.451	0.523	0.442	0.442	0.454
	0.010	0.555	0.498	0.467	0.409	0.543	0.368	0.523	0.465	0.415	0.459	0.437	0.482	0.479	0.388	0.521	0.489	0.500	0.438	0.435	0.525	0.402	0.522	0.379	0.486	0.508	0.487	0.473	0.427	0.477	0.492	0.396	0.451
SM	0.100	0.385	0.522	0.457	0.368	0.403	0.357	0.477	0.393	0.539	0.409	0.432	0.441	0.462	0.487	0.522	0.422	0.357	0.408	0.525	0.528	0.385	0.456	0.597	0.499	0.414	0.477	0.512	0.499	0.430	0.475	0.458	0.473
	0.300	0.501	0.452	0.454	0.514	0.461	0.351	0.547	0.584	0.460	0.355	0.478	0.493	0.564	0.371	0.472	0.450	0.374	0.554	0.512	0.537	0.552	0.520	0.590	0.361	0.354	0.421	0.448	0.499	0.445	0.460	0.475	0.464
	0.001	0.507	0.479	0.345	0.466	0.479	0.483	0.481	0.507	0.531	0.430	0.462	0.468	0.551	0.440	0.438	0.510	0.542	0.444	0.487	0.494	0.483	0.445	0.445	0.399	0.472	0.441	0.342	0.504	0.472	0.452	0.553	0.410
CBM	0.010	0.498	0.555	0.521	0.507	0.518	0.469	0.390	0.419	0.483	0.434	0.398	0.497	0.481	0.484	0.399	0.488	0.502	0.535	0.479	0.462	0.522	0.445	0.498	0.497	0.426	0.429	0.388	0.528	0.366	0.558	0.413	0.520
	0.100	0.466	0.404	0.445	0.465	0.446	0.557	0.423	0.467	0.454	0.517	0.479	0.463	0.449	0.534	0.483	0.435	0.355	0.480	0.328	0.453	0.415	0.384	0.546	0.404	0.450	0.422	0.522	0.428	0.532	0.339	0.417	0.435
	0.300	0.425	0.430	0.392	0.584	0.429	0.457	0.527	0.573	0.477	0.489	0.508	0.389	0.400	0.537	0.387	0.422	0.445	0.624	0.440	0.457	0.483	0.383	0.440	0.396	0.439	0.484	0.482	0.467	0.459	0.548	0.475	0.328

Table B.9: The normalised results for problem set C generated by GenGA* and GenGA

		GenGA*																													
		1PX				2PX				UX				PMX				OX				CX				HX				CBUX			
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
FSM	0.001	0.739	0.769	0.660	0.564	0.761	0.805	0.643	0.622	0.790	0.722	0.704	0.388	0.602	0.538	0.473	0.512	0.799	0.806	0.587	0.632	0.706	0.806	0.861	0.855	0.786	0.735	0.585	0.407	0.833	0.756	0.733	0.741
	0.010	0.534	0.586	0.495	0.375	0.571	0.577	0.497	0.440	0.633	0.516	0.496	0.336	0.487	0.448	0.383	0.297	0.684	0.595	0.455	0.425	0.641	0.647	0.654	0.869	0.620	0.539	0.483	0.309	0.689	0.586	0.583	0.498
	0.100	0.493	0.418	0.418	0.338	0.528	0.459	0.374	0.311	0.468	0.443	0.307	0.203	0.394	0.382	0.238	0.187	0.521	0.459	0.287	0.352	0.522	0.517	0.517	0.853	0.498	0.450	0.342	0.235	0.592	0.533	0.427	0.387
	0.300	0.539	0.430	0.311	0.290	0.447	0.373	0.347	0.251	0.388	0.296	0.279	0.166	0.336	0.289	0.158	0.249	0.458	0.357	0.267	0.222	0.538	0.567	0.553	0.871	0.436	0.340	0.251	0.137	0.521	0.379	0.340	0.335
IM	0.001	0.853	0.832	0.728	0.672	0.794	0.807	0.786	0.642	0.865	0.807	0.632	0.465	0.871	0.764	0.681	0.517	0.817	0.797	0.737	0.579	0.852	0.778	0.846	0.867	0.855	0.770	0.588	0.548	0.763	0.824	0.741	0.806
	0.010	0.768	0.768	0.663	0.650	0.800	0.791	0.653	0.527	0.813	0.802	0.576	0.401	0.800	0.657	0.699	0.546	0.726	0.693	0.596	0.610	0.860	0.788	0.827	0.871	0.781	0.741	0.561	0.446	0.758	0.749	0.722	0.664
	0.100	0.633	0.582	0.499	0.658	0.643	0.552	0.444	0.641	0.607	0.504	0.341	0.668	0.578	0.509	0.425	0.771	0.596	0.533	0.427	0.686	0.723	0.716	0.808	0.687	0.598	0.556	0.298	0.647	0.638	0.603	0.564	
	0.300	0.635	0.542	0.529	0.401	0.688	0.528	0.404	0.281	0.657	0.541	0.413	0.197	0.594	0.544	0.380	0.208	0.608	0.560	0.374	0.242	0.709	0.816	0.673	0.822	0.659	0.567	0.378	0.231	0.758	0.637	0.536	0.434
	0.001	0.763	0.734	0.597	0.589	0.821	0.723	0.598	0.555	0.754	0.709	0.502	0.416	0.703	0.776	0.599	0.464	0.702	0.685	0.692	0.562	0.718	0.745	0.818	0.887	0.793	0.750	0.627	0.410	0.841	0.752	0.679	0.636
SM	0.010	0.633	0.582	0.532	0.491	0.624	0.565	0.466	0.434	0.559	0.613	0.397	0.243	0.572	0.638	0.398	0.336	0.674	0.599	0.450	0.410	0.635	0.632	0.735	0.817	0.508	0.557	0.361	0.341	0.600	0.512	0.568	0.526
	0.100	0.513	0.489	0.414	0.376	0.565	0.483	0.352	0.314	0.462	0.350	0.339	0.195	0.564	0.451	0.344	0.209	0.494	0.406	0.348	0.265	0.594	0.539	0.578	0.813	0.496	0.414	0.349	0.237	0.590	0.504	0.437	0.444
	0.300	0.362	0.427	0.375	0.313	0.371	0.396	0.353	0.279	0.427	0.370	0.279	0.194	0.385	0.313	0.242	0.101	0.480	0.339	0.302	0.223	0.513	0.662	0.559	0.868	0.393	0.344	0.282	0.190	0.636	0.446	0.368	0.285
	0.001	0.857	0.801	0.807	0.626	0.829	0.827	0.718	0.627	0.889	0.900	0.692	0.480	0.819	0.789	0.657	0.587	0.898	0.816	0.687	0.619	0.860	0.861	0.837	0.862	0.908	0.849	0.618	0.425	0.842	0.859	0.765	0.789
CBM	0.010	0.880	0.766	0.797	0.611	0.848	0.787	0.671	0.614	0.864	0.789	0.666	0.449	0.855	0.823	0.705	0.593	0.797	0.818	0.683	0.597	0.845	0.803	0.794	0.867	0.842	0.843	0.700	0.442	0.853	0.866	0.759	0.777
	0.100	0.868	0.805	0.734	0.637	0.872	0.842	0.682	0.631	0.877	0.754	0.694	0.458	0.840	0.836	0.627	0.577	0.892	0.811	0.714	0.615	0.875	0.824	0.800	0.868	0.850	0.762	0.722	0.472	0.860	0.829	0.788	0.756
	0.300	0.946	0.831	0.719	0.647	0.761	0.756	0.694	0.615	0.828	0.810	0.579	0.432	0.859	0.773	0.604	0.479	0.880	0.770	0.660	0.538	0.865	0.815	0.858	0.859	0.874	0.699	0.592	0.571	0.783	0.846	0.725	0.814

		GenGA																													
		1PX				2PX				UX				PMX				OX				CX				HX				CBUX			
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
FSM	0.001	0.889	0.851	0.840	0.386	0.862	0.846	0.856	0.858	0.863	0.856	0.847	0.853	0.856	0.833	0.837	0.860	0.876	0.870	0.843	0.865	0.875	0.873	0.848	0.844	0.856	0.885	0.845	0.847	0.860	0.836	0.864	0.855
	0.010	0.862	0.862	0.854	0.383	0.805	0.872	0.854	0.846	0.861	0.831	0.845	0.800	0.876	0.847	0.860	0.866	0.864	0.850	0.838	0.850	0.832	0.865	0.861	0.867	0.852	0.885	0.864	0.838	0.839	0.874	0.844	0.881
	0.100	0.861	0.849	0.845	0.821	0.856	0.835	0.869	0.826	0.872	0.869	0.849	0.848	0.853	0.835	0.863	0.865	0.847	0.888	0.861	0.846	0.861	0.835	0.870	0.866	0.856	0.867	0.894	0.852	0.839	0.876	0.848	0.862
	0.300	0.867	0.852	0.868	0.871	0.872	0.835	0.834	0.871	0.864	0.853	0.836	0.849	0.845	0.862	0.853	0.849	0.842	0.828	0.861	0.866	0.855	0.843	0.841	0.856	0.880	0.855	0.850	0.895	0.850	0.868	0.841	0.843
	0.001	0.876	0.836	0.861	0.861	0.852	0.840	0.872	0.863	0.842	0.833	0.869	0.858	0.881	0.846	0.855	0.847	0.858	0.845	0.853	0.852	0.864	0.843	0.853	0.856	0.842	0.866	0.835	0.859	0.840	0.860	0.841	0.831
IM	0.010	0.857	0.857	0.862	0.854	0.873	0.870	0.851	0.864	0.856	0.861	0.821	0.831	0.861	0.853	0.874	0.844	0.863	0.836	0.873	0.844	0.843	0.854	0.849	0.864	0.854	0.862	0.841	0.865	0.865	0.839	0.863	0.843
	0.100	0.842	0.924	0.857	0.839	0.864	0.875	0.860	0.877	0.871	0.868	0.852	0.856	0.840	0.838	0.873	0.864	0.844	0.835	0.867	0.826	0.903	0.860	0.843	0.881	0.855	0.866	0.868	0.900	0.877	0.841	0.854	
	0.300	0.870	0.876	0.866	0.839	0.894	0.862	0.843	0.898	0.835	0.862	0.900	0.866	0.880	0.818	0.869	0.863	0.855	0.858	0.878	0.851	0.859	0.896	0.839	0.801	0.837	0.838	0.854	0.845	0.885	0.826	0.868	0.848
	0.001	0.838	0.872	0.857	0.851	0.867	0.834	0.849	0.858	0.841	0.847	0.854	0.872	0.865	0.859	0.845	0.859	0.870	0.859	0.844	0.850	0.859	0.900	0.869	0.845	0.894	0.864	0.865	0.843	0.840	0.741	0.846	0.848
SM	0.010	0.872	0.853	0.871	0.863	0.850	0.861	0.877	0.862	0.845	0.885	0.851	0.831	0.855	0.860	0.845	0.844	0.812	0.850	0.878	0.849	0.838	0.863	0.904	0.868	0.843	0.862	0.851	0.847	0.694	0.874	0.862	0.808
	0.100	0.871	0.842	0.852	0.889	0.914	0.890	0.861	0.867	0.899	0.866	0.825	0.828	0.843	0.854	0.900	0.839	0.822	0.842	0.856	0.888	0.865	0.836	0.843	0.848	0.832	0.866	0.862	0.855	0.707	0.867	0.849	0.860
	0.300	0.858	0.880	0.874	0.866	0.872	0.840	0.838	0.877	0.881	0.844	0.888	0.855	0.862	0.858	0.847	0.868	0.865	0.825	0.861	0.880	0.848	0.859	0.864	0.876	0.832	0.862	0.862	0.874	0.700	0.866	0.851	0.898
	0.001	0.562	0.604	0.610	0.721	0.599	0.613	0.607	0.599	0.569	0.582	0.597	0.593	0.597	0.577	0.577	0.601	0.620	0.568	0.558	0.567	0.468	0.452	0.460	0.582	0.449	0.459	0.470	0.445	0.450	0.465	0.434	0.465
CBM	0.010	0.573	0.566	0.586	0.699	0.572	0.590	0.579	0.588	0.627	0.613	0.613	0.568	0.603	0.595	0.596	0.596	0.576	0.556	0.579	0.597	0.457	0.457	0.450	0.598	0.470	0.439	0.475	0.477	0.492	0.436	0.464	0.443
	0.100	0.600	0.599	0.599	0.715	0.584	0.598	0.603	0.617	0.604	0.608	0.569	0.613	0.556	0.583	0.605	0.584	0.608	0.611	0.593	0.543	0.470	0.449	0.439	0.593	0.464	0.440	0.457	0.469	0.480	0.444	0.448	0.444
	0.300	0.624	0.574	0.593	0.610	0.593	0.578	0.598	0.591	0.586	0.584	0.567	0.603	0.610	0.562	0.582	0.582	0.602	0.597	0.612	0.602	0.453	0.460	0.441	0.447	0.469	0.478	0.485	0.435	0.472	0.436	0.449	0.429

Table B.10: The normalised results for problem set D generated by GenGA* and GenGA

