

The University of Nottingham
UNITED KINGDOM · CHINA · MALAYSIA
School of Computer Science

A Computational Study and Heuristic Algorithms for the Home Healthcare Scheduling and Routing Problem

Rodrigo Lankaites Pinheiro

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

November 2017

Abstract

The workforce scheduling and routing problem (WSRP) arises in many scenarios in which skilled workers need to deliver services at different locations. Examples of these scenarios include nurses visiting patients at home, technicians carrying out repairs at customers locations, security guards performing rounds at different premises, etc. Hence, finding a solution to this type of problem involves tackling both the *scheduling of tasks* to be carried out and the *routing of workers* to visit the locations of tasks. The focus of this thesis is a challenging real-world problem in the planning and scheduling of healthcare delivery, denoted by home healthcare (HHC) problem. The first intended contribution of this PhD research is to provide to the research community six benchmark datasets for a real-world workforce scheduling and routing problem arising in healthcare delivery as well as benchmark results obtained with heuristic methods. The next contribution is an algorithm to obtain lower bound values for the HHC problem that is capable of providing results when mathematical techniques are not applicable. The lower bounds are obtained by splitting and relaxing the problem into a scheduling and a routing subproblems, and calculating individual lower bounds for each subproblem. In order to further understand the HHC problem, its multiobjective characteristics were assessed. Understanding the relationships between objectives in a multiobjective optimisation problem is impor-

tant for developing tailored and efficient solving techniques. In particular, when tackling combinatorial optimisation problems with many objectives that arise in real-world scenarios, better support for the decision maker can be achieved through better understanding of the often complex fitness landscape. This thesis makes a contribution in this direction by presenting a technique that allows a visualisation and analysis of the local and global relationships between objectives in optimisation problems with many objectives. The proposed technique uses four steps: first the global pairwise relationships are analysed using the Kendall correlation method; then the ranges of the values found on the given Pareto front are estimated and assessed; next these ranges are used to plot a map using Gray code, similar to Karnaugh maps, that has the ability to highlight the trade-offs between multiple objectives; and finally local relationships are identified using scatter-plots. Results show that each dataset has different characteristics, and the relationships between objectives and their importance vary across datasets; also, instances of the same dataset share similar fitness landscapes. A Variable Neighbourhood Search (VNS) metaheuristic algorithm is proposed to tackle the HHC problem, incorporating three heuristics tailored to the problem-domain. The first heuristic restricts the search space using a priority list of candidate workers; the second heuristic seeks to reduce the violation of worker availabilities soft constraints; the third heuristic estimate the objective costs of all possible individual assignments and uses the estimated costs instead of the objective function, hence substantially increasing the speed of the search. Two greedy constructive heuristics are presented to give the VNS a good starting point. It is shown that the proposed VNS obtains substantial improvements in the quality of solutions regardless of the datasets unique features. The proposed VNS provides

the current benchmark results for the set of real-world HHC scenarios considered. The last contribution of this thesis is a multiobjective solving methodology that exploits the fact that instances of the same dataset share similar fitness landscapes. The methodology consists of obtaining an approximation set of a single instance and using that approximation set, goal programming and the proposed VNS to reach target solutions in the remaining instances. Results show that the methodology is accurate enough to reach the target solution and is able to provide quick results when multiobjective algorithms take long computational times.

Contents

Abstract **v**

List of Figures **xxviii**

List of Tables **xxix**

Acknowledgements **xxx**

1 Introduction **1**

1.1 The Home Healthcare Project 3

1.2 Overview of the Thesis 4

1.2.1 Chapter 2 4

1.2.2 Chapter 3 4

1.2.3 Chapter 4 4

1.2.4 Chapter 5 5

1.2.5 Chapter 6 5

1.2.6 Chapter 7 5

1.2.7 Chapter 8 6

1.2.8 Chapter 9 6

1.2.9 Appendices 6

1.3 Contributions 7

1.4 Publications 8

2 Background Knowledge **12**

2.1 Optimisation Problems 12

2.1.1 Constraints 13

2.1.2 Objective Function 13

2.1.3 Search Space and Objective Space 14

2.1.4 Computational Complexity 14

2.2 Exact Solving Methods 15

2.3 Heuristic Algorithms 16

2.3.1 Constructive 17

2.3.2 Local Search 18

2.4 Metaheuristics 19

2.4.1 Local Search 20

2.4.2 Variable Neighbourhood Search 21

2.4.3 Constructive 23

2.4.4 Population-Based 24

2.5 Multiobjective Optimisation Problems 29

2.5.1 Pareto Dominance 30

2.5.2 Decision Making Process 32

2.6 Many-objective Optimisation 35

2.7 Objective Relationships in Multiobjective Optimisation . . 36

2.7.1 Conflicting Relationships 36

2.7.2 Harmonious Relationships 37

2.7.3 Independent Relationships 37

2.8 Multiobjective Visualisation Techniques 38

2.8.1 Scatter-Plots 38

2.8.2 Parallel Coordinates 39

2.8.3 Heatmaps 39

2.8.4 Other Visualisation Mechanisms 40

2.9 Solving Methodologies 41
2.9.1 Nondominated Sorting Genetic Algorithm 2 43
2.9.2 Multiobjective Evolutionary Algorithm Based on Decomposition 44

3 The Home Healthcare Problem 46

3.1 The Home Healthcare Problem: Definition and Formulation 48
3.1.1 Parameters, Variables and Constraints 50
3.1.2 Objective Function 54
3.2 Six Real-World HHC Problem Datasets 60
3.2.1 Description of Data Model 64
3.3 Experiments and Results 65
3.3.1 Greedy Constructive Heuristic 65
3.3.2 MIP Solver and Decomposition Method 66
3.3.3 Scheduling and Routing Balance Evaluation 67
3.4 Detailed Solution Analysis 69
3.5 Conclusion 74

4 Lower Bounds for Workforce Scheduling and Routing Problems 76

4.1 Methodology 79
4.2 Lower Bounds for the Relaxed Scheduling Subproblem . . 83
4.2.1 Unassigned Visits 88
4.2.2 Time and Area Availabilities 98
4.3 Lower Bounds for the Relaxed Routing Subproblem 99
4.4 Lower Bounds for the HHC problem 109
4.5 Experimental Results 111
4.6 Conclusion 116

5 Understanding Relationships Between Objectives in Optimisation Problems 119

5.1 Motivation and Related Work 122

5.2 Objective Relationships in Multiobjective Optimisation . . 125

5.3 A Four-Step Analysis and Visualisation Technique 127

5.3.1 Step 1 – Global Pairwise Relationship Analysis: . . 128

5.3.2 Step 2 – Objective Range Analysis: 129

5.3.3 Step 3 – Trade-off Regions Analysis: 130

5.3.4 Step 4 – Multiobjective Scatter Plot Analysis: 134

5.3.5 The Reliability of the Approximation Set 136

5.4 Case Studies 137

5.4.1 Multiobjective Multidimensional Knapsack Problem (MMKP) 137

5.4.2 Multiobjective Nurse Scheduling Problem (MONSP) 139

5.4.3 Multiobjective Vehicle Routing Problem with Time Windows (MOVRPTW) 143

5.5 Application of the Four-Step Technique 144

5.5.1 Multiobjective Multidimensional Knapsack Problem 145

5.5.2 Multiobjective Nurse Scheduling Problem 154

5.5.3 Multiobjective Vehicle Routing Problem with Time Windows 160

5.6 Conclusion 166

6 Multiobjective Analysis of the Home Healthcare Problem 169

6.1 The Multiobjective Home Healthcare Optimisation Problem (MOHHC) 169

6.2 Experimental Configuration 171

6.2.1 Indirect Individual Encoding 171

6.2.2 Obtaining the Approximation Set 172
6.3 Four-Step Analysis 173
6.3.1 Step 1 - Global Pairwise Relationships Analysis . . 174
6.3.2 Step 2 - Objective Range Analysis 175
6.3.3 Step 3 - Trade-Off Regions Analysis 176
6.3.4 Step 4 - Multiobjective Scatter Plot Analysis 181
6.4 Conclusion 183

7 A Variable Neighbourhood Search for the Home Healthcare Problem 186

7.1 Variable Neighbourhood Search 187
7.1.1 Shaking Neighbourhoods 188
7.1.2 Randomised Hill Climbing (RHC) 189
7.1.3 Priority-Based Search 189
7.1.4 Availability Violations Seeker (AVS) 193
7.1.5 Flat-Costs Search 195
7.1.6 Constructive Heuristics 198
7.2 Experimental Results 199
7.2.1 Starting Point Evaluation 201
7.2.2 Evaluation of Individual Components 202
7.2.3 Overall Performance 203
7.2.4 Flat Cost Search 205
7.2.5 Comparative Results 207
7.2.6 Constructive Heuristics 208
7.3 Conclusion 213

8 Using VNS on Estimated Pareto Fronts to Solve Multiobjective Problems 216

8.1 Goal Programming (GP) 217

8.2 Proposed Estimation Methodology 220
8.2.1 Chebyshev Goal Programming 221
8.2.2 Derived Weight Vector 222
8.2.3 Euclidian Distances 224
8.3 Experimental Configuration 225
8.4 Results 226
8.5 Discussion 234
8.6 Conclusion 236

9 Conclusions **239**
9.1 Review of Contributions 240
9.1.1 Real World HHC Problem Datasets 240
9.1.2 Lower Bounds for the HHC Problem 241
9.1.3 Analysis of Objectives Relationships in Multiobjective Problems Using Trade-Off Region Maps . . . 242
9.1.4 A Variable Neighbourhood Search for the HHC Problem 243
9.1.5 Using VNS on Estimated Pareto Fronts to Solve the Multiobjective HHC Problem 244
9.1.6 Collaborative Research and Development (R&D) Projects for Optimisation Solutions 246
9.2 Final Remarks 247
9.3 Future Work 249

Bibliography **251**

Appendices **277**

A A Development and Integration Framework for Optimisation-Based Enterprise Solutions **279**

A.1 Related Work 280
A.2 First Component: Data Model 282
A.2.1 Modelling the Data 283
A.3 Second Component: Data Extraction and Validation 285
A.3.1 Data Extractor 285
A.3.2 Data Validator 285
A.4 Third Component: Solution Visualisation and Auxiliary Platform 287
A.4.1 Solution Visualiser 287
A.4.2 Auxiliary Platform 288
A.5 Conclusion 289

B A Data-Centric Application Program Interface for Optimisation Problems 290

B.1 Related Work 291
B.2 API Concept 293
B.3 XML Data Model 297
B.4 Internal Data Structures 298
B.4.1 Code Optimisation 300
B.5 The Solution Builder 303
B.5.1 Centralised Evaluation Mechanism 303
B.5.2 Solution Dispenser 304
B.5.3 Code Optimisation 305
B.6 Summary 306
B.6.1 Improvements on the Research and Development . 307
B.7 Experimental Results 309
B.7.1 Solution Builder 309
B.7.2 Design Patterns 313

B.8 Conclusion 320

C Scatter Plots of the Home Healthcare Datasets **321**

List of Figures

2.1 Example of local and global optima in a minimisation problem. 21

2.2 Example of one point crossover operator. The cutting point is after the fourth gene. The number above each offpring's gene indicates which parent provided that gene. 27

2.3 Example of two points crossover operator. The cutting points are after the second and seventh genes. The number above each offpring's gene indicates which parent provided that gene. 27

2.4 Example of uniform crossover operator. The number above each offpring's gene indicates which parent provided that gene. 28

2.5 Example of scatter plots for two, three and four dimensions. 38

2.6 Example of a parallel coordinates graph considering four objectives. 39

3.1 Relative gap to the optimal solution with different objective function setups. 68

3.2 Comparative results from the average solution of each set. 71

4.1 Overview of the proposed technique to calculate a lower bound for the WSRP. 80

4.2 Solution representation. Each column represent the assignment of a visit to a worker. Note that visit 1 requires two workers. 82
4.3 Example of a conflict graph. On *i)* we have the vertices of the graph, representing the visits and the times in which they occur. On *ii)* we have the full conflict graph with the maximal cliques expressed in different colours. 91
4.4 Example of a conflicting clique. In *i)*a maximal conflicting clique consisting of four vertices is presented. Near each vertex is the list of available workers (namely A, B and C). On *ii)* we have the resulting graph after removing the vertex 4, hence eliminating all conflicts. 92
4.5 Sample execution of the algorithm to detect unassigned visits. In *i)* the problem is presented with the three maximal cliques presented in different colours. Each subsequent part of the figure presents the removal of a vertex according to the algorithm criteria. 97
4.6 Types of vertices considered by the algorithm. 101
4.7 Example showing that limiting the number of edges is required. On *ii)* an optimal solution is shown. On *iii)* a valid classification of the vertices that incurs an invalid lower bound is shown. On *iv)* a valid classification where the number of vertices is retricted presents a lower bound. . . 103
4.8 Example of the proposed algorithm. The algorithm follows the vertex numbering to iterate and classify all vertices.105
4.9 Example of the proposed algorithm. The algorithm follows the reverse vertex numbering to iterate and classify all vertices. 107

4.10 Example showing that it is necessary to know which visits are left unassigned in the optimal solution of the HHC. . . 108
4.11 Summary of the comparison between all algorithms on all datasets. 112
4.12 Cumulative improvements on the lower bounds obtained by varying the clique size of the $LB_{UA/T}$ algorithm in relation to using a size 1 clique. 114
4.13 Cumulative improvements on the lower bounds obtained by varying the clique size of the LB_{UAT} algorithm in relation to using a size 1 clique. 114
4.14 Gap from the distances lowerbounds obtained by the RRS algorithm against the distances presented in the best known solution. 115

5.2 Three-way conflicting objectives 126
5.3 Region map schematics. 132
5.4 Importance of choosing the right pivot on a three-objective approximation set: in a) Z_1–Z_3 conflicting relationship is easily spotted, while in b) the same relationship can not be visualised. 135
5.5 Pairwise correlations for each set of instances. 146
5.6 Results for the objective ranges analysis. The central point shows the average value for the set, the upward lines represent the maximum value found and the downward line represents the minimum value. 147
5.7 Threshold analysis showing the number of scenarios with solutions found in region r_0 when the threshold decreases. 148
5.8 Results for the trade-off regions analysis for Set A. 148
5.9 Results for the trade-off regions analysis for Set C. 149

5.10 Results for the trade-off regions analysis for Set D. 149

5.11 Results for the trade-off regions analysis for Set X. 150

5.12 Results for the scatter plot analysis. 151

5.13 Pairwise correlations for each set of instances. 155

5.14 Range analysis results for both the generational sets and the distinct cases. The central point shows the average value for the set, the upward lines represent the maximum value found and the downward line represents the minimum value. 155

5.15 Threshold analysis showing the number of scenarios with solutions found in region r_0 when the threshold decreases. 156

5.16 Region maps for Cases A. 157

5.17 Region maps for Cases B. 157

5.18 Results for the scatter plot analysis. 158

5.19 Pairwise correlations for each set of instances. 161

5.20 Range analysis results for each objective considering instances with different time windows and delta set-ups. The central point shows the average value for the set, the upward lines represent the maximum value found and the downward line represents the minimum value. 162

5.21 Threshold analysis showing the number of scenarios with solutions found in region r_0 when the threshold decreases. 163