														GenGA*																				
				1PX				2PX				UX				PMX				OX				CX				HX				CBUX			
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
	0.001	0.002	0.003	0.004	0.004	0.003	0.002	0.003	0.003	0.004	0.002	0.004	0.004	0.003	0.003	0.003	0.003	0.973	0.970	0.865	0.800	0.975	0.981	0.988	0.979	0.975	0.975	0.979	0.874	0.977	0.979	0.976	0.972
	0.010	0.002	0.003	0.003	0.003	0.003	0.002	0.004	0.004	0.003	0.003	0.003	0.004	0.002	0.003	0.003	0.003	0.932	0.907	0.813	0.742	0.931	0.943	0.980	0.987	0.936	0.917	0.905	0.833	0.927	0.922	0.925	0.924
RSM	0.100	0.004	0.003	0.003	0.004	0.004	0.003	0.003	0.004	0.004	0.003	0.004	0.004	0.003	0.003	0.003	0.004	0.796	0.755	0.671	0.656	0.810	0.838	0.926	0.989	0.791	0.769	0.769	0.744	0.792	0.799	0.800	0.810
	0.300	0.004	0.003	0.002	0.004	0.003	0.003	0.003	0.003	0.003	0.002	0.003	0.003	0.002	0.003	0.003	0.005	0.004	0.762	0.713	0.630	0.609	0.778	0.803	0.884	0.983	0.765	0.741	0.707	0.688	0.756	0.761	0.764	0.775
	0.001	0.752	0.755	0.759	0.745	0.752	0.751	0.757	0.756	0.754	0.755	0.751	0.751	0.757	0.757	0.757	0.748	0.754	0.984	0.978	0.864	0.795	0.989	0.988	0.979	0.986	0.990	0.981	0.983	0.857	0.978	0.981	0.985	0.981
IM	0.010	0.755	0.755	0.756	0.753	0.754	0.755	0.751	0.758	0.752	0.756	0.752	0.752	0.758	0.752	0.755	0.752	0.981	0.972	0.884	0.773	0.979	0.985	0.985	0.989	0.983	0.976	0.982	0.783	0.985	0.979	0.979	0.979	0.975
	0.100	0.754	0.756	0.753	0.732	0.760	0.751	0.754	0.756	0.762	0.759	0.755	0.755	0.759	0.755	0.755	0.755	0.757	0.959	0.957	0.858	0.706	0.969	0.969	0.969	0.952	0.963	0.950	0.791	0.961	0.963	0.959	0.959	0.965
	0.300	0.755	0.757	0.758	0.755	0.756	0.748	0.756	0.761	0.754	0.754	0.755	0.756	0.757	0.757	0.758	0.756	0.750	0.955	0.933	0.822	0.760	0.953	0.952	0.974	0.984	0.958	0.952	0.947	0.928	0.951	0.954	0.952	0.933
	0.001	0.759	0.756	0.755	0.762	0.755	0.753	0.756	0.752	0.753	0.759	0.758	0.757	0.752	0.754	0.754	0.754	0.756	0.983	0.971	0.875	0.786	0.978	0.974	0.989	0.982	0.978	0.976	0.970	0.871	0.977	0.979	0.978	0.974
	0.010	0.758	0.760	0.752	0.757	0.752	0.757	0.752	0.754	0.748	0.755	0.755	0.758	0.759	0.755	0.758	0.758	0.932	0.916	0.819	0.728	0.938	0.951	0.976	0.987	0.924	0.909	0.892	0.816	0.927	0.910	0.918	0.911
SM	0.100	0.757	0.751	0.758	0.757	0.749	0.756	0.760	0.754	0.755	0.755	0.759	0.760	0.759	0.750	0.756	0.762	0.788	0.740	0.655	0.646	0.805	0.827	0.925	0.984	0.757	0.729	0.744	0.720	0.759	0.763	0.764	0.764	0.776
	0.300	0.755	0.755	0.760	0.753	0.754	0.751	0.755	0.759	0.757	0.756	0.757	0.759	0.756	0.754	0.756	0.756	0.758	0.706	0.605	0.595	0.774	0.789	0.878	0.985	0.727	0.710	0.693	0.674	0.739	0.726	0.734	0.735
	0.001	0.755	0.758	0.753	0.759	0.753	0.756	0.754	0.752	0.759	0.755	0.756	0.758	0.756	0.757	0.763	0.750	0.764	0.766	0.766	0.574	0.763	0.762	0.767	0.765	0.761	0.764	0.760	0.758	0.753	0.762	0.760	0.760	0.766
CBM	0.010	0.755	0.759	0.757	0.758	0.756	0.752	0.759	0.751	0.754	0.753	0.759	0.757	0.755	0.749	0.755	0.754	0.762	0.762	0.759	0.544	0.739	0.764	0.772	0.759	0.757	0.761	0.764	0.757	0.755	0.767	0.763	0.763	0.766
	0.100	0.756	0.752	0.752	0.756	0.756	0.757	0.759	0.755	0.756	0.761	0.755	0.756	0.759	0.752	0.754	0.752	0.760	0.761	0.763	0.729	0.764	0.769	0.764	0.765	0.755	0.764	0.755	0.772	0.759	0.763	0.761	0.761	0.762
	0.300	0.758	0.750	0.757	0.754	0.760	0.757	0.753	0.759	0.752	0.752	0.756	0.757	0.756	0.756	0.756	0.756	0.762	0.769	0.772	0.760	0.760	0.761	0.770	0.764	0.762	0.763	0.759	0.758	0.771	0.767	0.763	0.763	0.765
														GenGA																				
				1PX				2PX				UX				PMX				OX				CX				HX				CBUX			
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
	0.001	0.003	0.003	0.005	0.005	0.004	0.003	0.003	0.004	0.005	0.003	0.005	0.005	0.003	0.004	0.004	0.004	0.749	0.686	0.604	0.546	0.756	0.763	0.779	0.815	0.774	0.777	0.790	0.826	0.772	0.774	0.793	0.823
	0.010	0.003	0.004	0.003	0.004	0.004	0.002	0.005	0.006	0.004	0.005	0.004	0.005	0.003	0.004	0.004	0.004	0.757	0.692	0.599	0.566	0.765	0.755	0.784	0.816	0.761	0.774	0.776	0.839	0.773	0.776	0.779	0.823
RSM	0.100	0.005	0.004	0.003	0.005	0.005	0.003	0.004	0.005	0.003	0.003	0.005	0.005	0.003	0.004	0.004	0.003	0.754	0.695	0.588	0.556	0.769	0.775	0.782	0.817	0.773	0.780	0.788	0.831	0.773	0.778	0.784	0.818
	0.300	0.005	0.004	0.003	0.005	0.004	0.004	0.003	0.004	0.003	0.003	0.003	0.004	0.003	0.003	0.003	0.003	0.764	0.688	0.603	0.564	0.767	0.797	0.782	0.816	0.767	0.780	0.782	0.823	0.765	0.782	0.782	0.819
	0.001	0.958	0.950	0.958	0.953	0.955	0.958	0.961	0.959	0.958	0.952	0.959	0.955	0.956	0.952	0.955	0.954	0.759	0.695	0.589	0.554	0.771	0.763	0.776	0.812	0.771	0.778	0.780	0.821	0.767	0.766	0.778	0.818
	0.010	0.962	0.961	0.961	0.951	0.951	0.959	0.958	0.952	0.954	0.957	0.954	0.955	0.956	0.953	0.954	0.949	0.745	0.693	0.598	0.556	0.785	0.766	0.781	0.805	0.757	0.764	0.782	0.831	0.763	0.760	0.791	0.831
IM	0.100	0.956	0.961	0.956	0.955	0.952	0.960	0.953	0.955	0.954	0.956	0.950	0.956	0.957	0.961	0.956	0.961	0.762	0.705	0.605	0.569	0.778	0.777	0.781	0.817	0.774	0.769	0.783	0.823	0.783	0.784	0.790	0.828
	0.300	0.956	0.957	0.950	0.958	0.958	0.962	0.959	0.952	0.955	0.954	0.955	0.953	0.951	0.956	0.954	0.957	0.770	0.706	0.621	0.591	0.781	0.787	0.792	0.822	0.787	0.789	0.784	0.815	0.785	0.780	0.795	0.827
	0.001	0.953	0.953	0.948	0.952	0.962	0.957	0.955	0.951	0.964	0.953	0.951	0.952	0.955	0.953	0.955	0.960	0.755	0.674	0.588	0.551	0.765	0.767	0.783	0.827	0.767	0.772	0.786	0.821	0.767	0.772	0.795	0.821
	0.010	0.963	0.958	0.958	0.952	0.958	0.954	0.963	0.954	0.960	0.957	0.956	0.960	0.949	0.952	0.954	0.946	0.743	0.677	0.586	0.547	0.776	0.781	0.795	0.818	0.770	0.776	0.779	0.813	0.761	0.765	0.783	0.823
SM	0.100	0.963	0.960	0.957	0.950	0.958	0.951	0.955	0.954	0.963	0.954	0.955	0.956	0.954	0.962	0.953	0.957	0.752	0.670	0.596	0.566	0.764	0.770	0.777	0.820	0.771	0.778	0.799	0.822	0.770	0.785	0.769	0.822
	0.300	0.960	0.952	0.962	0.956	0.953	0.961	0.962	0.959	0.958	0.960	0.955	0.965	0.953	0.962	0.955	0.958	0.762	0.691	0.592	0.556	0.782	0.786	0.786	0.819	0.773	0.782	0.782	0.830	0.770	0.766	0.786	0.825
	0.001	0.962	0.960	0.953	0.953	0.954	0.955	0.956	0.955	0.960	0.959	0.955	0.953	0.950	0.952	0.967	0.955	0.969	0.973	0.972	0.729	0.968	0.968	0.973	0.971	0.966	0.970	0.964	0.962	0.956	0.967	0.964	0.972
CBM	0.010	0.961	0.953	0.959	0.957	0.954	0.954	0.950	0.956	0.956	0.960	0.949	0.954	0.957	0.959	0.953	0.960	0.967	0.967	0.964	0.691	0.963	0.969	0.980	0.964	0.961	0.966	0.970	0.961	0.958	0.974	0.969	0.972
	0.100	0.958	0.954	0.956	0.960	0.954	0.956	0.954	0.954	0.958	0.956	0.954	0.950	0.960	0.961	0.958	0.957	0.965	0.966	0.968	0.926	0.970	0.976	0.970	0.970	0.959	0.970	0.959	0.980	0.964	0.969	0.967	0.967
	0.300	0.952	0.957	0.957	0.958	0.954	0.955	0.953	0.958	0.951	0.953	0.956	0.962	0.955	0.952	0.962	0.952	0.977	0.980	0.965	0.964	0.966	0.978	0.970	0.967	0.968	0.963	0.963	0.978	0.974	0.968	0.969	0.971

Table B.11: The normalised results for problem set E generated by GenGA* and GenGA

		GenGA*																													
		1PX				2PX				UX				PMX				OX				CX				HX				CBUX			
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
RSM	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.931	0.917	0.769	0.635	0.921	0.940	0.962	0.971	0.941	0.927	0.908	0.664	0.928	0.942	0.927	0.929
	0.010	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.703	0.672	0.551	0.483	0.746	0.786	0.928	0.974	0.733	0.711	0.665	0.548	0.741	0.745	0.746	0.752
	0.100	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.359	0.276	0.203	0.197	0.401	0.447	0.701	0.964	0.381	0.361	0.362	0.312	0.382	0.388	0.397	0.407
	0.300	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.287	0.185	0.109	0.122	0.319	0.350	0.539	0.975	0.305	0.273	0.238	0.217	0.300	0.290	0.301	0.321
	0.001	0.109	0.114	0.118	0.108	0.114	0.118	0.113	0.116	0.115	0.116	0.113	0.118	0.116	0.116	0.113	0.941	0.945	0.808	0.640	0.959	0.959	0.959	0.954	0.965	0.969	0.951	0.955	0.656	0.945	0.953	0.960	0.946
IM	0.010	0.108	0.118	0.112	0.109	0.113	0.112	0.116	0.112	0.114	0.118	0.112	0.116	0.116	0.116	0.120	0.906	0.904	0.774	0.600	0.902	0.902	0.902	0.950	0.955	0.896	0.914	0.888	0.474	0.903	0.881	0.882	0.889
	0.100	0.109	0.114	0.117	0.114	0.116	0.113	0.115	0.115	0.112	0.116	0.116	0.117	0.116	0.117	0.118	0.798	0.748	0.650	0.411	0.799	0.798	0.869	0.962	0.784	0.785	0.733	0.422	0.762	0.760	0.764	0.757
	0.300	0.114	0.113	0.110	0.116	0.111	0.117	0.113	0.116	0.113	0.119	0.115	0.113	0.114	0.717	0.696	0.564	0.462	0.720	0.741	0.849	0.967	0.721	0.716	0.736	0.715	0.713	0.696	0.715	0.710			
	0.001	0.110	0.117	0.111	0.118	0.116	0.114	0.116	0.113	0.112	0.115	0.114	0.115	0.115	0.114	0.945	0.903	0.775	0.636	0.940	0.947	0.953	0.972	0.924	0.923	0.903	0.639	0.926	0.909	0.922	0.930		
	0.010	0.113	0.111	0.115	0.117	0.113	0.118	0.113	0.112	0.116	0.113	0.111	0.112	0.114	0.107	0.112	0.110	0.676	0.664	0.538	0.440	0.717	0.768	0.923	0.960	0.699	0.662	0.632	0.484	0.718	0.706	0.681	0.664
SM	0.100	0.114	0.112	0.113	0.115	0.111	0.113	0.111	0.113	0.110	0.116	0.114	0.115	0.111	0.114	0.113	0.112	0.328	0.278	0.214	0.223	0.329	0.403	0.685	0.965	0.330	0.296	0.299	0.252	0.328	0.313	0.338	0.359
	0.300	0.115	0.113	0.118	0.112	0.117	0.120	0.109	0.116	0.111	0.115	0.114	0.111	0.112	0.113	0.112	0.110	0.262	0.197	0.151	0.150	0.288	0.319	0.524	0.982	0.278	0.257	0.225	0.201	0.280	0.259	0.259	0.271
	0.001	0.114	0.111	0.110	0.116	0.115	0.118	0.113	0.117	0.111	0.112	0.115	0.116	0.111	0.115	0.115	0.115	0.921	0.922	0.809	0.586	0.942	0.941	0.946	0.948	0.920	0.935	0.912	0.671	0.935	0.930	0.925	0.911
CBM	0.010	0.112	0.114	0.114	0.116	0.113	0.108	0.116	0.111	0.117	0.117	0.114	0.115	0.114	0.117	0.116	0.116	0.823	0.807	0.706	0.505	0.828	0.854	0.916	0.930	0.823	0.824	0.802	0.629	0.818	0.807	0.820	0.830
	0.100	0.117	0.114	0.112	0.110	0.104	0.112	0.116	0.116	0.111	0.116	0.113	0.115	0.112	0.116	0.119	0.111	0.587	0.570	0.478	0.429	0.584	0.619	0.805	0.945	0.597	0.576	0.559	0.476	0.600	0.599	0.579	0.591
	0.300	0.119	0.115	0.112	0.116	0.115	0.116	0.114	0.112	0.115	0.112	0.117	0.114	0.116	0.109	0.112	0.109	0.525	0.504	0.398	0.375	0.521	0.551	0.675	0.947	0.519	0.520	0.479	0.412	0.516	0.520	0.515	0.518

		GenGA																													
		1PX				2PX				UX				PMX				OX				CX				HX				CBUX			
		0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1	0.01	0.1	0.5	1
RSM	0.001	0.120	0.124	0.111	0.111	0.120	0.124	0.113	0.111	0.129	0.117	0.121	0.126	0.126	0.118	0.126	0.122	0.126	0.588	0.472	0.270	0.201	0.668	0.668	0.720	0.767	0.686	0.661	0.637	0.757	0.740	0.639	0.598	0.683
	0.010	0.116	0.119	0.116	0.106	0.122	0.112	0.110	0.107	0.118	0.117	0.118	0.111	0.117	0.104	0.115	0.116	0.117	0.586	0.458	0.246	0.197	0.648	0.646	0.807	0.790	0.710	0.596	0.660	0.758	0.610	0.609	0.660	0.662
	0.100	0.110	0.092	0.083	0.080	0.111	0.092	0.089	0.073	0.107	0.098	0.092	0.094	0.102	0.096	0.089	0.084	0.556	0.392	0.209	0.192	0.618	0.600	0.673	0.768	0.588	0.554	0.570	0.809	0.568	0.560	0.609	0.676
	0.300	0.080	0.080	0.059	0.049	0.094	0.084	0.059	0.053	0.086	0.079	0.065	0.068	0.085	0.076	0.068	0.054	0.545	0.356	0.214	0.174	0.549	0.569	0.627	0.781	0.541	0.567	0.602	0.788	0.537	0.547	0.585	0.664
	0.001	0.252	0.258	0.251	0.246	0.245	0.249	0.238	0.251	0.255	0.257	0.251	0.242	0.261	0.259	0.252	0.253	0.617	0.439	0.275	0.198	0.630	0.627	0.659	0.813	0.650	0.633	0.624	0.872	0.685	0.634	0.615	0.638
IM	0.010	0.252	0.257	0.249	0.231	0.246	0.255	0.254	0.237	0.248	0.255	0.254	0.251	0.260	0.256	0.249	0.260	0.605	0.434	0.242	0.207	0.645	0.726	0.662	0.793	0.623	0.645	0.613	0.738	0.592	0.658	0.591	0.711
	0.100	0.252	0.250	0.248	0.244	0.252	0.246	0.255	0.242	0.255	0.251	0.253	0.252	0.253	0.246	0.247	0.245	0.589	0.391	0.200	0.187	0.608	0.639	0.703	0.872	0.606	0.623	0.638	0.709	0.595	0.642	0.624	0.720
	0.300	0.250	0.246	0.252	0.239	0.251	0.247	0.246	0.229	0.253	0.256	0.254	0.252	0.251	0.250	0.251	0.249	0.576	0.407	0.284	0.227	0.626	0.623	0.673	0.811	0.595	0.644	0.618	0.733	0.615	0.625	0.626	0.703
	0.001	0.261	0.254	0.235	0.245	0.258	0.257	0.247	0.243	0.243	0.247	0.253	0.255	0.253	0.258	0.256	0.259	0.595	0.441	0.271	0.193	0.711	0.624	0.691	0.866	0.659	0.662	0.718	0.699	0.674	0.661	0.596	0.687
	0.010	0.244	0.251	0.245	0.233	0.249	0.249	0.243	0.235	0.258	0.255	0.246	0.253	0.241	0.245	0.248	0.248	0.646	0.414	0.260	0.164	0.646	0.612	0.697	0.826	0.624	0.623	0.604	0.772	0.578	0.651	0.590	0.703
SM	0.100	0.242	0.223	0.212	0.210	0.254	0.224	0.209	0.205	0.235	0.227	0.228	0.215	0.239	0.229	0.226	0.205	0.481	0.335	0.206	0.167	0.568	0.584	0.664	0.800	0.532	0.541	0.568	0.775	0.504	0.531	0.573	0.677
	0.300	0.225	0.208	0.179	0.180	0.229	0.204	0.192	0.185	0.224	0.202	0.187	0.185	0.216	0.214	0.192	0.183	0.518	0.362	0.205	0.177	0.532	0.571	0.635	0.848	0.533	0.547	0.575	0.720	0.551	0.544	0.577	0.772
	0.001	0.252	0.249	0.247	0.241	0.258	0.248	0.250	0.254	0.258	0.261	0.261	0.253	0.258	0.265	0.253	0.255	0.743	0.556	0.418	0.229	0.757	0.762	0.790	0.834	0.758	0.800	0.719	0.891	0.784	0.721	0.733	0.741
CBM	0.010	0.256	0.249	0.248	0.243	0.245	0.236	0.241	0.235	0.244	0.255	0.250	0.259	0.249	0.247	0.230	0.244	0.631	0.537	0.377	0.212	0.698	0.747	0.767	0.827	0.700	0.729	0.663	0.782	0.732	0.662	0.683	0.738
	0.100	0.215	0.225	0.218	0.209	0.217	0.224	0.215	0.211	0.215	0.206	0.209	0.215	0.208	0.209	0.219	0.211	0.585	0.456	0.323	0.292	0.707	0.657	0.802	0.852	0.706	0.625	0.668	0.874	0.674	0.647	0.683	0.736
	0.300	0.183	0.184	0.183	0.176	0.181	0.178	0.175	0.177	0.176	0.187	0.180	0.185	0.192	0.195	0.175	0.188	0.610	0.462	0.339	0.310	0.668	0.679	0.726	0.891	0.620	0.647	0.694	0.772	0.583	0.631	0.667	0.728

Appendix C

Baseline GA Results

Chapter 5 proposes an indirect chromosome encoding scheme of WSRP problems with six novel cost-based genetic operators tailored for the indirect chromosome encoding scheme. Experiments were conducted to assess the performance of the proposed GA on the 42 real-world WSRP scenarios from Table 2.4. This section provides all results using the parameter settings shown in Table 5.1.