5.22 Overall region maps. 163

5.23 Results for the scatter plot analysis. 164

6.1 Pairwise correlations for each set of instances. The tick black line highligths a trend. 174

6.2 Range analysis results for both the generational sets and the distinct cases. The central point shows the average value for the set, the upward lines represent the maximum value found and the downward line represents the minimum value. 175
6.3 Threshold analysis showing the number of scenarios with solutions found in region r_0 when the threshold decreases. 177
6.4 Region maps for instance set A. 177
6.5 Region maps for instance set B. 178
6.6 Region maps for instance set C. 178
6.7 Region maps for instance set D. 179
6.8 Region maps for instance set E. 180
6.9 Region maps for instance set F. 180
6.10 Scatter plots of each combined dataset, showing the relationship between objective Z_4 and each of the other four objectives. In each case, the x axis shows the value of objective Z_4 and the y axis shows the values of each of the other objectives in different colours. 182

7.1 Diagram for the Priority-Based Search (PBS). 191
7.2 Diagram for the Availability Violations Seeker Search (AVS).193
7.3 Diagram for the Flat-Costs Search (FCS). 196
7.4 Average gap to the best known solution for each set. Featuring the proposed VNS starting from the constructive heuristics (1K-VNS) and from a random solution (1K-RVNS).201
7.5 Relative gap comparison between the proposed VNS (Full-VNS), a VNS with only the AVS local search (AVS-VNS), a VNS with only the PBS local search (PBS-VNS), and a VNS with only the FCS local search (FCS-VNS) 203

7.6 Comparison of the proposed VNS (Full-VNS) against the VNS with a simple hill climbing neighbourhood (HC-VNS) and a VNS with only the AVS and PBS local searches (AVPB-VNS) in relation to the average gap to the best known solution for each set. 204

7.7 Number of iterations for the HC-VNS, the PBS-VNS and the FCS-VNS on eight hours runs for the F dataset. 206

7.8 Comparison of the VNS using only the FCS local search (FCS-VNS) and the VNS using all local searches and a ρ value of 1.000 (1K-VNS) and 10.000 (10K-VNS). 206

7.9 Comparative results between the proposed Full-VNS, the genetic algorithm (GA) and the decomposition technique using exact methods (Decomp.). 208

7.10 Performance comparison of the FCH heuristic against the Greedy heuristic. 209

7.11 Time comparison between the FCH heuristic and the Greedy heuristic. 210

8.1 Overview of the proposed methodology. The numbered steps are explained below. 220

8.2 Average percentage of the solutions in the approximation set of each target instance such that $\vec{w}\vec{Z}^t \leq \vec{w}\vec{Z}^i$. 226

8.3 Summary of results for Dataset A. 228

8.4 Summary of results for Dataset B. 229

8.5 Summary of results for Dataset C. 230

8.6 Summary of results for Dataset D. 231

8.7 Summary of results for Dataset E. 232

8.8 Summary of results for Dataset F. 233

8.9 Example of an inaccurate objective-function on a Pareto front that is well-spread (a) and on a Pareto front with gaps (b). 234

B.1 Overview of the API. 293

B.2 Main features of the WSRP API, where (1) refers to the optimisation solver (mathematical solver, heuristic algorithms, etc.), (2) and (10) to the data parser, (3) to the internal memory structures, (4) to (8) to the operations and methods supported by the API and (9) to the Solution Dispenser. 294

B.3 Class Diagram for the Main Problem Class and the Tasks-related Classes. 299

B.4 Memory mapping from memory to CPU cache of an aligned chunk of data. 302

B.5 Performance loss of a memory mapping of misaligned data. 303

B.6 Flow Diagram for the Solution Builder. The Rectangles Inside the Solution Builder Box Represent the Functionalities Given to the User. Arrows Represent Data Flow. . . . 304

B.7 Time comparison between the Solution Dispenser (SD) and the Java Garbage Collector (GC) to instantiate and dispose new solutions. 310

B.8 Memory comparison between the Solution Dispenser (SD) and the Java Garbage Collector (GC). 311

B.9 Number of generations of a genetic algorithm with different solution sizes. 312

B.10 Increase in the number of generation when employing the Solution Builder in detriment of the built-in Garbage Collector. 313

B.11 Performance comparison of using values contained locally inside a structure (Local structure) against only the references to the values contained locally (External structure). . 314

B.12 Time comparison of performing sequential operations on a matrix using row order and using column order. 315

B.13 Results for 10,000 generations of MKP solutions using a random search employing an implementation using the data locality and matrix ordering design patterns (A) and an implementation not using these patterns (B). 318

B.14 Performance loss for not employing the data locality and matrix ordering design patterns. 319

C.1 Scatter plots of each of the WSRP-A instances, showing the relationship between objective Z_1 and each of the other four objectives. In each case, the x axis shows the value of objective Z_1 and the y axis shows the values of each of the other objectives in different colours. 322

C.2 Scatter plots of each of the WSRP-B instances, showing the relationship between objective Z_1 and each of the other four objectives. In each case, the x axis shows the value of objective Z_1 and the y axis shows the values of each of the other objectives in different colours. 323

C.3 Scatter plots of each of the WSRP-C instances, showing the relationship between objective Z_1 and each of the other four objectives. In each case, the x axis shows the value of objective Z_1 and the y axis shows the values of each of the other objectives in different colours. 324

C.4 Scatter plots of each of the WSRP-D instances, showing the relationship between objective Z_1 and each of the other four objectives. In each case, the x axis shows the value of objective Z_1 and the y axis shows the values of each of the other objectives in different colours. 325

C.5 Scatter plots of each of the WSRP-E instances, showing the relationship between objective Z_1 and each of the other four objectives. In each case, the x axis shows the value of objective Z_1 and the y axis shows the values of each of the other objectives in different colours. 326

C.6 Scatter plots of each of the WSRP-F instances, showing the relationship between objective Z_1 and each of the other four objectives. In each case, the x axis shows the value of objective Z_1 and the y axis shows the values of each of the other objectives in different colours. 327

C.7 Scatter plots of each of the WSRP-A instances, showing the relationship between objective Z_2 and each of the other four objectives. In each case, the x axis shows the value of objective Z_2 and the y axis shows the values of each of the other objectives in different colours. 328

C.8 Scatter plots of each of the WSRP-A instances, showing the relationship between objective Z_2 and each of the other four objectives. In each case, the x axis shows the value of objective Z_2 and the y axis shows the values of each of the other objectives in different colours. 329

C.9 Scatter plots of each of the WSRP-C instances, showing the relationship between objective Z_2 and each of the other four objectives. In each case, the x axis shows the value of objective Z_2 and the y axis shows the values of each of the other objectives in different colours. 330

C.10 Scatter plots of each of the WSRP-D instances, showing the relationship between objective Z_2 and each of the other four objectives. In each case, the x axis shows the value of objective Z_2 and the y axis shows the values of each of the other objectives in different colours. 331

C.11 Scatter plots of each of the WSRP-E instances, showing the relationship between objective Z_2 and each of the other four objectives. In each case, the x axis shows the value of objective Z_2 and the y axis shows the values of each of the other objectives in different colours. 332

C.12 Scatter plots of each of the WSRP-F instances, showing the relationship between objective Z_2 and each of the other four objectives. In each case, the x axis shows the value of objective Z_2 and the y axis shows the values of each of the other objectives in different colours. 333

C.13 Scatter plots of each of the WSRP-A instances, showing the relationship between objective Z_3 and each of the other four objectives. In each case, the x axis shows the value of objective Z_3 and the y axis shows the values of each of the other objectives in different colours. 334

C.14 Scatter plots of each of the WSRP-B instances, showing the relationship between objective Z_3 and each of the other four objectives. In each case, the x axis shows the value of objective Z_3 and the y axis shows the values of each of the other objectives in different colours. 335