Table C.1: The normalised results for problem set A

$M = 100$

		FCX			CGX			HUX			PMFCX			PMGreedyX			1PX			2PX			UX		
		0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1
FCF	0.01	0.028322	0.026961	0.019284	0.045633	0.052637	0.020194	0.106142	0.030982	0.019941	0.142006	0.169254	0.208737	0.050045	0.048239	0.048732	0.129808	0.087703	0.049366	0.11009	0.110719	0.075151	0.099553	0.049186	0.025973
	0.1	0.028387	0.008942	0.008918	0.025098	0.024119	0.011959	0.063965	0.02666	0.01418	0.08201	0.116513	0.193639	0.048094	0.059932	0.049173	0.078317	0.066536	0.041336	0.112615	0.077845	0.070204	0.060232	0.050705	0.018379
	0.3	0.009841	0.004812	0.007262	0.026653	0.021287	0.020675	0.045109	0.028801	0.014801	0.05449	0.074836	0.164274	0.026638	0.020858	0.030869	0.042154	0.056802	0.03635	0.033368	0.05933	0.02811	0.040445	0.019979	0.011748
IM	0.01	0.091693	0.06416	0.05779	0.045981	0.043591	0.02256	0.167065	0.053391	0.018946	0.299268	0.327421	0.403749	0.06351	0.055467	0.054592	0.143528	0.082992	0.077777	0.178145	0.170247	0.140732	0.120991	0.051394	0.021929
	0.1	0.057973	0.050874	0.048995	0.033251	0.040341	0.041922	0.121464	0.041435	0.026913	0.352377	0.319046	0.413277	0.070518	0.048082	0.051016	0.102173	0.089561	0.057723	0.167907	0.134669	0.117493	0.104131	0.047061	0.028428
	0.3	0.046549	0.039402	0.043177	0.04116	0.03621	0.017039	0.089817	0.024272	0.02189	0.293004	0.323879	0.418917	0.035307	0.038578	0.06463	0.092598	0.074163	0.037241	0.151449	0.086402	0.065019	0.069152	0.032881	0.020382
SM	0.01	0.035783	0.04954	0.030317	0.044539	0.027366	0.008063	0.10616	0.056412	0.016842	0.294905	0.270213	0.372856	0.052022	0.025434	0.033216	0.112474	0.083321	0.061454	0.1279	0.062552	0.044458	0.085405	0.04071	0.025356
	0.1	0.031749	0.019475	0.017594	0.014374	0.031678	0.011014	0.0657	0.029191	0.006211	0.273144	0.24587	0.308476	0.043379	0.019038	0.029067	0.079771	0.057143	0.020735	0.077043	0.02969	0.038108	0.085568	0.029629	0.010111
	0.3	0.020185	0.014042	0.012909	0.016839	0.015551	0.018428	0.036528	0.012405	0.004775	0.1939	0.231713	0.371908	0.014816	0.011152	0.029444	0.037752	0.030162	0.011546	0.061977	0.026582	0.010279	0.0453	0.014584	0.005983
RSM	0.01	0.054998	0.032557	0.027199	0.04642	0.036823	0.012507	0.134345	0.047434	0.015323	0.30151	0.324548	0.391817	0.050184	0.049467	0.043558	0.262763	0.07716	0.063402	0.143463	0.096286	0.074153	0.087267	0.046786	0.023531
	0.1	0.035638	0.0213	0.01826	0.035962	0.030702	0.026332	0.084038	0.035413	0.010402	0.222657	0.251047	0.328192	0.026907	0.016192	0.031718	0.081662	0.052179	0.055716	0.076719	0.062004	0.069681	0.063575	0.037607	0.014936
	0.3	0.027519	0.01606	0.013344	0.017378	0.020836	0.005379	0.05133	0.017579	0.011971	0.215445	0.226469	0.320276	0.040605	0.020647	0.032972	0.052835	0.058952	0.013598	0.046161	0.03559	0.032943	0.049646	0.017731	0.006262
Zero Flip	0.01	0.101204	0.072765	0.060494	0.069056	0.043129	0.023348	0.166351	0.057628	0.016314	0.387105	0.376883	0.418332	0.063561	0.033788	0.033478	0.188493	0.070856	0.055009	0.222686	0.205556	0.174044	0.123394	0.057422	0.023974
	0.1	0.073694	0.035805	0.046853	0.062724	0.035376	0.022881	0.159499	0.055953	0.032773	0.406691	0.380478	0.421105	0.063827	0.047241	0.064282	0.200148	0.102937	0.105674	0.254452	0.195325	0.189577	0.156077	0.055978	0.025266
	0.3	0.070018	0.031854	0.037651	0.047915	0.024565	0.029152	0.156347	0.060035	0.026533	0.413018	0.38052	0.504319	0.073721	0.061829	0.060602	0.177023	0.116089	0.074827	0.274647	0.209557	0.13463	0.1405	0.064808	0.035483

$M = 250$

		FCX			CGX			HUX			PMFCX			PMGreedyX			1PX			2PX			UX		
		0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1
FCF	0.01	0.032041	0.014324	0.00963	0.040952	0.006025	0.013736	0.092097	0.013734	0.007854	0.135945	0.178869	0.231884	0.041762	0.02865	0.026925	0.109432	0.077969	0.047423	0.109071	0.106067	0.075222	0.076932	0.017761	0.007768
	0.1	0.010118	0.009064	0.008237	0.019315	0.008909	0.005649	0.055566	0.015662	0.004015	0.088812	0.10847	0.187995	0.042677	0.037795	0.039793	0.068353	0.025939	0.026871	0.052596	0.039318	0.039217	0.056131	0.016143	0.005171
	0.3	0.003802	0.005102	0.006546	0.015338	0.010836	0.005245	0.021229	0.005111	0.003442	0.044513	0.077182	0.185462	0.011538	0.005095	0.04128	0.044578	0.014165	0.01237	0.044613	0.026697	0.006365	0.03355	0.008012	0.004778
IM	0.01	0.072828	0.058296	0.047763	0.027434	0.01479	0.008826	0.117839	0.021562	0.007727	0.322029	0.35289	0.400296	0.053404	0.026288	0.0549	0.147096	0.085837	0.059884	0.222513	0.132371	0.127319	0.083749	0.02813	0.006439
	0.1	0.054448	0.044021	0.046601	0.032432	0.019717	0.021097	0.089238	0.015083	0.005782	0.299788	0.319411	0.412299	0.052612	0.029347	0.046742	0.101026	0.049305	0.038432	0.141307	0.100046	0.085004	0.074073	0.023826	0.005845
	0.3	0.053786	0.040234	0.040801	0.018659	0.011171	0.013959	0.092324	0.009113	0.004441	0.306245	0.409347	0.424812	0.028305	0.030638	0.056314	0.06415	0.035611	0.033386	0.10081	0.07621	0.063538	0.060668	0.012866	0.00594
SM	0.01	0.040104	0.030048	0.038848	0.024492	0.006761	0.011174	0.078419	0.019908	0.008208	0.300007	0.291012	0.38814	0.044564	0.024495	0.036541	0.088565	0.05913	0.025112	0.087634	0.079764	0.042986	0.092589	0.010043	0.005659
	0.1	0.036254	0.019337	0.015311	0.019977	0.007144	0.005035	0.040991	0.008198	0.006834	0.189537	0.243463	0.297776	0.015298	0.016936	0.041415	0.045629	0.052996	0.012165	0.052977	0.028482	0.017319	0.060397	0.011487	0.007523
	0.3	0.020525	0.014516	0.011256	0.017066	0.002977	0.004899	0.03048	0.004959	0.003333	0.182278	0.205798	0.381506	0.013329	0.006194	0.038519	0.034124	0.011919	0.004217	0.160538	0.008395	0.007656	0.018897	0.008223	0.004803
RSM	0.01	0.047125	0.036846	0.02496	0.031844	0.006512	0.008325	0.080211	0.027683	0.006123	0.300583	0.314024	0.344458	0.060614	0.02429	0.032551	0.101667	0.0474	0.043478	0.139516	0.104324	0.085955	0.088224	0.021833	0.006097
	0.1	0.046136	0.020262	0.016822	0.018522	0.006381	0.011187	0.057239	0.01753	0.00557	0.241419	0.255929	0.328666	0.020267	0.037147	0.02826	0.07005	0.055003	0.023989	0.213776	0.048218	0.034146	0.052554	0.014664	0.004466
	0.3	0.02411	0.014877	0.012329	0.011407	0.011219	0.017373	0.037783	0.007769	0.004768	0.208775	0.225381	0.358497	0.01927	0.015058	0.033203	0.03705	0.020399	0.009886	0.051477	0.010093	0.011836	0.052794	0.007131	0.004348
Zero Flip	0.01	0.087215	0.05472	0.045185	0.037648	0.018688	0.008111	0.142353	0.037512	0.011565	0.383601	0.375744	0.411623	0.053305	0.039612	0.040284	0.179288	0.083502	0.032554	0.257679	0.175381	0.166459	0.123542	0.031096	0.011029
	0.1	0.065729	0.03676	0.033757	0.039383	0.00643	0.015859	0.151073	0.031859	0.011649	0.377815	0.369334	0.42985	0.057585	0.039586	0.057406	0.164408	0.058236	0.039188	0.239599	0.205195	0.166191	0.142432	0.025812	0.006306
	0.3	0.049946	0.029911	0.028983	0.027337	0.011556	0.012619	0.152773	0.032534	0.009297	0.389881	0.361695	0.48521	0.049119	0.039042	0.062084	0.193321	0.075839	0.075299	0.283993	0.173558	0.160815	0.116152	0.044003	0.018945

$M = 500$

		FCX			CGX			HUX			PMFCX			PMGreedyX			1PX			2PX			UX		
		0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1
FCF	0.01	0.020475	0.010559	0.009225	0.01453	0.005291	0.009919	0.062194	0.009998	0.004564	0.151112	0.163702	0.218262	0.033034	0.030689	0.021073	0.098794	0.04521	0.02779	0.070951	0.091135	0.065394	0.059385	0.011385	0.003692
	0.1	0.008212	0.006844	0.006371	0.005679	0.004533	0.005874	0.049326	0.005208	0.004442	0.071043	0.112095	0.179425	0.035103	0.006704	0.036172	0.059139	0.023536	0.012124	0.056533	0.051861	0.03338	0.041245	0.012351	0.004555
	0.3	0.00631	0.005398	0.003073	0.010007	0.008176	0.004908	0.019522	0.004376	0.001711	0.025999	0.034271	0.199827	0.009797	0.008975	0.052119	0.016303	0.017639	0.012945	0.024794	0.012213	0.011256	0.021054	0.003116	0.002115
IM	0.01	0.06774	0.044919	0.05225	0.027789	0.024166	0.008381	0.102362	0.014559	0.00443	0.328962	0.353594	0.403153	0.034496	0.01977	0.029951	0.101622	0.049334	0.03251	0.184508	0.140711	0.113528	0.078274	0.016797	0.004268
	0.1	0.065151	0.183363	0.040919	0.016024	0.004854	0.012046	0.046959	0.010595	0.007687	0.322794	0.30556	0.393455	0.021376	0.019097	0.058493	0.109854	0.052892	0.024495	0.136505	0.078334	0.095181	0.058413	0.010698	0.003392
	0.3	0.036455	0.042269	0.038083	0.009971	0.007391	0.01699	0.051904	0.010901	0.004236	0.261529	0.302238	0.461749	0.013284	0.023031	0.061987	0.182279	0.015597	0.018207	0.079807	0.068632	0.044702	0.043717	0.00991	0.002431
SM	0.01	0.039503	0.026784	0.020462	0.018749	0.005957	0.006454	0.049035	0.013766	0.004468	0.28244	0.31605	0.380613	0.036957	0.019588	0.028524	0.083323	0.047118	0.023324	0.080397	0.070052	0.023302	0.049377	0.014551	0.005929
	0.1	0.017922	0.015606	0.014521	0.013112	0.011118	0.007698	0.050085	0.005061	0.002864	0.194662	0.228502	0.356505	0.014398	0.00657	0.04168	0.057104	0.029718	0.010018	0.029537	0.033579	0.012926	0.019497	0.009651	0.002067
	0.3	0.019753	0.013458	0.012931	0.004209	0.004132	0.006726	0.02087	0.005748	0.002926	0.182581	0.163446	0.379591	0.007792	0.004176	0.028774	0.03063	0.008077	0.005468	0.021368	0.016184	0.003027	0.017709	0.003682	0.005689
RSM	0.01	0.032923	0.027723	0.021467	0.016769	0.005884	0.005373	0.068043	0.010336	0.002724	0.268029	0.317738	0.369886	0.022708	0.008772	0.012408	0.099743	0.04609	0.025921	0.120392	0.060837	0.050157	0.064987	0.01022	0.006359
	0.1	0.030423	0.016997	0.013574	0.01765	0.004682	0.009978	0.034887	0.008025	0.003381	0.23079	0.26732	0.339021	0.025284	0.012938	0.021656	0.067984	0.024197	0.013602	0.090914	0.024799	0.0016	0.039081	0.007906	0.004225
	0.3	0.020695	0.015533	0.012892	0.005814	0.004572	0.006303	0.033939	0.004308	0.00315	0.185638	0.309016	0.346448	0.01198	0.007233	0.034381	0.029897	0.009981	0.00923	0.026097	0.008218	0.016523	0.023391	0.004577	0.001263
Zero Flip	0.01	0.067854	0.050255	0.036997	0.018208	0.0107	0.00409	0.104275	0.024996	0.004	0.382584	0.371652	0.399245	0.036445	0.025655	0.022489	0.142775	0.048067	0.029607	0.238112	0.15829	0.136839	0.087154	0.014282	0.005074
	0.1	0.047058	0.030108	0.025174	0.024113	0.008229	0.006389	0.09897	0.016083	0.007331	0.383312	0.386368	0.471473	0.029757	0.021834	0.03013	0.141261	0.066441	0.034974	0.25334	0.183811	0.131861	0.096704	0.014673	0.005182
	0.3	0.053441	0.027702	0.021518	0.009462	0.008221	0.014384	0.127295	0.024379	0.005728	0.379351	0.389748	0.469001	0.041016	0.025887	0.068191	0.13754	0.072866	0.037038	0.258076	0.169762	0.159319	0.096074	0.013822	0.009756

Table C.2: The normalised results for problem set B

$M = 100$

		FCX			CGX			HUX			PMFCX			PMGreedyX			1PX			2PX			UX		
		0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1
FCF	0.01	0.06094	0.058374	0.060844	0.051797	0.046213	0.040211	0.188095	0.087075	0.064712	0.254593	0.266762	0.356586	0.084506	0.052146	0.047708	0.228759	0.154101	0.139094	0.209029	0.150017	0.143123	0.178936	0.101695	0.057782
	0.1	0.058305	0.052681	0.049289	0.04831	0.043784	0.043445	0.178329	0.088083	0.052877	0.21288	0.250744	0.38466	0.082989	0.049052	0.04426	0.178853	0.146659	0.122207	0.176288	0.129341	0.085585	0.162428	0.095479	0.048772
	0.3	0.045509	0.040351	0.041121	0.051058	0.037964	0.039469	0.095015	0.064251	0.049886	0.135644	0.21037	0.457274	0.06394	0.038309	0.040711	0.132803	0.098842	0.088606	0.120137	0.09336	0.071742	0.109847	0.068786	0.043737
IM	0.01	0.123005	0.108679	0.091793	0.053512	0.049774	0.047593	0.246495	0.141126	0.067001	0.516968	0.512353	0.590941	0.080465	0.049794	0.050146	0.28824	0.201266	0.16699	0.347801	0.218973	0.164846	0.25327	0.143596	0.070591
	0.1	0.101961	0.085979	0.080156	0.055855	0.046961	0.0465	0.23505	0.091899	0.053449	0.487565	0.527413	0.644902	0.081028	0.052901	0.058847	0.228701	0.156082	0.127737	0.251234	0.170294	0.134095	0.211545	0.10809	0.05879
	0.3	0.095712	0.074651	0.075293	0.060374	0.046692	0.044415	0.151083	0.070753	0.045131	0.49152	0.517235	0.605604	0.067265	0.044176	0.04509	0.192222	0.117513	0.099187	0.205482	0.132852	0.092174	0.161775	0.06981	0.04211
SM	0.01	0.09499	0.085474	0.083877	0.062012	0.047194	0.049366	0.198316	0.109212	0.065001	0.466958	0.473648	0.576862	0.081153	0.05952	0.049982	0.232	0.197866	0.161024	0.20276	0.170644	0.133258	0.223314	0.112107	0.065123
	0.1	0.070734	0.0648	0.062164	0.048558	0.043104	0.04701	0.13685	0.081017	0.051751	0.451084	0.468213	0.697967	0.069497	0.049322	0.039139	0.169625	0.130627	0.098169	0.186934	0.110954	0.092061	0.164856	0.084019	0.052233
	0.3	0.0724	0.057442	0.050141	0.044449	0.045276	0.042266	0.11923	0.060011	0.04155	0.447061	0.447029	0.593662	0.056686	0.039668	0.041665	0.142922	0.088912	0.068902	0.148354	0.078153	0.053504	0.142194	0.05403	0.037794
RSM	0.01	0.104075	0.085136	0.073515	0.058278	0.049285	0.043554	0.208251	0.105838	0.061659	0.484631	0.47092	0.514742	0.083489	0.052686	0.041278	0.243066	0.175956	0.159606	0.255511	0.168311	0.136932	0.211527	0.122743	0.064074
	0.1	0.087587	0.072329	0.066437	0.054903	0.04232	0.039997	0.181728	0.080252	0.05737	0.465733	0.506713	0.613258	0.079783	0.049146	0.045935	0.204646	0.134669	0.113455	0.200217	0.141348	0.099274	0.209644	0.085387	0.045377
	0.3	0.076251	0.05687	0.055058	0.050195	0.0516	0.040441	0.139264	0.075129	0.045671	0.447337	0.488663	0.641921	0.066467	0.052752	0.040791	0.165092	0.110898	0.072368	0.150981	0.095405	0.060143	0.130575	0.063139	0.046246
Zero Flip	0.01	0.160032	0.143876	0.109049	0.061147	0.050633	0.047108	0.361896	0.166738	0.073087	0.62185	0.610002	0.66188	0.104615	0.056068	0.049173	0.441285	0.268906	0.172901	0.466119	0.304837	0.228793	0.347755	0.143401	0.078434
	0.1	0.15476	0.122308	0.102305	0.053725	0.053638	0.048873	0.387831	0.173124	0.072557	0.621165	0.627456	0.689623	0.086794	0.063081	0.048983	0.436144	0.286584	0.190873	0.43874	0.28602	0.230521	0.365281	0.16796	0.071495
	0.3	0.151358	0.119875	0.103862	0.060093	0.051955	0.049971	0.378065	0.164053	0.075429	0.621386	0.620382	0.738597	0.096262	0.052554	0.049573	0.451366	0.276101	0.190599	0.477064	0.330155	0.255974	0.379201	0.168581	0.080159

$M = 250$

		FCX			CGX			HUX			PMFCX			PMGreedyX			1PX			2PX			UX		
		0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1
FCF	0.01	0.059544	0.060136	0.06073	0.045929	0.040109	0.041969	0.140134	0.065906	0.042502	0.227067	0.251274	0.380927	0.069258	0.041782	0.042116	0.180564	0.129503	0.091252	0.183309	0.111194	0.094029	0.149787	0.068551	0.041807
	0.1	0.049545	0.048179	0.046785	0.042952	0.040316	0.042149	0.108736	0.055726	0.039078	0.160825	0.231451	0.407785	0.059036	0.038858	0.039877	0.116764	0.088134	0.08439	0.135224	0.086554	0.081253	0.111676	0.051454	0.04216
	0.3	0.041567	0.038006	0.038471	0.038469	0.036639	0.035721	0.091196	0.041657	0.037512	0.128875	0.178431	0.449699	0.041926	0.035597	0.038622	0.103241	0.065597	0.061628	0.080644	0.064271	0.04898	0.084486	0.04331	0.034162
IM	0.01	0.11113	0.086417	0.08043	0.049072	0.040726	0.04496	0.209895	0.086322	0.049767	0.503742	0.51179	0.604638	0.064965	0.04921	0.046819	0.293156	0.177966	0.117482	0.307338	0.192607	0.136165	0.212768	0.081123	0.049121
	0.1	0.100977	0.082626	0.074833	0.04803	0.045599	0.047152	0.149169	0.063011	0.043712	0.485365	0.483186	0.716253	0.068363	0.045976	0.045069	0.199259	0.133602	0.081362	0.239929	0.136435	0.098758	0.142097	0.068791	0.042943
	0.3	0.084147	0.074372	0.072217	0.047437	0.042649	0.044442	0.129319	0.051826	0.040753	0.42552	0.474931	0.590602	0.050937	0.041924	0.044673	0.171957	0.087941	0.073068	0.158931	0.09232	0.074002	0.113435	0.052059	0.038373
SM	0.01	0.090407	0.076091	0.066926	0.052133	0.041972	0.043102	0.153611	0.077598	0.047542	0.478696	0.46913	0.569562	0.062139	0.043173	0.04576	0.202161	0.128276	0.092616	0.206617	0.133127	0.099508	0.150866	0.072243	0.046139
	0.1	0.072567	0.056962	0.055342	0.044437	0.047295	0.045668	0.126058	0.055963	0.03946	0.442	0.461367	0.64819	0.057714	0.039863	0.041471	0.141732	0.095548	0.065264	0.145255	0.083805	0.065092	0.117952	0.053869	0.038785
	0.3	0.065151	0.052477	0.046708	0.039183	0.041327	0.039038	0.09343	0.045866	0.035094	0.413273	0.449753	0.59985	0.044922	0.036899	0.040365	0.117416	0.070868	0.047897	0.083305	0.059574	0.044844	0.095168	0.046811	0.036001
RSM	0.01	0.088075	0.068557	0.077387	0.047683	0.042945	0.045408	0.14783	0.076662	0.043871	0.465516	0.535296	0.586035	0.058986	0.045508	0.043592	0.180251	0.155214	0.102925	0.204223	0.126175	0.099272	0.147355	0.073722	0.046016
	0.1	0.074751	0.055023	0.055948	0.045835	0.03755	0.042597	0.15756	0.053285	0.038899	0.450519	0.459105	0.630609	0.060298	0.04367	0.040899	0.193275	0.123597	0.072444	0.161808	0.08805	0.079526	0.148673	0.065475	0.040261
	0.3	0.060214	0.052023	0.051611	0.045184	0.037269	0.036935	0.093913	0.047246	0.038057	0.419986	0.408023	0.637815	0.041715	0.037475	0.040642	0.132509	0.082176	0.057407	0.119795	0.074428	0.052877	0.098296	0.045587	0.034207
Zero Flip	0.01	0.153433	0.111028	0.103955	0.053857	0.048846	0.045028	0.270486	0.097002	0.04984	0.592771	0.599903	0.759951	0.068075	0.046078	0.046391	0.372245	0.205875	0.120443	0.407938	0.276238	0.187988	0.252372	0.099864	0.05182
	0.1	0.150439	0.102961	0.089204	0.054706	0.047056	0.042189	0.289066	0.09286	0.055046	0.594684	0.592253	0.696733	0.07046	0.046757	0.048971	0.379544	0.198365	0.120923	0.415662	0.243703	0.166624	0.283233	0.095309	0.051596
	0.3	0.144166	0.104016	0.087179	0.050473	0.047121	0.042972	0.327336	0.110147	0.059366	0.627119	0.6181	0.711584	0.066963	0.049628	0.046699	0.41602	0.223504	0.143048	0.419988	0.286773	0.184128	0.308134	0.105857	0.05397

$M = 500$

		FCX			CGX			HUX			PMFCX			PMGreedyX			1PX			2PX			UX		
		0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1
FCF	0.01	0.062323	0.059439	0.052716	0.043308	0.039976	0.039003	0.106704	0.05346	0.036427	0.222744	0.265148	0.47488	0.044824	0.040581	0.043701	0.13887	0.103463	0.065647	0.129615	0.08704	0.06434	0.102983	0.050112	0.038788
	0.1	0.053271	0.049609	0.048368	0.038672	0.038726	0.038247	0.085505	0.045949	0.036286	0.153574	0.203053	0.428968	0.046358	0.038107	0.03851	0.115259	0.07753	0.058458	0.133626	0.066423	0.056516	0.089829	0.043168	0.034695
	0.3	0.04171	0.039154	0.037292	0.034391	0.033572	0.034444	0.061144	0.038125	0.03483	0.106325	0.138775	0.487581	0.039048	0.03431	0.040914	0.081792	0.061762	0.045328	0.081835	0.052554	0.043604	0.062988	0.039509	0.03256
IM	0.01	0.101702	0.07695	0.072391	0.049586	0.04212	0.043392	0.162929	0.065759	0.04187	0.524649	0.507663	0.661374	0.052199	0.045803	0.044976	0.202328	0.124906	0.086233	0.279693	0.148893	0.107619	0.142533	0.06597	0.04101
	0.1	0.09205	0.07066	0.063402	0.045434	0.041616	0.041391	0.117453	0.053449	0.038274	0.459726	0.494599	0.668136	0.049008	0.045124	0.046817	0.201753	0.099018	0.070685	0.172639	0.108689	0.077056	0.12596	0.053432	0.038235
	0.3	0.078477	0.067253	0.056962	0.046794	0.038435	0.039942	0.095381	0.047089	0.036176	0.394753	0.455048	0.592114	0.046788	0.041831	0.043379	0.131073	0.078864	0.057806	0.139031	0.096923	0.06569	0.093875	0.047316	0.035184
SM	0.01	0.078779	0.068037	0.064643	0.044308	0.038605	0.040376	0.118274	0.057724	0.040138	0.482617	0.484494	0.639807	0.051382	0.041789	0.043288	0.199578	0.106763	0.076714	0.162989	0.098559	0.084849	0.12306	0.056219	0.040503
	0.1	0.071465	0.054351	0.04854	0.03956	0.040385	0.037001	0.089042	0.043475	0.035962	0.385175	0.447065	0.630511	0.042699	0.038229	0.043379	0.128293	0.074757	0.051564	0.132516	0.068679	0.049298	0.091495	0.044113	0.036456
	0.3	0.060153	0.045769	0.042387	0.038047	0.037113	0.038956	0.073042	0.040193	0.033596	0.33435	0.425459	0.606615	0.038555	0.038213	0.040627	0.081901	0.052208	0.041063	0.085319	0.048885	0.041041	0.063315	0.038755	0.033078
RSM	0.01	0.079391	0.065955	0.068013	0.043311	0.043179	0.039542	0.13091	0.054961	0.037032	0.443483	0.458286	0.565441	0.051139	0.042461	0.043865	0.200737	0.124331	0.07016	0.168688	0.121307	0.070992	0.122931	0.055673	0.038861
	0.1	0.068135	0.0572	0.050643	0.041296	0.041843	0.039294	0.08533	0.049465	0.035819	0.414506	0.481609	0.627657	0.04867	0.039644	0.039341	0.153317	0.061882	0.063823	0.125605	0.089275	0.055752	0.106926	0.051475	0.036042
	0.3	0.063366	0.051122	0.042002	0.039361	0.036651	0.03758	0.084445	0.038406	0.034488	0.403316	0.422083	0.617437	0.040022	0.037189	0.04063	0.110872	0.065084	0.049104	0.103575	0.059111	0.041535	0.080616	0.041606	0.031794
Zero Flip	0.01	0.141063	0.084421	0.084844	0.051463	0.041434	0.040398	0.214761	0.066102	0.043153	0.600741	0.615827	0.659844	0.052336	0.046085	0.045449	0.31117	0.157265	0.087357	0.342804	0.209412	0.150073	0.195191	0.068698	0.043578
	0.1	0.138672	0.096283	0.079636	0.049423	0.045456	0.041472	0.243122	0.071275	0.044138	0.587138	0.596444	0.818081	0.055085	0.045738	0.047808	0.329567	0.151509	0.097561	0.369095	0.210781	0.157023	0.193873	0.071191	0.042871
	0.3	0.136103	0.094147	0.08501	0.047962	0.043017	0.042122	0.213191	0.077321	0.044666	0.626689	0.621004	0.831675	0.059806	0.044119	0.046971	0.341528	0.16605	0.097994	0.397386	0.213265	0.170742	0.220093	0.077393	0.047139