C.15 Scatter plots of each of the WSRP-C instances, showing the relationship between objective Z_3 and each of the other four objectives. In each case, the x axis shows the value of objective Z_3 and the y axis shows the values of each of the other objectives in different colours. 336

C.16 Scatter plots of each of the WSRP-D instances, showing the relationship between objective Z_3 and each of the other four objectives. In each case, the x axis shows the value of objective Z_3 and the y axis shows the values of each of the other objectives in different colours. 337

C.17 Scatter plots of each of the WSRP-E instances, showing the relationship between objective Z_3 and each of the other four objectives. In each case, the x axis shows the value of objective Z_3 and the y axis shows the values of each of the other objectives in different colours. 338

C.18 Scatter plots of each of the WSRP-F instances, showing the relationship between objective Z_3 and each of the other four objectives. In each case, the x axis shows the value of objective Z_3 and the y axis shows the values of each of the other objectives in different colours. 339

C.19 Scatter plots of each of the WSRP-A instances, showing the relationship between objective Z_4 and each of the other four objectives. In each case, the x axis shows the value of objective Z_4 and the y axis shows the values of each of the other objectives in different colours. 340

C.20 Scatter plots of each of the WSRP-B instances, showing the relationship between objective Z_4 and each of the other four objectives. In each case, the x axis shows the value of objective Z_4 and the y axis shows the values of each of the other objectives in different colours. 341

C.21 Scatter plots of each of the WSRP-C instances, showing the relationship between objective Z_4 and each of the other four objectives. In each case, the x axis shows the value of objective Z_4 and the y axis shows the values of each of the other objectives in different colours. 342

C.22 Scatter plots of each of the WSRP-D instances, showing the relationship between objective Z_4 and each of the other four objectives. In each case, the x axis shows the value of objective Z_4 and the y axis shows the values of each of the other objectives in different colours. 343

C.23 Scatter plots of each of the WSRP-E instances, showing the relationship between objective Z_4 and each of the other four objectives. In each case, the x axis shows the value of objective Z_4 and the y axis shows the values of each of the other objectives in different colours. 344

C.24 Scatter plots of each of the WSRP-F instances, showing the relationship between objective Z_4 and each of the other four objectives. In each case, the x axis shows the value of objective Z_4 and the y axis shows the values of each of the other objectives in different colours. 345

C.25 Scatter plots of each of the WSRP-A instances, showing the relationship between objective Z_5 and each of the other four objectives. In each case, the x axis shows the value of objective Z_5 and the y axis shows the values of each of the other objectives in different colours. 346

C.26 Scatter plots of each of the WSRP-B instances, showing the relationship between objective Z_5 and each of the other four objectives. In each case, the x axis shows the value of objective Z_5 and the y axis shows the values of each of the other objectives in different colours. 347

C.27 Scatter plots of each of the WSRP-A instances, showing the relationship between objective Z_5 and each of the other four objectives. In each case, the x axis shows the value of objective Z_5 and the y axis shows the values of each of the other objectives in different colours. 348

C.28 Scatter plots of each of the WSRP-A instances, showing the relationship between objective Z_5 and each of the other four objectives. In each case, the x axis shows the value of objective Z_5 and the y axis shows the values of each of the other objectives in different colours. 349

C.29 Scatter plots of each of the WSRP-E instances, showing the relationship between objective Z_5 and each of the other four objectives. In each case, the x axis shows the value of objective Z_5 and the y axis shows the values of each of the other objectives in different colours. 350
C.30 Scatter plots of each of the WSRP-F instances, showing the relationship between objective Z_5 and each of the other four objectives. In each case, the x axis shows the value of objective Z_5 and the y axis shows the values of each of the other objectives in different colours. 351

List of Tables

3.1 Notation used in MIP model for WSRP 49

3.2 Values for the λ coefficients. 58

3.3 Features present in the datasets. 60

3.4 Basic information on the proposed WSRP scenarios. 61

3.5 Comparing the decomposition technique against the greedy constructive heuristic for the different weight setups in the objective function. 68

3.6 Detailed information for the solutions found for each instance. 69

4.1 Detailed results for all algorithms. The distances are included in the results. 117

5.1 Generational patterns for the MNSP using the NSPLib files. Letters represent base files. 142

7.1 Detailed results of the experiments for datasets A, B and C. 211

7.2 Detailed results of the experiments for datasets D, E and F. 212

B.1 Results for the memory alignment pattern. 316

Acknowledgements

"*For from Him and through Him and to Him are all things. To Him be the glory forever! Amen.*" Romans 11:36

I wish to express my sincere appreciation to those who have contributed to this thesis and supported me in one way or the other during this amazing journey.

First and foremost I would like to thank my wife, Marcia, for supporting me throughout this journey. From dropping her career in Brazil, to come to the UK and work all this time to make this possible. To you, my love, my eternal gratitude.

Words cannot express the feelings I have for my parents, Agnaldo and Guiomar, and my sister Izabela for their constant unconditional support - both emotionally and financially. I would not be here if it not for you. Special thanks are also due to my grandma Maria who supported me in the beginning of my academic journey; and grandma Izabel.

My sincere gratitude to my main supervisor, Associate Professor Dario Landa-Silva for his invaluable insights, suggestions, and all the help on personal subjects. I really appreciate all the patience and thoroughness in teaching me how to be a better researcher and scientist. You are an inspiration. Also, I would like to thank my second supervisor, Associate Professor Jason Atkin, for the advice and help during this journey.

Thanks to the University of Nottingham for the opportunity and re-

sources; and members of the ASAP research group for general advice and camaraderie. Special thanks to Wasakorn, Arturo, Haneen and Jonata for the collaboration and good times.

My thanks go to Innovate UK, the Knowledge Transfer Partnership programme, and Webroster Ltd. for choosing, trusting, and allowing me time to fulfill my academic obligations while working on the project. Special thanks to Richard, Gary, and Nigel. Also, my gratitude goes to my second supervisor of the KTP, Associate Professor Rong Qu, for encouraging, guiding and sharing her immense knowledge.

This journey would not be possible without the help and guidance of former teachers, lecturers, and professors that through the course of my life taught me and shaped me. Thanks to Colégio Santo Inácio for the complete basic education. Thanks to the Universidade Estadual de Maringá for the excellent academic formation. Special thanks to my former supervisors, Professor Candido Ferreira Xavier de Mendonça Neto and Professor Ademir Aparecido Constantino; your help and guidance are still invaluable to this day.

Many thanks to my friends in Brazil. The conference calls, regular messaging and friendship helped me to endure this long time away from home. Special thanks to Rafael, Beto, Jão, Gustavo, the Bol brothers, Iury, and Afonso.

And finally, my gratitute goes as well to Dr. Andrew Parkes (internal examiner) from the University of Nottingham and Professor Greet Vanden Berghe (external examiner) from the KU Leuven University. Their valuable comments and suggestions during my PhD examination have been included in the final version of this thesis.

Chapter 1

Introduction

Healthcare delivery is an important and growing industry in the UK [172]. A typical home healthcare business needs managers and coordinators to carefully generate work plans for their care workers to visit their patients. By creating improved plans, the business is able to improve the quality of the service delivered (by allowing the best care worker to be assigned for the visit, considering the patient's preferences and continuity of care), reduce travel and payment costs and improve working conditions (by satisfying workers' preferences too). Using an automated solver can potentially allow businesses to achieve improved work plans, reduce the number of office staff (a coordinator using the system could manage more daily visits and staff) and therefore allow businesses to grow.