Table C.3: The normalised results for problem set C

$M = 100$

		FCX			CGX			HUX			PMFCX		PMGreedyX			1PX			2PX			UX		
		0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1
FCF	0.01	0.011395	0.023038	0.015206	0.042285	0.035443	0.051806	0.064581	0.054219	0.038187	0.049601	0.066723	0.104449	0.032936	0.047833	0.017885	0.031922	0.023226	0.059763	0.03164	0.056216	0.01127	0.027833	0.030443	0.033907
	0.1	0.021819	0.004693	0.00697	0.085095	0.085387	0.04275	0.026018	0.056711	0.05265	0.02167	0.037245	0.046115	0.040843	0.054798	0.059383	0.025539	0.024502	0.017845	0.030101	0.024547	0.02457	0.056872	0.030809	0.031689
	0.3	0.008951	0.015101	0.011826	0.089064	0.017786	0.047701	0.022846	0.017613	0.008024	0.052911	0.038594	0.051608	0.016282	0.01154	0.025014	0.023621	0.050021	0.021215	0.025612	0.04564	0.02626	0.060195	0.018706	0.017516
IM	0.01	0.069865	0.056382	0.042914	0.077902	0.036916	0.026072	0.069164	0.067497	0.052876	0.179609	0.166451	0.181378	0.026744	0.056044	0.03534	0.129632	0.070546	0.077333	0.154883	0.08568	0.107468	0.058526	0.054496	0.012081
	0.1	0.047341	0.050744	0.103801	0.049901	0.062356	0.092864	0.051188	0.022404	0.019866	0.126966	0.117062	0.162503	0.069476	0.02592	0.058476	0.141916	0.138492	0.056894	0.108161	0.065275	0.075846	0.049992	0.019779	0.015189
	0.3	0.043475	0.060173	0.028717	0.033251	0.035144	0.029609	0.068097	0.026965	0.018748	0.12911	0.101512	0.163227	0.031482	0.031224	0.05524	0.145175	0.055693	0.073459	0.107627	0.13889	0.117335	0.048357	0.032942	0.071368
SM	0.01	0.053457	0.041787	0.06505	0.029953	0.064843	0.048098	0.067349	0.020697	0.042418	0.147107	0.16993	0.146535	0.013767	0.017761	0.02053	0.045679	0.038459	0.04314	0.105047	0.052913	0.122746	0.052305	0.045942	0.025917
	0.1	0.044273	0.064145	0.065912	0.020596	0.043841	0.026281	0.057791	0.056584	0.025284	0.166385	0.104062	0.20909	0.110466	0.057818	0.058153	0.101626	0.049961	0.132925	0.092235	0.071991	0.049385	0.131238	0.02374	0.034756
	0.3	0.042356	0.150628	0.0709	0.135484	0.037132	0.026259	0.039762	0.0325	0.018428	0.107301	0.089215	0.163947	0.029051	0.022744	0.014882	0.06207	0.210784	0.031896	0.162518	0.055738	0.154671	0.035305	0.095239	0.160537
RSM	0.01	0.044074	0.033295	0.038151	0.031594	0.024157	0.016683	0.06317	0.033505	0.084827	0.125938	0.116401	0.164092	0.050905	0.026242	0.016288	0.062214	0.102491	0.068866	0.100373	0.133732	0.067477	0.052124	0.022727	0.026042
	0.1	0.039897	0.079741	0.0589	0.035653	0.05557	0.12708	0.177438	0.238049	0.012478	0.109265	0.110309	0.163578	0.021575	0.02279	0.034679	0.093231	0.053981	0.023674	0.116561	0.092654	0.064058	0.120568	0.019583	0.05383
	0.3	0.065791	0.018103	0.01214	0.031307	0.011284	0.034337	0.033777	0.033505	0.016273	0.078611	0.084129	0.171878	0.050791	0.03503	0.026925	0.06119	0.062083	0.057504	0.176812	0.068403	0.027731	0.064853	0.176488	0.017509
Zero Flip	0.01	0.049027	0.039627	0.037702	0.03064	0.021738	0.056484	0.086257	0.039858	0.029279	0.178695	0.138459	0.152309	0.042469	0.021134	0.017637	0.155905	0.068518	0.058413	0.117997	0.088209	0.119861	0.059476	0.025235	0.01843
	0.1	0.042497	0.030191	0.0301	0.032771	0.025451	0.097372	0.063914	0.034635	0.02614	0.136594	0.133333	0.165854	0.034837	0.039257	0.101695	0.108296	0.068615	0.05565	0.136174	0.081753	0.103827	0.062068	0.038624	0.046616
	0.3	0.052663	0.074608	0.051101	0.064156	0.073111	0.0594	0.06389	0.036316	0.013356	0.168542	0.128861	0.179545	0.032658	0.021993	0.01947	0.098607	0.057941	0.062195	0.118678	0.092519	0.099089	0.060565	0.023995	0.01384

$M = 250$

		FCX			CGX			HUX			PMFCX		PMGreedyX			1PX			2PX			UX		
		0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1
FCF	0.01	0.040969	0.077973	0.115367	0.049166	0.084368	0.056382	0.021293	0.051012	0.050728	0.032026	0.046323	0.079704	0.029349	0.048088	0.054182	0.032267	0.079635	0.050879	0.070876	0.044493	0.058665	0.035091	0.053494	0.008859
	0.1	0.044225	0.017762	0.007867	0.01465	0.043534	0.153275	0.017593	0.01	0.057129	0.039797	0.080129	0.074331	0.051199	0.098751	0.053119	0.053977	0.022443	0.124933	0.02497	0.020012	0.029243	0.035631	0.003913	0.113024
	0.3	0.052419	0.081937	0.007531	0.011621	0.119264	0.039912	0.015529	0.053448	0.045922	0.050272	0.021988	0.048379	0.019016	0.005506	0.040168	0.011536	0.077903	0.043282	0.015628	0.089308	0.009073	0.01175	0.003224	0.011382
IM	0.01	0.104769	0.046471	0.038903	0.093403	0.12138	0.056058	0.181934	0.027387	0.057134	0.136292	0.171517	0.204579	0.052021	0.064648	0.051094	0.083337	0.052171	0.089565	0.10012	0.074422	0.054424	0.06064	0.057003	0.043666
	0.1	0.050863	0.065939	0.064448	0.122425	0.050088	0.051376	0.050266	0.091179	0.01267	0.146075	0.142359	0.213954	0.026848	0.086662	0.046437	0.106729	0.085552	0.071634	0.087584	0.131727	0.084736	0.075109	0.024016	0.048857
	0.3	0.033517	0.065388	0.099903	0.019526	0.04226	0.050366	0.047106	0.016788	0.012639	0.116165	0.13608	0.177812	0.022852	0.019506	0.056741	0.070264	0.073265	0.064684	0.083314	0.051638	0.110229	0.048513	0.022416	0.04915
SM	0.01	0.081528	0.045931	0.095985	0.066127	0.112425	0.080887	0.051197	0.016284	0.064456	0.306751	0.133073	0.148771	0.027258	0.064151	0.052314	0.067722	0.046642	0.069717	0.125186	0.116246	0.111073	0.11716	0.12054	0.016218
	0.1	0.033531	0.096313	0.037859	0.014774	0.053299	0.080129	0.057981	0.037275	0.009256	0.342322	0.114309	0.199055	0.017801	0.042662	0.014125	0.057522	0.09119	0.023706	0.049755	0.091902	0.078532	0.060015	0.051008	0.057875
	0.3	0.069269	0.052783	0.091767	0.013323	0.078121	0.050861	0.182924	0.182954	0.011613	0.136431	0.130323	0.177272	0.029254	0.05546	0.082903	0.084408	0.023908	0.013685	0.109238	0.034166	0.02112	0.039295	0.012298	0.090677
RSM	0.01	0.07783	0.066341	0.099132	0.019358	0.057284	0.127517	0.071789	0.059324	0.052308	0.105043	0.1154	0.12869	0.088339	0.048115	0.020374	0.061268	0.041301	0.045655	0.083134	0.060166	0.050695	0.043307	0.061413	0.02145
	0.1	0.035427	0.046769	0.055234	0.051482	0.059037	0.005917	0.099442	0.01948	0.005671	0.104253	0.205207	0.180805	0.016618	0.082156	0.077981	0.111094	0.084133	0.020447	0.136369	0.051307	0.029072	0.039961	0.057131	0.043281
	0.3	0.04652	0.027819	0.049467	0.096499	0.008733	0.112888	0.051764	0.192171	0.042479	0.102212	0.210995	0.17811	0.021475	0.014413	0.076599	0.124957	0.062875	0.089496	0.100283	0.040305	0.056366	0.069798	0.036687	0.044774
Zero Flip	0.01	0.058324	0.076819	0.066399	0.054922	0.152678	0.041568	0.049347	0.027895	0.084425	0.136534	0.177892	0.177516	0.026554	0.056527	0.033243	0.084483	0.061302	0.075383	0.095591	0.097005	0.062208	0.104929	0.023269	0.005278
	0.1	0.111508	0.068565	0.024865	0.032595	0.088499	0.097968	0.067888	0.025578	0.045302	0.131682	0.123414	0.152691	0.024432	0.01637	0.015768	0.099105	0.050172	0.068851	0.089085	0.112831	0.100565	0.078054	0.01965	0.045227
	0.3	0.081621	0.05875	0.059277	0.010598	0.009824	0.046772	0.067863	0.017296	0.01463	0.180326	0.126434	0.254567	0.036894	0.054708	0.020202	0.141845	0.088409	0.071674	0.082774	0.102293	0.084077	0.068357	0.031633	0.054416

$M = 500$

		FCX			CGX			HUX			PMFCX		PMGreedyX			1PX			2PX			UX		
		0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1
FCF	0.01	0.053656	0.047108	0.121987	0.103873	0.017832	0.117428	0.057113	0.056327	0.07776	0.117916	0.068265	0.108031	0.017183	0.036964	0.11203	0.069023	0.096442	0.117209	0.103763	0.057518	0.129997	0.018209	0.0535	0.038862
	0.1	0.008409	0.045808	0.008742	0.082501	0.078581	0.084765	0.044529	0.041515	0.145337	0.08545	0.039577	0.046228	0.048391	0.121373	0.042779	0.081746	0.114542	0.009341	0.052154	0.118103	0.091516	0.009864	0.080003	0.037492
	0.3	0.119621	0.076988	0.042289	0.01077	0.081283	0.150904	0.054887	0.045203	0.006426	0.087788	0.026616	0.108458	0.145818	0.008176	0.077711	0.046003	0.078656	0.045437	0.006221	0.150093	0.078726	0.054109	0.080858	0.074259
IM	0.01	0.103217	0.046088	0.072295	0.082489	0.087984	0.116062	0.03876	0.013494	0.004229	0.180552	0.173475	0.178948	0.09703	0.07899	0.08388	0.114635	0.122462	0.1106	0.14088	0.138583	0.171015	0.036497	0.125557	0.016312
	0.1	0.079036	0.13888	0.099871	0.059403	0.045349	0.15187	0.039209	0.015779	0.047746	0.168878	0.15073	0.217195	0.025891	0.083547	0.083996	0.093868	0.046932	0.104909	0.135505	0.133525	0.125553	0.159823	0.061726	0.046181
	0.3	0.03294	0.078161	0.134739	0.014992	0.07958	0.118077	0.07306	0.053018	0.010951	0.177552	0.126702	0.183493	0.066414	0.009082	0.017067	0.097688	0.03849	0.097046	0.071143	0.131338	0.04018	0.199607	0.010335	0.043852
SM	0.01	0.07068	0.062594	0.061939	0.051538	0.044657	0.114753	0.029991	0.051078	0.080999	0.120292	0.127487	0.206509	0.092871	0.043091	0.004031	0.069958	0.071696	0.139776	0.158327	0.066529	0.082063	0.06567	0.01242	0.046661
	0.1	0.098778	0.135511	0.059757	0.011605	0.045004	0.086106	0.11821	0.047022	0.082063	0.154111	0.12452	0.268976	0.045947	0.045879	0.114613	0.041938	0.066065	0.089276	0.126866	0.068821	0.06298	0.233847	0.015736	0.047359
	0.3	0.085391	0.082405	0.080972	0.079613	0.04566	0.112495	0.022593	0.043819	0.041636	0.129469	0.102351	0.223444	0.046151	0.005454	0.074518	0.135486	0.078183	0.055095	0.131327	0.085056	0.106025	0.095686	0.020698	0.111411
RSM	0.01	0.041473	0.136336	0.130023	0.05627	0.116105	0.114941	0.039112	0.019943	0.078203	0.121114	0.117153	0.207378	0.053664	0.077161	0.081426	0.058655	0.076816	0.095898	0.118948	0.04338	0.128932	0.029458	0.050907	0.112525
	0.1	0.034462	0.057767	0.061246	0.077455	0.113909	0.113682	0.033268	0.081337	0.042354	0.095711	0.165508	0.266175	0.046425	0.114812	0.006711	0.085985	0.087848	0.0509	0.064015	0.10281	0.053872	0.022956	0.014647	0.048961
	0.3	0.056904	0.015292	0.081279	0.011434	0.076417	0.04378	0.170264	0.085528	0.073883	0.103893	0.184223	0.193509	0.051973	0.005451	0.005254	0.05547	0.221578	0.008605	0.087934	0.065144	0.071403	0.030774	0.048336	0.004516
Zero Flip	0.01	0.100035	0.146434	0.10386	0.08645	0.082364	0.150606	0.077361	0.089417	0.051616	0.178811	0.163481	0.195275	0.014384	0.044907	0.045396	0.094617	0.044068	0.064972	0.094437	0.127892	0.080632	0.041953	0.013195	0.116145
	0.1	0.126713	0.060862	0.096566	0.09662	0.042662	0.152614	0.036329	0.051668	0.113606	0.158763	0.165775	0.224974	0.024127	0.007573	0.079839	0.105628	0.089095	0.109383	0.083919	0.089427	0.062408	0.066527	0.012661	0.043047
	0.3	0.039589	0.020385	0.136226	0.007527	0.078838	0.113529	0.172327	0.041224	0.045206	0.142481	0.150394	0.193735	0.030306	0.08639	0.008509	0.08664	0.053287	0.075584	0.090907	0.137272	0.055637	0.116336	0.015131	0.083602

Table C.4: The normalised results for problem set D

$M = 100$

		FCX			CGX			HUX			PMFCX			PMGreedyX			1PX			2PX			UX		
		0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1
FCF	0.01	0.143832	0.028361	0.011014	0.374952	0.637454	0.727777	0.377416	0.343397	0.294898	0.414801	0.416152	0.671293	0.178689	0.051893	0.027206	0.406106	0.368534	0.351932	0.381446	0.365567	0.343041	0.400112	0.336909	0.294272
	0.1	0.122799	0.024949	0.008261	0.350876	0.680172	0.727777	0.376118	0.34259	0.29867	0.386115	0.399338	0.55635	0.157108	0.054101	0.029108	0.379914	0.361163	0.345499	0.375352	0.362564	0.340033	0.376407	0.336818	0.292456
	0.3	0.122623	0.026258	0.008102	0.387723	0.638553	0.727777	0.36702	0.336931	0.2962	0.380952	0.395124	0.535343	0.148959	0.053932	0.029643	0.373776	0.360849	0.348831	0.371048	0.35557	0.340824	0.371372	0.338018	0.291171
IM	0.01	0.142495	0.028829	0.009785	0.311491	0.637769	0.727777	0.459929	0.339593	0.296656	0.501062	0.547782	0.645441	0.156951	0.056801	0.029705	0.424551	0.38241	0.35537	0.47532	0.370596	0.34312	0.462325	0.359728	0.299872
	0.1	0.139739	0.032764	0.018584	0.39374	0.65893	0.727777	0.385745	0.33953	0.305402	0.404282	0.410093	0.670391	0.153019	0.052068	0.035738	0.39317	0.369508	0.355749	0.394431	0.365695	0.345781	0.378905	0.344237	0.31146
	0.3	0.175844	0.063985	0.041542	0.487732	0.680244	0.727777	0.386779	0.363709	0.340143	0.410265	0.440737	0.588893	0.176488	0.065535	0.04099	0.379534	0.370373	0.360932	0.394201	0.368902	0.353581	0.398672	0.35817	0.337824
SM	0.01	0.135086	0.025455	0.009257	0.347527	0.679997	0.727777	0.480298	0.368705	0.299523	0.512307	0.472204	0.664823	0.160805	0.055527	0.032867	0.394474	0.380988	0.35391	0.457537	0.381493	0.337359	0.427227	0.36179	0.295583
	0.1	0.137349	0.024931	0.008031	0.410077	0.658536	0.727777	0.386888	0.338478	0.297764	0.418813	0.416067	0.573472	0.154665	0.051973	0.034981	0.389979	0.361896	0.352534	0.380131	0.361968	0.342419	0.404976	0.338512	0.294864
	0.3	0.13039	0.030166	0.011285	0.305035	0.703773	0.727777	0.373605	0.338974	0.3029	0.384683	0.413153	0.511885	0.146672	0.054535	0.028486	0.374565	0.358171	0.345282	0.377721	0.357848	0.339634	0.374571	0.3371	0.303376
RSM	0.01	0.139087	0.026561	0.009517	0.370528	0.635434	0.727777	0.421016	0.360049	0.296511	0.486083	0.447302	0.653066	0.159771	0.05083	0.029262	0.406371	0.369233	0.353279	0.41256	0.385202	0.342106	0.39892	0.340751	0.295388
	0.1	0.123283	0.029514	0.0094	0.38105	0.659016	0.727777	0.374057	0.33706	0.299688	0.388558	0.413504	0.565988	0.159518	0.051209	0.033013	0.376894	0.368243	0.349158	0.379102	0.363408	0.340527	0.375012	0.338493	0.296813
	0.3	0.133843	0.023937	0.009845	0.330101	0.636755	0.727777	0.374729	0.338674	0.299973	0.382661	0.400874	0.505847	0.147161	0.052717	0.032506	0.374396	0.361901	0.349827	0.372078	0.357745	0.338126	0.371863	0.331947	0.300252
Zero Flip	0.01	0.129355	0.026749	0.007555	0.346505	0.659061	0.727777	0.411771	0.344478	0.296782	0.43015	0.438092	0.669861	0.168163	0.053755	0.034356	0.396949	0.369205	0.352591	0.392784	0.386694	0.343218	0.399722	0.340513	0.293817
	0.1	0.132961	0.026283	0.014063	0.361053	0.65872	0.727777	0.378738	0.337724	0.300275	0.408932	0.396844	0.591567	0.156607	0.050836	0.028499	0.380506	0.364648	0.354672	0.379733	0.365157	0.34294	0.378528	0.3358	0.295629
	0.3	0.120927	0.025246	0.008867	0.365177	0.63772	0.727777	0.373481	0.340154	0.297731	0.395294	0.407587	0.526325	0.147893	0.051859	0.03239	0.376537	0.366506	0.352776	0.378657	0.3629	0.343412	0.374202	0.34168	0.298296