The *Workforce Scheduling and Routing Problem* (WSRP) is an NP-Hard combinatorial optimisation problem arising in several industries such as home healthcare, security and telecommunication services where a mobile workforce carries out jobs at different locations [42]. WSRP combines constraints and objectives from both scheduling and routing problems. Regarding the scheduling aspects, the problem consists of finding

a work plan for skilled mobile workforce to deliver services [91, 212]. Each service or visit usually requires some mandatory skills and some desirable skills too. Often, clients' preferences for specific staff are indicated for some visits. Working regulations and preferences must also be considered. That is, maximum working hours must always be respected while staff availability for certain working times and geographical areas of work should be observed whenever possible. Hence, an assignment of a worker to a visit must fulfil the mandatory skill requirements and working regulations whilst maximising the satisfaction of desirable requirements and preferences [78, 198]. Regarding the routing aspects, each service or visit has an associated geographical location and therefore routing the travel of workers between different locations is also required in a solution to these problems. The quality of a solution is measured in terms of the operational cost (travelling distance and staff cost), satisfaction of clients and workers preferences and the number of unassigned visits. This PhD thesis considers the *Home Healthcare (HHC) Problem* which is a type of WSRP in which the mobile workforce are care workers and nurses visiting patients at their homes in order to deliver medication, help with physical therapy and other daily tasks such as personal hygiene, etc. [23, 74]. The demand for home healthcare services has been growing in the UK since 1988 [222] and was accentuated during the last decade [172]. Hence, creating high-quality work plans addressing the various aspects mentioned above is challenging and increasingly important in the public and private healthcare sectors.

1.1 The Home Healthcare Project

For some of the period of this PhD studies we were engaged in a *Research and Development* (R&D) project in collaboration with Webroster Ltd., the industrial partner in this research, who provides enterprise resource planning, including workforce scheduling, to home healthcare providers including companies and local government authorities in the UK. Their existing information system collects all the problem-related data and provides an interface to assist human decision makers in the process of assigning workers to visits. We were in charge of developing the decision support module that couples well with the information management system being developed and maintained by the industrial partner.

This R&D project was part of a Knowledge Transfer Partnership (KTP), a program funded by the UK government that brings state-of-the-art academic knowledge into the industry. The KTP was a project between the ASAP research group in the University of Nottingham and the industrial partner; and its purpose was to develop an optimisation engine for tackling home healthcare planning scenarios in the UK. The author of this thesis was the research associate in this KTP, directly working in the company, collecting the data, defining specifications and implementing the solutions.

Although the KTP has resulted in a proprietary algorithmic solver employed by Webroster Ltd. (which is not presented in this thesis), the KTP allowed us to work closely with the owners of the data and therefore to further understand the problem. Even though the focus of the company was to obtain the solution for their optimisation problem, the KTP regulation ensured that scientific research was conducted, hence

part of this thesis is dedicated to the findings and contributions obtained during the project.

1.2 Overview of the Thesis

The research presented in this thesis has in its core improving the understanding of the HHC problem and the development of solution techniques. In order to achieve that goal, several paths were explored – from an analysis of real-world data provided by our industrial partner to the multiobjective nature of the problem and tailored heuristics algorithms.

1.2.1 Chapter 2

Chapter 2 starts by introducing basic concepts and background knowledge, including optimisation problems, baseline solving techniques such as exact methods and heuristics; several metaheuristics; and multiobjective optimisation and multiobjective optimisation algorithms.

1.2.2 Chapter 3

Then, the thesis focuses on presenting the HHC problem – Chapter 3 presents the mixed integer programming formulation and the definition of the HHC problem. It also introduces the real-world datasets provided by our industrial partner, an analysis of the features of the datasets and early results obtained with a simple constructive heuristic.

1.2.3 Chapter 4

Chapter 4 presents algorithms to calculate lower bound values for the problem, including the largest datasets where exact optimisation meth-

ods such as mathematical programming are not applicable.The algorithm splits the HHC problem into a scheduling and a routing subproblems and tackles each one individually. Results are presented, showing that the algorithm manages to find satisfactory lower bound values.

1.2.4 Chapter 5

Next on the thesis, the focus is the multiobjective aspects of the HHC problem proposed. Chapter 5 presents a novel analysis technique focused on highlighting non-obvious relationships between objectives in multiobjective optimisation problems. It also presents the application of the analysis techniques on three different multiobjective problems as to validate the approach itself.

1.2.5 Chapter 6

Chapter 6 presents the results of the multiobjective analysis to the multiobjective version of the HHC problem. The analysis shows that each dataset has a unique fitness landscape and the objectives have different degrees of importance across datasets. Hence, it is concluded that solving techniques based on tackling individual constraints might present different performance on different datasets. Also, it is clear that instances of the same dataset present similar fitness landscapes.

1.2.6 Chapter 7

Chapter 7 introduces a variable neighbourhood search with novel tailored local search neighbourhoods to solve the HHC problem. The algorithm comprises three local search structures, where two are independent of constraints (as suggested in Chapter 6) and the other ex-

plores constraints that widely appear as very important in the workforce scheduling and routing literature. A thorough experimentation of the proposed algorithms is presented.

1.2.7 Chapter 8

Chapter 8 presents a methodology to solve multiobjective problems by estimating the Pareto front using known instances of the dataset and employing goal programming with efficient single-objective algorithms.

1.2.8 Chapter 9

To conclude the thesis, Chapter 9 recapitulates the key aspects of the research and the contributions. Comments about the results obtained and the achievements of this research are provided. The chapter also presents suggestions and research questions for future work.

1.2.9 Appendices

The two first appendices of this thesis consist of the collaborative aspects of the company and the academics involved in this research. Appendix A presents a collaborative framework for the development of optimisation solutions for enterprise systems. Appendix B presents a methodology for the development of a data-driven, problem-centric application programming interface for optimisation problems. It also presents the experimental setup and results obtained for both the application programming interface and the proposed collaborative framework. Appendix C presents the scatter plots of the multiobjective analysis of the HHC datasets for all the instances with all objectives set as the pivot.

1.3 Contributions

The intended contributions of this thesis are listed below:

1. A new set of HHC problem instances from real-world home healthcare planning scenarios made publicly available to the research community.

2. A set of benchmark solutions to these problem instances obtained with heuristic and mathematical programming methods.

3. Algorithms to calculate lower bound values for the HHC problem, including the instances where exact methods are not applicable.

4. A technique to analyse and visualise global and local relationships between many objectives in order to achieve a clearer understanding of the fitness landscape in a multiobjective problem. The technique is shown to facilitate a comparison between the fitness landscapes of different problems, and the usefulness of this information for the crafting of solution algorithms is observed – given two problems with similar fitness landscapes, an efficient solution algorithm for the first problem is more likely to also be efficient for the second one, given that it has similar exploratory capability on both problems.

5. Improved insights on the generation of multiobjective benchmark scenarios are presented. It is shown that in order to generate a multiobjective problem benchmark dataset, it may not be sufficient to parametrise constraints to produce scenarios with distinct fitness landscapes. Instead, a combination of parametrised constraints and correlated data is shown to be more effective.

6. An improved understanding of the multiobjective nature of the read world HHC scenarios presented, obtained through an analysis of the objective relationships and the fitness landscape of the problem.

7. A variable neighbourhood search algorithm that produces benchmark best-known solutions for the real-world HHC scenarios with three domain-specific local search neighbourhoods used by the proposed algorithm which can be adapted to similar problems.

8. A novel methodology based on using goal programming and the proposed variable neighbourhood search to solve the multiobjective version of the studied HHC scenarios.

1.4 Publications

We list previous and planned publications directly related to the content of this thesis. In summary, there are six published papers arisen from the contributions of this thesis and three more that are being prepared.

1. R. L. Pinheiro, W. Laesanklang, D. Landa-Silva, R. Qu, R. Ward, and G. Doades. Benchmark dataset and results for a real-world workforce scheduling and routing problem in healthcare delivery. Submitted to *International Transactions in Operational Research*.

This article presents the HHC problem formulation, the datasets provided by our industrial partner and an analysis of the datasets using both mixed-integer programming and heuristic algorithms. It summarises the contents of Chapter 3.

2. R. L. Pinheiro, W. Laesanklang, D. Landa-Silva, A. A. Constantino. Lower bounds for a workforce scheduling and routing problem. *In preparation*.

This paper contains the algorithms presented in Chapter 4 to calculate lower bound values for the HHC problem.

3. R. L. Pinheiro, D. Landa-Silva, and J. Atkin. Analysis of objectives relationships in multiobjective problems using trade-off region maps. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 735–742, New York, NY, USA, 2015.

4. R. L. Pinheiro, D. Landa-Silva, and J. Atkin. A Technique Based on Trade-off Maps to Visualise and Analyse Relationships Between Objectives in Optimisation Problems. To appear in *Journal of Multi-Criteria Decision Analysis*. 2017.

These two publications summarise the contents of Chapter 5. Publication 3 presents the novel four-step analysis technique to identify and visualise relationships between objectives in a multiobjective optimisation problem. Publication 4 is an extended journal article that introduces improvements to the technique and presents the application of the technique to three different multiobjective problems.

5. 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: Proceedings of the 7th World Congress on Nature and Biologically Inspired Computing (NaBIC2015)*, volume 419 of *Advances in Intelligent Systems and Computing*, pages 247–259, Springer International Publishing, 2016.

Publication 5 presents an earlier version of the variable neighbourhood algorithm proposed in Chapter 7, containing two local search neighbourhoods and the results for the majority of the HHC datasets.

6. R. L. Pinheiro, and D. Landa-Silva. Using VNS on estimated Pareto fronts to solve a multiobjective home healthcare problem. *In preparation*.

This is an article containing the contents of Chapter 8, proposing the use of goal programming and single-objective heuristic algorithms to solve multiobjective problems based on an estimation of their Pareto fronts.

7. R. L. Pinheiro, and D. Landa-Silva. A development and integration framework for optimisation-based enterprise solutions. In *ICORES 2014 - Proceedings of the 3rd International Conference on Operations Research and Enterprise Systems, Angers, Loire Valley, France, March 6-8,* pages 233–240, 2014.

The collaborative environment between academics and the industrial partner led us to identify gaps in the literature where this kind of collaboration is required. Hence, publication 7 presents a collaborative framework to enable academics and researchers to achieve improved results on research and development projects for optimisation-based enterprise systems. The content of this paper is presented in Appendix A.

8. R. L. Pinheiro, D. Landa-Silva, Rong Qu, E. Yanaga, and A. A. Constantino. Towards an Efficient API for Optimisation Problems Data. In *Proceedings of the 18th International Conference on Enterprise Information Systems (ICEIS 2016)*, Vol. 2, pages 89–98. Scitepress, Rome, Italy, 2016.

9. R. L. Pinheiro, D. Landa-Silva, Rong Qu, A. A. Constantino, and E. Yanaga. An application programming interface with improved performance for optimisation problems data. In *Journal of Management Analytics*, Vol. 3, No. 4, pages 305–332. Taylor & Francis, 2016.

Publications 8 present a methodology for the development of an application programming interface for optimisation-based solutions. Publication 9 extends that work by including design patterns and techniques to be included in the application programming interface in order to increase its efficiency. The contents of both papers are presented in Appendix B.

Chapter 2

Background Knowledge

This chapter presents a series of basic concepts to enable the reader to understand the remaining of this thesis.

2.1 Optimisation Problems

An optimisation problem is the problem of finding the *best* solution from all feasible solutions.

Definition 1 (Optimisation Problem)**.** *Without loss of generality, let us consider a minimisation problem. Let* $\mathbb{X}$ *be a solution to the problem, comprising values assigned to the problem's decision variables,* $f(\mathbb{X})$ *be the objective function to be minimised over the variables in* $\mathbb{X}$ *and* $\mathbb{S}$ *be the set of feasible solutions, then an optimisation problem can be defined as:*

$$\begin{aligned} &\text{minimise} \quad f(\mathbb{X}) \\ &\text{subject to} \quad \mathbb{X} \in \mathbb{S} \end{aligned} \tag{2.1}$$

Optimisation problems consider either continuous or discrete decision variables. The problems described in this thesis employ discrete variables that can be represented as binary or integer values; and they

are represented in graph structures. These problems are also known as combinatorial optimisation problems.

2.1.1 Constraints

An optimisation problem is subject to a set of constraints, or rules, that can be either hard or soft. *Hard constraints* define $\mathbb{S}$ (Equation 2.2) and are conditions that have to be satisfied by a solution. A solution that satisfies all hard constraints is said to be *feasible*, while a solution that does not satisfy one or more hard constraints is an *infeasible solution*.

Soft constraints are conditions that we would like to satisfy, but are not necessary. These often include preferred features or desired aspects of the solution. A solution that does not meet one or more soft constraints but does not violate any hard constraints is still feasible. Soft constraints are often present as terms in the objective function adding penalty values when the condition of the soft constraint is not met.

2.1.2 Objective Function

While the hard constraints and the decision variables define the size of $\mathbb{S}$, the *objective*, or *fitness* function define how solutions in $\mathbb{S}$ compare to each other. Many problems have an objective function represented by a summation of the penalty values of its soft constraints. Additionally, the objective function may include elements of the solution in which we want to obtain the best possible values.

Because the objective function may comprise many terms, it is common to have a mechanism to prioritise more important objectives or features. The most common manner of achieving this is by means of a weighted function, in which terms of the objective function are multi-

plied by weight values such that more important objectives or soft constraints have a higher weight and less important soft constraints of objectives have a smaller weight.

2.1.3 Search Space and Objective Space

Two concepts are widely used when studying optimisation problems. That of search space and that of objective space, which refers respectively to the decision variables of a solution and the fitness of a solution.

The *search space* refers to the set or domain through which an algorithm searches. All permutations of values of the decision variables of a problem define the search space of that problem.

The *objective space* refers to the set or domain comprising the fitness values of all feasible solutions in the search space.

2.1.4 Computational Complexity

The size of $\mathbb{S}$ is often exponential (or factorial) in relation to the number of decision variables of the problem. Thus, using enumeration techniques to iterate through all solutions in $\mathbb{S}$ to find one with minimal costs is impracticable. Also, many optimisation problems are proven to be NP-Hard, hence there is no known algorithm with polynomial time complexity capable of solving them [57]. Therefore, among the methods and techniques to tackle optimisation problems, we highlight exact solving methods and heuristic algorithms.

2.2 Exact Solving Methods

Exact methods consist of solving techniques that are capable of finding optimal solutions and guaranteeing their optimality [38]. Even though most of them still have exponential complexity in the worst case, they are effective in solving optimisation problems of considerable size. Hence, because these methods guarantee optimality when studying a new problem, it is important to assess whether it can be solved by exact methods or not. Among the most common exact methods, we highlight linear programming and branch and bound methods.

Linear Programming is the technique consisting of formulating the optimisation problem as linear expressions and solve it using methods such as Simplex. When all variables in the model are restricted to integer values, the method can be referred as an Integer Programming (IP). If one or more variables in the model are continuous and assume real values, then the problem is a Mixed Integer Programming (MIP) and techniques such as Column Generation [105] can be effective to solve it.

Branch and Bound is a category of methods that divides a set of feasible solutions into smaller sets and evaluate them. The division is named *branching* as the algorithm builds a tree. The method calculates upper and lower bounds for the branches and further explores branches that could contain an optimal solution based on the current boundaries of all branches, hence potentially reducing the number of solutions evaluated. However, these methods tend to consume a large amount of physical memory during runtime, making them impracticable on larger problems. It escapes the scope of this thesis to further detail these methods. For that purpose we recommend the work of Lawler and Wood [143]

and Hillier and Lieberman [105].

2.3 Heuristic Algorithms

Exact methods tend to consume large amounts of physical memory and processing time, depending on the size of the problem. Therefore, on large problems, it is common that the resources available are insufficient. In these cases, the use of *heuristic algorithms* might be a viable option.