$M = 250$

		FCX			CGX			HUX			PMFCX			PMGreedyX			1PX			2PX			UX		
		0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1
FCF	0.01	0.103971	0.034062	0.029416	0.506547	0.727777	0.727777	0.376565	0.354879	0.339648	0.429905	0.505687	0.696066	0.155323	0.09251	0.101278	0.380548	0.36438	0.356188	0.384818	0.366724	0.354293	0.382053	0.355893	0.342109
	0.1	0.096791	0.029784	0.027388	0.507077	0.727777	0.727777	0.372068	0.353741	0.342338	0.396264	0.408152	0.692122	0.155592	0.093904	0.105727	0.37733	0.363391	0.358554	0.376944	0.362751	0.354142	0.372233	0.355493	0.342083
	0.3	0.10653	0.027178	0.024062	0.464097	0.727777	0.727777	0.371477	0.350729	0.340002	0.381924	0.404132	0.582248	0.147284	0.092599	0.10353	0.374674	0.363849	0.354672	0.373334	0.365878	0.355889	0.370729	0.354173	0.342728
IM	0.01	0.110675	0.032833	0.028657	0.508353	0.62309	0.727777	0.439928	0.357934	0.342714	0.528227	0.576832	0.711142	0.156432	0.09098	0.104762	0.415647	0.365288	0.356111	0.441612	0.366469	0.354288	0.408593	0.367987	0.34284
	0.1	0.116708	0.045528	0.033432	0.444737	0.727777	0.727777	0.380905	0.362469	0.348942	0.409506	0.449872	0.716061	0.163464	0.097671	0.105423	0.378567	0.367749	0.358998	0.381123	0.366415	0.35811	0.397437	0.367863	0.347814
	0.3	0.159979	0.073257	0.064779	0.551206	0.727777	0.727777	0.400018	0.371295	0.364437	0.405113	0.632155	0.726053	0.181511	0.111249	0.109014	0.40108	0.372845	0.365332	0.3854	0.373917	0.384868	0.382259	0.37132	0.365497
SM	0.01	0.107998	0.035354	0.028666	0.489749	0.727777	0.727777	0.395146	0.357325	0.343261	0.509929	0.502038	0.71306	0.157376	0.092658	0.106373	0.400583	0.364263	0.353709	0.38389	0.365062	0.35725	0.477701	0.375536	0.342914
	0.1	0.114437	0.036041	0.02673	0.507962	0.727777	0.727777	0.374524	0.357606	0.341633	0.420843	0.424803	0.670233	0.159403	0.101145	0.101551	0.378632	0.36358	0.357676	0.380488	0.364648	0.355338	0.383842	0.355852	0.341561
	0.3	0.118275	0.043939	0.03914	0.468665	0.727777	0.727777	0.37317	0.398445	0.346637	0.384987	0.41542	0.580402	0.152948	0.098943	0.106336	0.377005	0.366954	0.358031	0.388306	0.364661	0.354429	0.373867	0.356719	0.343432
RSM	0.01	0.110122	0.033749	0.032178	0.463679	0.727777	0.727777	0.446134	0.357837	0.343616	0.436961	0.528138	0.694583	0.151196	0.094169	0.104777	0.415883	0.366273	0.357572	0.381789	0.364739	0.372008	0.40875	0.357528	0.342076
	0.1	0.10963	0.03165	0.026788	0.46574	0.727777	0.727777	0.374254	0.355876	0.342609	0.392664	0.41539	0.636523	0.149124	0.099205	0.107357	0.376913	0.364363	0.357156	0.385005	0.364776	0.355082	0.376118	0.356215	0.34131
	0.3	0.107189	0.035106	0.031065	0.443711	0.727777	0.727777	0.372106	0.353449	0.345878	0.381843	0.426215	0.552378	0.153023	0.092968	0.105271	0.376257	0.364487	0.35819	0.37542	0.364201	0.35767	0.373044	0.353891	0.345753
Zero Flip	0.01	0.107915	0.032215	0.021461	0.463028	0.727777	0.727777	0.404531	0.356482	0.341262	0.444161	0.561632	0.692331	0.161893	0.094366	0.101389	0.379671	0.364403	0.355547	0.392452	0.364877	0.355404	0.377449	0.357792	0.339262
	0.1	0.105925	0.03038	0.023876	0.464157	0.727777	0.727777	0.385577	0.371948	0.34217	0.386765	0.407253	0.704515	0.159466	0.095464	0.100408	0.378316	0.363624	0.356586	0.384711	0.365473	0.35732	0.377529	0.354864	0.343698
	0.3	0.100346	0.036733	0.036399	0.487572	0.727777	0.727777	0.37309	0.354881	0.341576	0.401591	0.389962	0.571642	0.149933	0.093027	0.108032	0.374339	0.36643	0.356685	0.373133	0.365057	0.356573	0.374728	0.353506	0.343232

$M = 500$

		FCX			CGX			HUX			PMFCX			PMGreedyX			1PX			2PX			UX		
		0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1
FCF	0.01	0.189919	0.145398	0.175011	0.681854	0.727777	0.727777	0.468083	0.372809	0.367442	0.492638	0.660524	0.727212	0.225083	0.188239	0.215147	0.400307	0.396046	0.373038	0.419646	0.395152	0.389217	0.402444	0.408245	0.385381
	0.1	0.189679	0.144725	0.177991	0.66344	0.727777	0.727777	0.382227	0.405758	0.383559	0.390617	0.559196	0.677622	0.219853	0.191014	0.212144	0.385247	0.374772	0.370302	0.382852	0.39469	0.407166	0.398384	0.390681	0.386642
	0.3	0.187749	0.142281	0.17662	0.704131	0.727777	0.727777	0.378549	0.371	0.366877	0.402766	0.389431	0.642749	0.221499	0.188586	0.214051	0.380627	0.374549	0.372484	0.397452	0.392588	0.38875	0.396732	0.390474	0.367427
IM	0.01	0.186966	0.145268	0.17755	0.724427	0.727777	0.727777	0.417725	0.389978	0.383673	0.57856	0.710467	0.703383	0.224756	0.185472	0.213061	0.403367	0.376742	0.370994	0.4219	0.39624	0.389067	0.435822	0.392704	0.388133
	0.1	0.205289	0.148805	0.176078	0.662619	0.727777	0.727777	0.433989	0.390032	0.422388	0.511276	0.674676	0.727224	0.232309	0.19081	0.216908	0.384688	0.395394	0.391254	0.402289	0.427938	0.40807	0.451868	0.394438	0.388393
	0.3	0.231487	0.170606	0.170857	0.724429	0.727777	0.727777	0.413447	0.397521	0.41156	0.44515	0.643881	0.687675	0.238758	0.198033	0.217963	0.436596	0.426335	0.377311	0.406501	0.431873	0.395814	0.437925	0.380945	0.377759
SM	0.01	0.195202	0.144455	0.17744	0.681611	0.727777	0.727777	0.438063	0.389783	0.365789	0.578798	0.662432	0.712337	0.22337	0.188109	0.214402	0.41431	0.395486	0.390421	0.470824	0.379366	0.389604	0.437052	0.393387	0.368436
	0.1	0.186804	0.145364	0.180311	0.702899	0.727777	0.727777	0.400128	0.374678	0.369909	0.473663	0.510885	0.694658	0.223281	0.189706	0.215099	0.399054	0.377248	0.372166	0.418635	0.427391	0.372077	0.418862	0.391847	0.369408
	0.3	0.20358	0.151081	0.167152	0.660509	0.727777	0.727777	0.414577	0.374308	0.368886	0.422187	0.475645	0.641729	0.227171	0.187979	0.215344	0.417126	0.396213	0.389771	0.43284	0.409615	0.514238	0.416574	0.375565	0.368364
RSM	0.01	0.191923	0.143504	0.17752	0.727777	0.727777	0.727777	0.41932	0.392844	0.385194	0.577228	0.645185	0.736988	0.223753	0.184904	0.216415	0.403615	0.375438	0.370266	0.455091	0.412606	0.373082	0.451387	0.391166	0.386188
	0.1	0.202134	0.14836	0.171387	0.683259	0.727777	0.727777	0.39562	0.372055	0.384908	0.404027	0.508437	0.710647	0.219376	0.192722	0.213617	0.398893	0.387731	0.390759	0.416455	0.376982	0.370725	0.418155	0.375234	0.368465
	0.3	0.198096	0.145665	0.176384	0.660448	0.727777	0.727777	0.397549	0.372906	0.366163	0.385564	0.450605	0.569059	0.219754	0.189254	0.215737	0.380611	0.373504	0.372161	0.37988	0.392269	0.372375	0.379905	0.370313	0.37089
Zero Flip	0.01	0.180774	0.147547	0.1742	0.704304	0.727777	0.727777	0.436241	0.407524	0.402175	0.509313	0.638573	0.727222	0.227404	0.188248	0.213113	0.427376	0.37595	0.3706	0.403853	0.377264	0.370429	0.418743	0.406918	0.369452
	0.1	0.189826	0.14678	0.173381	0.70336	0.727777	0.727777	0.398472	0.373636	0.382668	0.407116	0.468159	0.708495	0.21369	0.186871	0.21648	0.399101	0.375478	0.389938	0.384202	0.37644	0.371978	0.398038	0.392614	0.386309
	0.3	0.189934	0.144745	0.171803	0.725581	0.727777	0.727777	0.395641	0.390391	0.385152	0.421408	0.524231	0.611038	0.225715	0.186005	0.213627	0.380451	0.375819	0.371893	0.381125	0.411352	0.390056	0.397459	0.39028	0.369033

Table C.5: The normalised results for problem set E

$M = 100$

		FCX			CGX			HUX			PMFCX			PMGreedyX			1PX			2PX			UX		
		0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1
FCF	0.01	0.02116	0.003834	0.002754	0.167103	0.68779	0.916848	0.286212	0.208776	0.160898	0.333014	0.359595	0.861094	0.055069	0.016719	0.013452	0.313032	0.282515	0.25888	0.312327	0.274341	0.246673	0.287443	0.209455	0.159093
	0.1	0.018922	0.003836	0.003086	0.19577	0.676504	0.916452	0.278533	0.205187	0.159378	0.322035	0.32847	0.598903	0.054657	0.016193	0.012186	0.302784	0.272214	0.251315	0.301403	0.26688	0.243363	0.282917	0.201427	0.162379
	0.3	0.021457	0.004163	0.002877	0.167911	0.687542	0.888846	0.27506	0.20729	0.160715	0.315103	0.325431	0.47716	0.052082	0.016778	0.012795	0.296586	0.268584	0.24871	0.295077	0.263216	0.23476	0.274913	0.201903	0.159925
IM	0.01	0.020593	0.004404	0.002791	0.185842	0.687925	0.91668	0.311834	0.20794	0.162346	0.4807	0.550167	0.905879	0.058601	0.016543	0.011818	0.315155	0.280724	0.260318	0.311401	0.272414	0.247299	0.283733	0.207161	0.163449
	0.1	0.025283	0.003881	0.002954	0.169719	0.65748	0.916506	0.287187	0.211613	0.173404	0.327679	0.335749	0.808646	0.052853	0.018345	0.012702	0.309825	0.280285	0.260618	0.304644	0.270986	0.245135	0.283517	0.212816	0.172089
	0.3	0.033092	0.007235	0.004686	0.284056	0.65942	0.888679	0.289706	0.241747	0.215308	0.327512	0.358858	0.813384	0.057202	0.019878	0.015153	0.306965	0.284449	0.265338	0.30724	0.27575	0.255011	0.292344	0.240633	0.217683
SM	0.01	0.021482	0.004339	0.003019	0.221399	0.626725	0.888998	0.316496	0.206658	0.15957	0.382982	0.359255	0.883593	0.056229	0.016922	0.01208	0.310883	0.282423	0.258051	0.311908	0.273528	0.245219	0.289097	0.207639	0.157496
	0.1	0.021835	0.003216	0.002581	0.192809	0.628536	0.887273	0.285815	0.206763	0.160338	0.329046	0.332301	0.627615	0.057699	0.017312	0.011384	0.307091	0.268326	0.249165	0.301822	0.265394	0.235889	0.286185	0.204385	0.158996
	0.3	0.021225	0.003483	0.003318	0.198626	0.627677	0.888379	0.281232	0.210129	0.168135	0.323624	0.329885	0.692508	0.050096	0.016409	0.011478	0.301716	0.266299	0.241349	0.297508	0.259204	0.235445	0.279124	0.210131	0.165448
RSM	0.01	0.020293	0.004332	0.002637	0.185271	0.687704	0.889346	0.284338	0.207137	0.156636	0.357221	0.360821	0.792377	0.058121	0.016241	0.011826	0.313395	0.285678	0.259064	0.310991	0.271264	0.247097	0.289053	0.206947	0.15095
	0.1	0.020289	0.004027	0.002727	0.168162	0.626724	0.91679	0.286631	0.207529	0.159697	0.330282	0.328284	0.574089	0.05433	0.016363	0.012431	0.305822	0.275374	0.25248	0.306435	0.266403	0.243663	0.283842	0.205277	0.161112
	0.3	0.017866	0.004302	0.002807	0.226499	0.657488	0.88918	0.280767	0.201866	0.161207	0.323288	0.328953	0.480968	0.053098	0.01693	0.011687	0.300316	0.270487	0.250079	0.296919	0.262826	0.235223	0.27736	0.203006	0.16444
Zero Flip	0.01	0.022138	0.004328	0.002779	0.250736	0.717558	0.888561	0.288834	0.20319	0.157867	0.336848	0.505839	0.931373	0.058944	0.016729	0.011317	0.33782	0.285598	0.258323	0.314435	0.275063	0.247618	0.285377	0.211187	0.160738
	0.1	0.022281	0.003675	0.002749	0.193824	0.687184	0.916103	0.291778	0.210023	0.162236	0.358264	0.35918	0.883433	0.050021	0.017263	0.0123	0.315831	0.282862	0.258375	0.312252	0.272658	0.246194	0.295472	0.209214	0.162672
	0.3	0.02212	0.004576	0.002948	0.18759	0.687916	0.888458	0.29519	0.212776	0.167791	0.336468	0.385964	0.765892	0.057949	0.017035	0.013211	0.318133	0.286741	0.26316	0.316306	0.277229	0.247225	0.295462	0.212869	0.167855