The word heuristic comes from the greek *heuriskein*, meaning "to find" or "to discover". The following definition can help us understanding what a heuristic is:

> "A heuristic technique (or simply heuristic) is a method which seeks good (i.e. near-optimal) solutions at a reasonable computation cost without being able to guarantee optimality, and possibly not feasibility. Unfortunately, it may not even be possible to state how close to optimality a particular heuristic solution is" [189].

Thus, a heuristic algorithm uses a methodology or information about the problem domain to substantially reduce the number of solutions explored during the search, while still attempting to obtain a near-optimal solution.

Unlike exact methods, there is no guarantee regarding the quality of the solutions obtained by heuristic techniques. Thus, when developing heuristics, researchers tend to empirically evaluate their quality, comparing the solutions against other known methods or against exact methods if those are available.

Heuristic methods can be categorised according to several criteria:

- **Nature Inspiration:** Many heuristics are inspired in natural events, such as the evolution of species, animal behaviour (ants, bees, etc.), immunology, physics and biology.

- **Memory usage:** Some heuristics use memory to create improved solutions. Hence, information stored in memory is used to create improved solutions.

- **Deterministic and stochastic:** A deterministic heuristic only makes deterministic decisions to obtain a solution, while a stochastic heuristic makes use of randomness. Hence, in deterministic heuristics, the multiple execution of a heuristic will always generate the same results, while on stochastic heurostics, the results may vary.

- **Population-based or single-solution**: Some heuristics work simultaneously on a set of solutions, while others work on a single solution at a time.

- **Constructive or local search**: Constructive heuristics construct a solution from scratch while improvement (local search) heuristics take an initial solution and attempts to improve it.

The categories listed are not comprehensive and they are not mutually exclusive, e.g. a constructive heuristic may use memory or not, or it may be deterministic or stochastic. This thesis uses constructive and local search to classify the heuristics (and later the metaheuristics) studied.

2.3.1 Constructive

Constructive heuristics are methods to build a solution from an empty state, systematically choosing the value of each decision variable until

a complete solution is obtained. Some search techniques benefit from a good starting solution, hence constructive heuristics are important in the context of optimisation problems. Also, constructive heuristics are often quick to execute because usually, they consist of a single pass through the decision variables.

2.3.2 Local Search

A local search method starts from a single current solution and iteratively performs small changes (or moves) in order to improve it. Usually, these changes affect a small part of the solution, hence it is said that the changes are *local*.

Neighbourhoods

The set of solutions that can be obtained by applying a single move to an existing solution $\mathbb{X}$ defines a *neighbourhood* $N(\mathbb{X})$ of that solution. Moves, or *neighbourhood structures*, can be general and applicable to many optimisation problems or very specific and related to the problem domain. Common general neighbourhood structures include:

- **Flip:** changes the value of a single decision variable; e.g. given the following solution composed by binary variables [1,0,1], the neighbourhood $N_{flip}([1,0,1]) = \{[0,0,1],[1,1,1],[1,0,0]\}$.

- **Swap:** swap the values of two decision variables; e.g. $N_{swap}([1,0,1]) = \{[0,1,1],[1,1,0]\}$.

- ***k*-opt:** given k decision variables, select the best values for all k variables using enumeration.

Hill Climbing

Hill Climbing (HC) is a search technique that takes a current solution, randomly generated or generated by a constructive heuristic, and performs a series of moves to the current solution in order to improve it. At each iteration, the HC evaluates the neighbourhood of the current solution and picks the best option. The algorithm stops when certain criteria are met or when no neighbour solution is an improvement to the current solution. When no neighbour solution presents improvements, we say that the current solution is a *local optimum*.

Because at each iteration the HC search only accepts better solutions, it is also called a *greedy heuristic*. We present a pseudocode for a general Hill Climbing local search:

Algorithm 1 Hill Climbing Local Search

Require: initial solution $\mathbb{X}$, neighbourhood structure N
Ensure: local optimum
1: **for** $\mathbb{X}' \in N(\mathbb{X})$ **do**
2: **if** $f(\mathbb{X}') > f(\mathbb{X})$ **then**
3: $\mathbb{X} \leftarrow \mathbb{X}'$
4: **end if**
5: **end for**
6: **return** $\mathbb{X}$

2.4 Metaheuristics

The term *metaheuristic* was first coined by Glover [89] and can be defined as:

> "a high-level problem-independent algorithmic *framework* that provides a set of guidelines or strategies to develop heuristic optimisation algorithms. The term is also used to refer

> to a problem-specic implementation of a heuristic optimisation algorithm according to the guidelines expressed in such a framework" [204].

Metaheuristics present a sound option to solve optimisation problems when exact methods are not an applicable alternative. An algorithm designer can easily search the literature to find metaheuristics applied to related problems and pick a set of suitable metaheuristics to experiment with.

One manner to categorise metaheuristics is according to how they manipulate solutions. Local search metaheuristics perform small operations to improve a single current solution. Constructive metaheuristics build solutions following a set of rules. Population-based metaheuristics work on a set of solutions and combine them to improve their overall quality. Additionally, hybrid metaheuristics combine different approaches to form a class of its own.

2.4.1 Local Search

Analogously to the local search heuristics described in Section 2.3, the local search metaheuristics start from an initial solution and iteratively perform changes on the solution in order to improve it. However, there is a difference: basic local search heuristics tend to get stuck into local optima while most metaheuristics address this problem. Take for instance Figure 2.1. In order for a local search to reach the global optimum (green dot), the initial solution must be between the valleys before and after it, otherwise the local search will get stuck in one of the local optima.

Some local search metaheuristics approach that issue by including a stochastic mechanism, allowing the search to accept worse solutions in

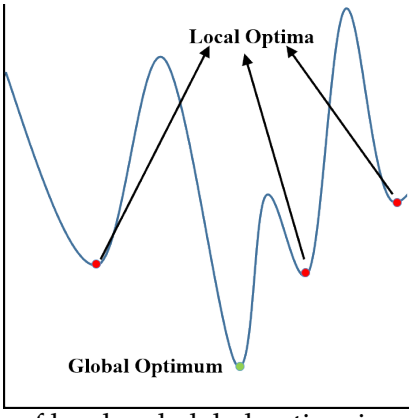


Figure 2.1: Example of local and global optima in a minimisation problem.

order to escape from local optima and continue the search on other regions of the search space aiming to find a global optimum or at least improved solutions. The literature presents several metaheuristics that are not part of the scope of this thesis. We highlight the Iterated Local Search [206], which consists of obtaining successive locally optimal solutions by successively performing a local search and a perturbation procedure, which modifies the current solution (possibly worsening it) to escape local optima; the Simulated Annealing, initially proposed by Kirkpatrick et al. [128], which emulates the annealing process of cooling crystalline solids; and the Tabu Search [89], which is a local search metaheuristic that uses memory to escape from local optima and plateaus (regions of the solution space where many solutions are similarly fit).