$M = 250$

		FCX			CGX			HUX			PMFCX			PMGreedyX			1PX			2PX			UX		
		0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1
FCF	0.01	0.022013	0.008418	0.008073	0.242895	1	1	0.290487	0.25629	0.236884	0.354023	0.456009	0.953866	0.056991	0.039056	0.043059	0.307096	0.281274	0.272368	0.30621	0.281392	0.264793	0.29051	0.254949	0.239407
	0.1	0.018929	0.007988	0.009124	0.417172	1	1	0.289647	0.254378	0.236972	0.32786	0.3322	0.71535	0.057186	0.038946	0.042444	0.305506	0.284081	0.268586	0.304753	0.279305	0.266264	0.28804	0.254662	0.237017
	0.3	0.021326	0.008506	0.008902	0.329461	1	1	0.286693	0.25467	0.236688	0.324923	0.329888	0.525614	0.059103	0.037268	0.043707	0.303311	0.282241	0.270117	0.301278	0.279526	0.263724	0.289681	0.253859	0.236548
IM	0.01	0.020823	0.009475	0.008126	0.424504	1	1	0.290206	0.254995	0.238591	0.450758	0.761436	1	0.058027	0.037688	0.041916	0.306883	0.282847	0.268262	0.303567	0.281456	0.26741	0.3115	0.256487	0.2378
	0.1	0.025241	0.009548	0.009417	0.361899	1	1	0.293338	0.260958	0.24648	0.32794	0.428415	0.950781	0.061109	0.039135	0.041415	0.309359	0.287091	0.273535	0.307023	0.280705	0.271554	0.294196	0.26083	0.243806
	0.3	0.034259	0.012441	0.012008	0.423184	0.916612	1	0.304403	0.275903	0.264338	0.327533	0.404782	0.90609	0.075006	0.044921	0.050933	0.311942	0.297807	0.282507	0.311448	0.291923	0.279519	0.302776	0.278213	0.268026
SM	0.01	0.020037	0.010004	0.009518	0.353169	0.971775	1	0.288949	0.254852	0.235933	0.378954	0.643209	0.950171	0.055754	0.038287	0.042226	0.331903	0.284725	0.270973	0.305017	0.277432	0.265573	0.290448	0.255671	0.236124
	0.1	0.022342	0.00885	0.008963	0.421267	1	1	0.291046	0.258697	0.235732	0.327771	0.354242	0.740755	0.060544	0.038765	0.04175	0.305231	0.283565	0.269781	0.304031	0.279476	0.26598	0.290638	0.256038	0.237964
	0.3	0.024559	0.009391	0.009874	0.325738	0.972169	1	0.291643	0.257231	0.241562	0.326759	0.32889	0.596037	0.057026	0.039643	0.043502	0.300881	0.281782	0.271469	0.303474	0.282381	0.265589	0.294223	0.257278	0.240521
RSM	0.01	0.018803	0.009638	0.009812	0.327442	1	1	0.29141	0.254036	0.229134	0.331738	0.501227	0.977464	0.058819	0.037183	0.042497	0.30938	0.285236	0.269454	0.303952	0.278422	0.263471	0.292508	0.254091	0.235086
	0.1	0.020691	0.009121	0.008864	0.33285	0.972002	1	0.292914	0.254575	0.236571	0.330271	0.354791	0.813454	0.057447	0.03626	0.042947	0.304059	0.281452	0.271509	0.304132	0.270839	0.261978	0.28997	0.254377	0.236282
	0.3	0.020841	0.008767	0.009573	0.448707	1	1	0.289574	0.252163	0.240054	0.325961	0.330835	0.623337	0.060954	0.039507	0.044898	0.305276	0.285872	0.271122	0.303335	0.278197	0.265233	0.290743	0.256756	0.236787
Zero Flip	0.01	0.022406	0.009776	0.008664	0.411807	1	1	0.291171	0.255902	0.236641	0.40812	0.740236	0.9761	0.055588	0.039575	0.044126	0.306395	0.283904	0.26974	0.305378	0.281999	0.26473	0.292898	0.254943	0.237325
	0.1	0.01917	0.010031	0.011304	0.300356	0.944007	1	0.293724	0.255222	0.238301	0.332533	0.380831	0.953558	0.058246	0.038161	0.04306	0.309527	0.283069	0.272681	0.305571	0.282187	0.26577	0.294383	0.2574	0.237761
	0.3	0.020605	0.008744	0.009162	0.29834	0.971894	1	0.294677	0.255752	0.237546	0.331332	0.381511	0.783152	0.059802	0.038524	0.042062	0.309692	0.287025	0.274028	0.307308	0.282869	0.270532	0.294825	0.260019	0.242105

$M = 500$

		FCX			CGX			HUX			PMFCX			PMGreedyX			1PX			2PX			UX		
		0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1	0.1	0.5	1
FCF	0.01	0.0567	0.041903	0.049092	0.746539	1	1	0.331611	0.294983	0.281048	0.525213	0.766474	1	0.101306	0.084893	0.103251	0.315598	0.302212	0.296211	0.315906	0.300928	0.293272	0.33486	0.291504	0.28232
	0.1	0.059233	0.038332	0.045433	0.687978	1	1	0.308624	0.291722	0.283209	0.376189	0.353456	0.860576	0.102667	0.083246	0.108121	0.314359	0.301266	0.295146	0.313415	0.301317	0.296521	0.306429	0.295011	0.284258
	0.3	0.059373	0.040097	0.05218	0.776394	1	1	0.30661	0.29193	0.284745	0.324768	0.423514	0.763105	0.102406	0.086505	0.109537	0.314519	0.301972	0.295366	0.31327	0.30054	0.295234	0.332963	0.292726	0.284013
IM	0.01	0.056346	0.040209	0.053502	0.832319	1	1	0.309875	0.291008	0.283969	0.64463	0.931568	0.975369	0.103572	0.086154	0.110449	0.314979	0.304093	0.294049	0.340261	0.300662	0.294808	0.360602	0.293842	0.283987
	0.1	0.060678	0.039555	0.047196	0.74773	1	1	0.312215	0.298523	0.291173	0.476297	0.718832	0.951812	0.105754	0.090035	0.107809	0.318111	0.304095	0.299764	0.315492	0.304205	0.297848	0.313153	0.296267	0.287157
	0.3	0.077659	0.045943	0.051817	0.747687	1	1	0.315092	0.303456	0.298408	0.44754	0.646514	0.902963	0.110575	0.091253	0.110588	0.344688	0.310258	0.304912	0.367618	0.30999	0.303842	0.340188	0.302435	0.297291
SM	0.01	0.060808	0.039967	0.051524	0.74445	1	1	0.309868	0.292312	0.284904	0.476072	0.763007	1	0.103418	0.085227	0.104339	0.316411	0.304487	0.294961	0.339233	0.29561	0.29262	0.309886	0.292717	0.284213
	0.1	0.058719	0.042247	0.05048	0.716342	1	1	0.308113	0.294735	0.283099	0.448239	0.522115	0.882591	0.101428	0.084023	0.106208	0.314998	0.303988	0.297083	0.314366	0.301536	0.294043	0.309371	0.293252	0.285806
	0.3	0.061918	0.041415	0.05022	0.777352	1	1	0.310531	0.293024	0.284731	0.373609	0.447894	0.832646	0.102587	0.087271	0.106521	0.314791	0.303838	0.297256	0.311117	0.302181	0.295273	0.309551	0.294491	0.28571
RSM	0.01	0.060034	0.039365	0.052268	0.805097	1	1	0.3334	0.292511	0.28641	0.572228	0.762959	0.951091	0.099058	0.083395	0.102193	0.313565	0.303613	0.295611	0.33707	0.303293	0.295378	0.334087	0.294888	0.284198
	0.1	0.061733	0.040144	0.050997	0.804767	1	0.857143	0.310852	0.29223	0.282943	0.377267	0.524616	0.858222	0.099937	0.086743	0.100759	0.31398	0.30402	0.295624	0.314725	0.302171	0.292089	0.333812	0.296098	0.281452
	0.3	0.056602	0.039861	0.048839	0.717947	1	1	0.309283	0.292287	0.285444	0.326812	0.425973	0.713449	0.105889	0.083337	0.105124	0.314262	0.302794	0.295573	0.336966	0.302352	0.294093	0.307025	0.292042	0.284222
Zero Flip	0.01	0.056056	0.037457	0.049396	0.746463	1	1	0.309999	0.292884	0.284335	0.617913	0.836909	1	0.101699	0.088468	0.109504	0.315902	0.303124	0.296123	0.338415	0.30376	0.296804	0.382475	0.291125	0.284054
	0.1	0.058113	0.041378	0.052921	0.716135	1	1	0.311604	0.29211	0.285731	0.330728	0.668291	0.975867	0.098133	0.083879	0.105897	0.341337	0.300954	0.295863	0.315778	0.302809	0.296237	0.309058	0.296388	0.285872
	0.3	0.060674	0.04358	0.052145	0.776788	1	1	0.311587	0.293899	0.28672	0.331295	0.526049	0.927298	0.102294	0.086981	0.107076	0.317332	0.305522	0.299969	0.316601	0.304186	0.29725	0.31343	0.293392	0.31062

Appendix D

Multiple Crossover GA Results

Chapter 7 proposes a multiple crossover framework that investigates the effect of crossovers corporations on the GA performances. Performance measurements are used to evaluate each crossover. This study helps to develop a new diversity-based GA customised for WSRP. Experiments were conducted to assess the performance of the proposed GA on the 42 real-world WSRP scenarios from Table 2.4. This section provides all results using the parameter settings shown in Table 7.1.

Table D.1: Results of AMC_f with different cycle sizes

		Cycle-size = 5		Cycle-size = 10		Cycle-size = 15		Cycle-size = 25		Cycle-size = 40		Cycle-size = 50	
		f(S)	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*
	1	3.49	2.28	3.49	2.27	3.49	1.39	3.49	1.54	3.49	1.52	3.49	1.27
	2	2.49	2.19	2.54	2.34	2.49	8.38	2.49	1.54	2.49	1.04	2.49	1.34
	3	3.01	39.82	3.02	14.83	3.02	63.45	3.02	164.22	3.01	159.18	3.01	40.36
A	**4**	1.42	37.24	1.42	41.84	1.42	48.61	1.42	37.07	1.42	31.25	1.42	40.20
	5	2.42	0.81	2.42	0.81	2.42	0.48	2.42	0.50	2.42	0.43	2.42	0.39
	6	3.55	11.13	3.55	3.21	3.55	11.45	3.55	21.27	3.55	3.06	3.55	8.13
	7	3.71	0.76	3.71	0.81	3.71	0.46	3.71	0.44	3.71	0.38	3.71	0.35
	1	1.72	173.37	1.72	141.90	1.72	126.70	1.72	53.42	1.72	69.57	1.72	56.57
	2	1.75	0.73	1.75	0.71	1.75	0.46	1.75	0.43	1.75	0.29	1.75	0.29
	3	1.78	429.94	1.78	555.93	1.80	366.64	1.79	293.71	1.78	475.20	1.79	253.69
B	**4**	2.08	3.60	2.08	4.06	2.08	2.33	2.08	2.47	2.08	2.56	2.08	3.45
	5	1.91	423.12	1.90	490.40	1.91	435.69	1.90	440.19	1.92	398.93	1.91	475.67
	6	1.65	282.57	1.65	80.54	1.64	233.08	1.64	52.27	1.64	184.46	1.65	79.07
	7	1.79	183.43	1.79	57.24	1.80	198.38	1.80	160.79	1.80	142.78	1.80	281.44
	1	114.81	547.44	114.81	513.89	114.87	402.88	114.82	636.52	114.75	591.50	114.81	465.10
	2	3.15	0.66	3.15	0.67	3.15	0.18	3.15	0.21	3.15	0.25	3.15	0.28
	3	104.88	583.52	104.65	675.85	105.02	523.95	104.83	638.53	104.88	438.24	104.95	584.06
C	**4**	11.15	1.96	11.15	1.84	11.15	1.30	11.15	1.46	11.15	1.22	11.15	1.20
	5	12.34	58.25	12.34	2.94	12.34	2.77	12.34	2.05	12.34	2.68	12.34	2.23
	6	180.86	488.32	180.80	301.95	180.80	464.63	180.67	415.24	180.86	535.54	180.86	612.04
	7	4.30	0.00	4.30	0.00	4.30	0.00	4.30	0.00	4.30	0.00	4.30	0.00
	1	171.58	3359.44	171.08	3408.46	171.33	3317.93	171.33	3327.88	171.83	3356.06	171.64	3338.33
	2	167.62	3450.02	168.11	3046.96	168.05	3331.77	166.93	3248.75	167.24	3288.78	167.00	3252.35
	3	181.05	3307.73	180.86	3361.38	181.17	3301.24	181.04	3265.26	181.04	3333.52	181.11	3309.33
D	**4**	168.81	3493.58	168.42	3323.28	168.55	3445.52	168.55	3205.52	168.55	3371.49	169.18	3421.87
	5	162.05	3391.26	161.99	3266.39	161.74	3176.14	161.87	3255.79	161.80	3230.10	161.99	3346.06
	6	178.04	3117.82	178.35	3209.79	178.29	3289.81	178.34	3252.99	178.22	3218.53	178.10	3160.83
	7	178.83	3262.37	178.71	3367.56	178.65	3401.26	178.83	3380.42	178.65	3254.92	178.59	3008.68
	1	1.18	3448.63	1.18	3366.72	1.18	3261.94	1.18	3332.52	1.18	3208.34	1.18	3392.45
	2	1.21	3375.66	1.21	3311.93	1.21	3445.05	1.21	3382.67	1.21	3360.36	1.21	3266.36
	3	1.21	3370.88	1.21	3428.17	1.22	3471.14	1.21	3389.79	1.21	3349.07	1.21	3454.17
E	**4**	1.30	3452.65	1.29	3512.18	1.30	3349.56	1.29	3473.30	1.30	3445.57	1.30	3480.99
	5	2.24	3421.14	2.24	3469.36	2.25	3220.36	2.24	3433.64	2.24	3445.60	2.24	3476.29
	6	1.30	3186.31	1.30	3233.54	1.30	3305.84	1.30	3406.23	1.30	3431.79	1.30	3392.44
	7	1.73	3273.79	1.73	3486.11	1.73	3194.47	1.73	3393.55	1.73	3439.09	1.73	3408.90
	1	2019.45	27545.09	2035.17	27760.40	2048.20	27720.13	2042.19	27867.38	2027.16	28221.73	2038.31	27825.95
	2	2084.59	27714.86	2084.84	27671.16	2083.91	26916.03	2085.10	27405.39	2084.53	28020.99	2084.78	27499.32
	3	608.14	27785.94	608.32	27767.11	609.01	27746.14	607.82	27981.40	608.13	27858.40	608.57	27809.54
F	**4**	1330.02	27419.83	1328.77	27863.52	1329.02	26246.46	1329.27	27177.61	1329.77	27328.58	1329.45	27999.67
	5	197.32	28233.26	197.57	26783.72	196.89	27527.14	197.64	27416.26	196.33	27796.63	197.39	28046.44
	6	625.08	28000.67	624.96	27835.21	624.96	27845.46	625.52	28009.39	625.08	28038.06	625.39	28046.17
	7	3496.11	27917.18	3495.55	28138.01	3496.67	26942.08	3496.17	27906.72	3497.30	26561.09	3496.86	27908.09

Table D.2: Results of AMC_d with different cycle sizes

		Cycle-size = 5		Cycle-size = 10		Cycle-size = 15		Cycle-size = 25		Cycle-size = 40		Cycle-size = 50	
		f(S)	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*
	1	3.49	1.99	3.49	1.85	3.49	1.84	3.49	1.78	3.49	1.82	3.49	2.26
	2	2.49	2.65	2.49	2.62	2.49	1.34	2.49	2.18	2.49	1.46	2.49	1.45
	3	3.07	92.91	3.02	58.57	3.04	81.95	3.01	69.89	3.08	6.47	3.01	138.28
A	**4**	1.42	41.35	1.42	67.69	1.42	38.79	1.42	21.47	1.42	63.46	1.42	33.87
	5	2.42	1.44	2.42	1.26	2.42	0.37	2.42	0.36	2.42	0.45	2.42	0.58
	6	3.55	8.40	3.55	8.30	3.55	2.61	3.55	15.30	3.55	5.06	3.55	11.23
	7	3.71	1.37	3.71	1.42	3.71	0.31	3.71	0.30	3.71	0.32	3.71	0.36
	1	1.73	142.49	1.72	276.01	1.73	194.98	1.72	94.37	1.73	209.30	1.73	154.88
	2	1.75	1.83	1.75	1.22	1.75	0.28	1.75	0.36	1.75	0.49	1.75	0.39
	3	1.78	497.99	1.78	570.47	1.79	471.38	1.79	427.43	1.78	223.01	1.79	376.51
B	**4**	2.08	8.49	2.08	21.25	2.08	2.27	2.08	2.52	2.08	2.27	2.08	2.31
	5	1.91	476.62	1.91	483.60	1.91	542.11	1.91	457.65	1.90	492.49	1.90	554.71
	6	1.65	287.19	1.64	72.75	1.65	166.75	1.64	63.15	1.65	176.88	1.64	291.29
	7	1.80	170.98	1.80	244.68	1.80	46.11	1.81	228.09	1.80	297.88	1.80	102.46
	1	115.06	653.46	114.75	566.20	114.75	475.60	115.12	499.97	114.94	503.92	114.87	542.00
	2	3.15	0.92	3.15	0.86	3.15	0.25	3.15	0.26	3.15	0.28	3.15	0.40
	3	104.95	577.64	104.89	434.64	104.82	461.37	104.71	623.05	104.95	585.44	104.89	589.27
C	**4**	11.15	1.87	11.15	1.79	11.15	1.81	11.15	1.89	11.15	2.41	11.15	1.95
	5	12.34	4.33	12.34	68.90	12.34	2.24	12.41	1.72	12.34	51.66	12.34	3.21
	6	180.80	451.39	180.80	431.25	180.86	516.50	180.92	507.93	180.86	484.45	180.79	430.35
	7	4.30	0.00	4.30	0.00	4.30	0.00	4.30	0.00	4.30	0.00	4.30	0.00
	1	170.90	3428.94	170.71	3334.63	170.96	3313.66	171.64	3355.35	171.14	3277.79	171.40	3236.50
	2	168.06	3055.58	167.31	3405.58	167.68	3352.52	167.18	3398.43	166.99	3183.83	167.75	3386.18
	3	181.35	3332.59	180.86	3102.37	180.80	3428.34	180.68	3200.94	180.92	3103.47	180.92	3449.86
D	**4**	168.68	3387.96	168.37	3476.31	168.11	3428.23	169.06	3442.53	168.87	3369.30	169.18	3409.02
	5	161.68	3372.65	161.86	3276.84	161.99	3301.03	161.86	3239.16	161.73	3298.28	161.80	3320.35
	6	178.29	3209.79	178.16	3353.26	178.35	3487.75	178.16	3313.71	177.98	3307.95	178.28	3363.56
	7	178.77	3271.09	178.65	3347.14	178.71	3425.99	178.53	3452.12	178.71	3364.08	178.59	3316.37
	1	1.18	3395.83	1.18	3404.99	1.18	3401.01	1.18	3370.88	1.18	3298.43	1.18	3178.00
	2	1.21	3496.47	1.21	3399.96	1.21	3339.53	1.21	3389.87	1.21	3518.68	1.21	3393.73
	3	1.22	3394.68	1.22	3460.71	1.21	3461.52	1.21	3363.94	1.22	3455.88	1.22	3370.93
E	**4**	1.29	3491.73	1.29	3493.56	1.29	3442.85	1.30	3452.70	1.30	3467.18	1.30	3431.34
	5	2.25	3461.32	2.25	3424.19	2.25	3434.04	2.24	3421.78	2.25	3404.35	2.24	3404.96
	6	1.30	3378.41	1.30	3356.92	1.30	3111.46	1.30	3364.64	1.30	3395.50	1.30	3314.98
	7	1.73	3329.45	1.73	3452.58	1.73	3414.89	1.73	3276.55	1.73	3423.63	1.73	3499.97
	1	2024.90	28127.22	2022.40	28294.63	2049.51	27972.18	2030.04	27920.94	2021.58	27489.73	2038.80	27914.03
	2	2083.10	27876.06	2083.97	27978.79	2084.66	27476.09	2084.09	27299.37	2084.16	27560.80	2083.72	26984.57
	3	608.69	27972.15	608.13	27867.36	608.45	28019.48	608.13	28115.86	608.26	27505.86	608.19	27317.73
F	**4**	1329.02	28263.44	1329.58	28029.66	1329.83	27267.75	1328.77	27890.37	1328.58	27470.42	1329.83	26718.77
	5	197.83	27837.37	197.20	28222.74	197.57	28118.77	196.89	27620.62	199.01	27571.43	197.89	26929.99
	6	625.58	28097.07	624.96	27996.64	625.33	27720.70	625.02	28287.13	625.71	27834.26	625.21	27195.10
	7	3496.86	27518.27	3496.73	27705.17	3496.11	27598.11	3496.42	27924.56	3497.05	28022.93	3495.92	27868.05

Table D.3: Results of AMC_h with different cycle sizes

		Cycle-size = 5		Cycle-size = 10		Cycle-size = 15		Cycle-size = 25		Cycle-size = 40		Cycle-size = 50	
		f(S)	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*
	1	3.49	1.98	3.49	2.23	3.49	2.29	3.49	1.65	3.49	2.31	3.49	2.08
	2	2.49	2.53	2.49	2.74	2.49	1.31	2.49	1.42	2.49	1.97	2.49	1.30
	3	3.01	54.20	3.02	136.87	3.01	50.60	3.02	115.48	3.01	128.35	3.01	76.60
A	**4**	1.42	31.07	1.42	73.36	1.42	64.82	1.42	66.38	1.42	40.18	1.42	49.23
	5	2.42	0.75	2.42	0.69	2.42	0.51	2.42	0.42	2.42	0.76	2.42	0.84
	6	3.55	18.12	3.55	5.07	3.55	15.25	3.55	8.93	3.55	16.80	3.55	6.14
	7	3.71	0.61	3.71	0.63	3.71	0.38	3.71	0.32	3.71	0.55	3.71	0.57
	1	1.73	222.96	1.72	142.12	1.73	195.78	1.72	262.21	1.72	106.75	1.73	141.27
	2	1.75	0.66	1.75	0.62	1.75	0.29	1.75	0.34	1.75	0.53	1.75	0.66
	3	1.79	362.28	1.79	244.00	1.79	368.47	1.78	435.03	1.79	407.60	1.79	382.24
B	**4**	2.08	2.53	2.08	2.78	2.08	3.09	2.08	7.28	2.08	3.22	2.08	3.77
	5	1.92	407.60	1.93	299.61	1.90	511.01	1.92	558.09	1.91	460.96	1.92	458.96
	6	1.65	146.97	1.65	381.47	1.65	201.70	1.65	137.81	1.64	281.41	1.65	378.18
	7	1.80	131.54	1.80	67.72	1.80	53.91	1.79	144.08	1.80	51.86	1.80	30.18
	1	114.81	599.83	114.87	553.38	114.82	598.87	114.88	641.68	114.87	639.05	114.93	574.19
	2	3.15	1.16	3.15	1.04	3.15	0.23	3.15	0.20	3.15	0.25	3.15	0.25
	3	104.96	559.87	104.83	557.28	104.90	526.95	104.82	535.76	105.02	700.38	105.01	657.08
C	**4**	11.15	3.73	11.15	3.63	11.15	2.17	11.15	2.25	11.15	1.91	11.15	2.14
	5	12.34	2.28	12.34	124.38	12.34	61.96	12.34	2.58	12.34	2.58	12.34	82.99
	6	180.80	417.75	180.92	643.55	180.86	599.68	180.86	377.21	180.80	395.24	180.92	450.12
	7	4.30	0.00	4.30	0.00	4.30	0.00	4.30	0.00	4.30	0.00	4.30	0.00
	1	171.45	3399.67	171.45	3392.00	171.14	3442.69	171.08	3255.77	171.52	3250.89	171.39	3209.53
	2	167.12	3437.03	167.94	3453.40	167.62	3453.72	168.31	3450.06	167.56	3444.91	168.12	3309.84
	3	181.61	3396.83	180.93	3360.48	181.42	3318.24	181.55	3390.17	180.98	3209.59	180.80	3340.14
D	**4**	168.93	3365.32	168.37	3367.09	168.43	3355.37	167.86	3412.84	169.37	3362.70	168.42	3407.70
	5	161.80	3147.08	161.61	3316.43	162.11	3186.40	161.92	3113.17	162.05	3399.84	162.05	3408.36
	6	178.16	3300.07	178.10	3435.03	178.35	3360.57	178.28	3278.63	178.16	3268.50	178.47	3342.57
	7	178.77	3403.97	178.83	3472.06	178.65	3291.01	178.59	3271.47	178.71	3009.97	178.77	3283.58
	1	1.18	3444.72	1.18	3252.87	1.18	3342.78	1.18	3206.71	1.18	3427.75	1.18	3421.40
	2	1.21	3357.68	1.21	3363.99	1.22	3274.14	1.21	3431.21	1.21	3434.15	1.21	3229.69
	3	1.22	3397.45	1.22	3435.25	1.22	3344.19	1.22	3395.24	1.22	3423.95	1.22	3383.35
E	**4**	1.30	3352.09	1.30	3218.46	1.30	3472.07	1.30	3432.41	1.30	3473.62	1.30	3444.60
	5	2.24	3473.13	2.24	3465.53	2.24	3373.85	2.25	3450.09	2.25	3500.50	2.25	3399.45
	6	1.30	3477.65	1.30	3076.20	1.30	3326.44	1.30	3377.66	1.30	3294.87	1.30	3313.30
	7	1.73	3382.28	1.73	3240.42	1.73	3476.05	1.73	3375.31	1.73	3415.71	1.73	3382.92
	1	2019.70	27539.20	2023.53	27576.29	2040.69	27203.20	2039.30	27768.47	2021.52	27840.59	2022.08	28361.81
	2	2083.66	27833.10	2084.78	27298.28	2084.91	27373.65	2084.47	27944.11	2084.53	26749.01	2084.65	27489.43
	3	608.45	25627.51	607.95	27834.72	608.38	27685.78	608.63	27280.61	608.45	26875.31	608.63	27584.57
F	**4**	1329.64	27894.24	1330.64	27638.47	1330.51	27890.51	1328.33	27308.04	1328.83	27716.27	1329.39	27386.44
	5	197.64	27252.07	197.89	27175.06	198.26	27787.59	198.51	28380.85	196.89	27739.92	197.14	27788.02
	6	626.52	28030.58	625.39	27763.85	625.08	28323.26	625.77	27519.31	625.71	27946.63	624.21	27766.26
	7	3496.86	26684.63	3496.55	27696.83	3496.92	28046.22	3496.36	28333.90	3495.92	28087.23	3495.73	27715.90

Table D.4: Results of *AMCGA* with different cycle sizes

		Cycle-size = 5		Cycle-size = 10		Cycle-size = 15		Cycle-size = 25		Cycle-size = 40		Cycle-size = 50	
		f(S)	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*	*f(S)*	*Cpt*
	1	3.49	2.80	3.49	2.81	3.49	1.48	3.49	1.58	3.49	1.82	3.49	1.78
	2	2.49	2.44	2.49	4.49	2.49	1.69	2.49	1.41	2.49	16.29	2.49	1.27
	3	3.06	95.17	3.02	221.92	3.04	286.05	3.08	5.18	3.01	75.08	3.02	27.63
A	**4**	1.42	43.16	1.42	51.27	1.42	37.78	1.42	58.82	1.42	55.21	1.42	42.71
	5	2.42	0.92	2.42	0.87	2.42	0.37	2.42	0.43	2.42	0.35	2.42	0.38
	6	3.55	4.66	3.55	26.59	3.55	12.04	3.55	8.33	3.55	5.98	3.55	7.05
	7	3.71	1.01	3.71	1.03	3.71	0.32	3.71	0.30	3.71	0.49	3.71	0.51
	1	1.73	313.12	1.72	160.06	1.72	89.33	1.73	298.03	1.73	163.29	1.73	146.45
	2	1.75	0.95	1.75	1.03	1.75	0.42	1.75	0.33	1.75	0.31	1.75	0.29
	3	1.79	297.41	1.77	327.03	1.79	370.93	1.78	534.24	1.78	482.33	1.79	468.52
B	**4**	2.08	5.25	2.08	5.12	2.08	2.99	2.08	2.98	2.08	3.24	2.08	3.17
	5	1.91	343.26	1.92	459.26	1.91	562.37	1.91	571.54	1.91	493.85	1.91	396.11
	6	1.64	117.25	1.65	97.23	1.65	299.55	1.64	41.67	1.64	260.05	1.64	146.05
	7	1.79	149.84	1.80	87.54	1.80	136.60	1.80	231.27	1.80	126.49	1.80	78.43
	1	114.75	672.78	114.87	644.23	114.75	584.60	114.83	661.41	114.94	750.26	114.94	640.35
	2	3.15	0.80	3.15	0.71	3.15	0.26	3.15	0.30	3.15	0.32	3.15	0.30
	3	104.75	710.21	104.75	804.69	104.95	753.96	104.82	646.56	104.76	792.24	104.82	681.10
C	**4**	11.15	3.49	11.15	3.14	11.15	2.36	11.15	2.39	11.15	2.43	11.15	2.64
	5	12.34	2.83	12.34	2.40	12.34	2.52	12.34	8.87	12.34	2.02	12.34	2.35
	6	180.92	637.29	180.92	586.95	180.99	642.23	180.99	542.92	180.86	520.03	180.99	446.39
	7	4.30	0.00	4.30	0.00	4.30	0.00	4.30	0.00	4.30	0.00	4.30	0.00
	1	170.29	2642.44	170.23	2533.55	169.86	2688.24	170.42	2914.28	170.17	2364.17	170.29	2545.39
	2	166.02	2514.73	166.58	2109.05	166.14	2491.34	166.51	2644.24	166.64	2715.30	166.26	2512.36
	3	179.69	2580.69	179.81	2490.46	179.69	2702.18	179.44	2848.37	179.63	2807.11	179.75	2370.22
D	**4**	166.70	2934.27	166.64	2382.08	166.82	2859.10	166.94	2502.69	166.44	2759.27	166.39	2511.47
	5	161.44	2832.85	161.50	2503.26	161.32	2899.74	161.32	2865.06	161.38	2465.06	161.37	2532.90
	6	178.23	2569.79	178.11	2765.96	177.99	2578.19	178.17	2381.86	178.30	2499.03	177.99	2368.70
	7	178.41	2743.62	178.41	3054.21	178.48	2360.13	178.48	2616.74	178.42	2027.89	178.41	2217.13
	1	1.17	2917.65	1.17	3390.71	1.17	2933.58	1.17	3103.12	1.17	3163.68	1.17	3285.83
	2	1.20	3291.38	1.20	3303.48	1.20	3379.91	1.20	3276.42	1.20	3287.68	1.20	3311.01
	3	1.21	3340.55	1.21	3181.93	1.21	3295.07	1.21	3165.87	1.21	3085.13	1.21	3137.97
E	**4**	1.29	3343.83	1.29	2932.00	1.29	3138.78	1.29	3120.54	1.29	3402.20	1.29	3123.67
	5	2.24	3291.58	2.24	2921.37	2.24	3449.67	2.24	3166.16	2.24	3355.34	2.24	3378.24
	6	1.30	3056.44	1.30	2539.72	1.30	2704.09	1.30	2984.45	1.30	2718.68	1.30	2975.27
	7	1.72	3172.71	1.72	3314.44	1.72	3410.30	1.72	3413.22	1.72	3317.77	1.72	3278.99
	1	2028.92	24054.86	2036.82	23117.21	2026.79	23474.04	2025.54	26021.25	2022.53	26269.72	2019.72	22709.26
	2	2081.92	22358.29	2081.99	23765.59	2080.99	23405.49	2082.61	25059.22	2082.05	21904.99	2081.74	22516.85
	3	604.77	24585.93	605.14	22220.34	604.58	24437.32	604.96	23856.19	605.40	23586.39	604.71	22658.50
F	**4**	1326.46	22606.42	1326.28	24782.11	1325.90	26262.39	1325.84	26862.95	1325.96	24967.53	1325.90	23458.95
	5	193.78	24042.89	193.59	25631.57	192.90	25798.48	193.84	26811.12	193.46	25220.68	192.78	25895.77
	6	622.22	26733.99	622.41	26415.45	622.15	26478.64	622.47	25364.21	622.09	26382.94	622.34	23680.04
	7	3493.24	25911.33	3492.87	24083.07	3493.31	25423.22	3493.99	23516.84	3492.93	26840.03	3493.18	24846.23