2.4.2 Variable Neighbourhood Search

The idea of systematically combining distinct neighbourhood structures to improve local search algorithms has been present in the literature for a long time [103, 118, 191]. In particular, when describing Tabu Search, Glover [90] mentions the use of multiple neighbourhoods in a hill climbing local search. Later, Mladenović and Hansen [164] formalised the con-

Algorithm 2 General Variable Neighbourhood Search

Require: Initial solution $\mathbb{X}$. Set of shaking neighbourhood structures N_k, for $k = 1, \ldots, k_{max}$.

while stop criteria is not met **do**
  for $k \leftarrow 1$ **to** k_{max} **do**
    $\mathbb{X}' \leftarrow$ random neighbour from the kth neighbourhood $N_k(\mathbb{X})$
    $\mathbb{X}'' \leftarrow VND(\mathbb{X}')$
    if $f(\mathbb{X}'') < f(\mathbb{X})$ **then**
      $\mathbb{X} \leftarrow \mathbb{X}''$
      $k \leftarrow 1$
    else
      $k \leftarrow k + 1$
    end if
  end for
end while
return $\mathbb{X}$

cept into a metaheuristic entitled *Variable Neighbourhood Search* (VNS), that starts from an initial solution (obtained by any mean) and performs successive local searches using multiple neighbourhoods to improve the solution. In order to avoid local optima, it randomly disturbs the current solution (possibly making it worse) at the end of each iteration. In this work, we use the general VNS framework which utilises a Variable Neighbourhood Descent (VND) as the local search mechanism. The VND is a simpler version of the VNS that does not possess the stochastic factor, hence, it cycles through a set of neighbourhoods accepting improvements until it reaches a local optimum.

Notably, the VNS uses two sets of neighbourhoods structures. The first one, N_k, is composed of *shaking neighbourhoods* which are designed to disturb the current solution, possibly worsening it, in order to escape from local optima. The second set, N_l, is composed of *local search neighbourhoods* that are used to improve the solution. Algorithm 2 and Algorithm 3 present the pseudocodes for both the VNS and the VND respectively.

Algorithm 3 Variable Neighbourhood Descent

Require: Initial solution $\mathbb{X}$. Set of local search structures N_l, for $l = 1, \ldots, l_{max}$.

while there is improvement **do**
  for $l \leftarrow 1$ **to** l_{max} **do**
    $\mathbb{X}' \leftarrow$ best neighbour of $\mathbb{X}$ using the neighbourhood N_l
    if $f(\mathbb{X}') < f(\mathbb{X})$ **then**
      $\mathbb{X} \leftarrow \mathbb{X}'$
      $l \leftarrow 0$
    else
      $l \leftarrow l + 1$
    end if
  end for
end while
return $\mathbb{X}$

One of the reasons that make this metaheuristic widely employed in the literature is that it allows a large degree of flexibility for domain-specific algorithmic components. The shaking neighbourhoods can be general or domain-specific and they are not necessarily related to the local search neighbourhoods. Several domain-specific shaking neighbourhoods and several local search neighbourhoods can be combined to provide the algorithm improved exploratory capabilities.

2.4.3 Constructive

Analogously to the constructive heuristics presented earlier in this chapter, the constructive metaheuristics generate new solutions without requiring previous solutions. This is usually done by procedurally assigning values to the decision variables attempting to obtain a near-optimal solution at the end. To improve the quality of the final solutions, most constructive heuristics use some sort of local search at the end. It is not in the scope of this thesis to survey constructive metaheuristics as they are not applied in this work. As examples of constructive meta-

heuristics we highlight the Greedy Randomised Adaptive Search Procedure (GRASP), proposed by Resende and Ribeiro [190], which is a metaheuristic that successively generates randomised solutions and performs a local search on each generated solution; and the bio-inspired metaheuristic (metaheuristics inspired in natural phenomena) Ant Colony Optimisation (ACO), which is based on the foraging behaviour of real-life ants [68].

2.4.4 Population-Based

Population-based metaheuristics use a set of solutions (often called population) and combine features from solutions in the population to generate new solutions and improve the overall quality of the population. Because population-based metaheuristics naturally deal with a multitude of solutions, they're often employed to solve multiobjective optimisation problems. The most common population-based metaheuristics are the evolutionary algorithms (EA) which mimic basic principles of natural evolution.

Genetic Algorithms

The Genetic Algorithm (GA) [92] is possibly the most common evolutionary algorithm studied in the literature. It is based on concepts of evolutionary biology such as heredity, mutation, natural selection and recombination. In order to escape from local optima and plateaus, the GA also possesses a stochastic factor allowing it to accept worse solutions and explore different regions of the search space. A basic GA is composed of five components: the individual, the selection, crossover and mutation genetic operators, and the population scheme.

Individual. A solution in the context of GAs is called an *individual*.

Each individual contains its genetic code, or *chromosome* (which consists of the set of decision variables or attributes that compose a solution). Each element in the chromosome is called a *gene*. The *encoding of individuals* is the scheme that translates a solution from the optimisation problem to the chromosome and vice-versa. A good encoding allows an efficient use of the genetic operators, while a bad encoding has the opposite effect. Moreover, usually the chromosome are encoded as binary or integer variables, depending on the problem domain.

Selection Operator. The *selection* genetic operator mimics the behaviour of natural selection. In every *generation* (an iteration of the main GA loop), a set of *parents* (individuals currently present in the population) are selected to generate *offspring* (new solutions). Any number of parents can be selected to generate new offpring, but usually two parents generate two offspring. In general terms, the selection mechanism tends to benefit more fit solutions over less fit, but not completely ignoring the less fit.

There are many common mechanisms used for selection, including:

- Roulette: a probability is attributed to each individual in the current population. The probability is based on the fitness of the individual such that more fit individuals have a higher probability. Then, the list of parents are selected by randomly drawing individuals from the population according to their probabilities. An individual may be selected multiple times.

- Tournament: to select a parent, a number (usualy two or three) of individuals are randomly drawn from the population. The most fit of the individuals drawn is the selected parent. Tournament presents some advantages over the roulette method. First, it it

not necessary to compute the probability of each individual in the population, hence the method requires less computational effort. Second, you can easily influence the 'pressure' towards more fit solutions by increasing the number of individuals drawn in each tournament.

- Ranking: a ranking is created and individuals in the population fall into different ranks. Each rank has a fixed probability which is uniformly distributed among its members. The parents are randomly drawn following the probability given by their rank.

Crossover Operator. After the selection, each set of parents is combined to generate new individuals. Usually two parents generate two offspring individuals. The genes from the parents are transcribed into the individuals according to some mechanism. Following, we present three common crossover mechanisms:

- One Point: the one point crossover simply selects a random *cutting point* anywhere after the first gene and before the last gene. The first offpring takes genes from the first parent until the cutting point, and from the second parent after the cutting point. The second offspring takes the genes from the other parent, i.e. from the second parent until the cutting point and from the first parent after the cutting point. Figure 2.2 illustrates this crossover operation.

- Two Points: the two points crossover follows a similar idea to the one point crossover, but instead of having one cutting point, it now has two. In fact, there are variants with any number of cutting point, however one and two are more common in the literature. In this crossover, two distinct cutting points are selected randomly.

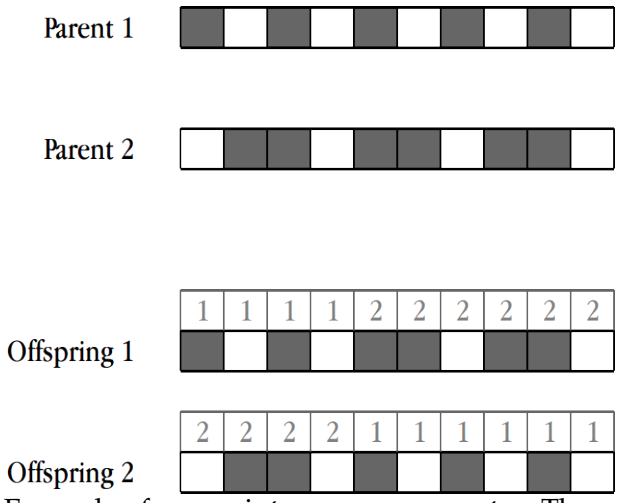


Figure 2.2: Example of one point crossover operator. The cutting point is after the fourth gene. The number above each offpring's gene indicates which parent provided that gene.

The first offspring takes the genes from the first parent until the first cutting point and after the second cutting point, and the genes from the second parent between cutting points. The second offspring takes the genes from the other parent. Figure 2.3 illustrates this operator.

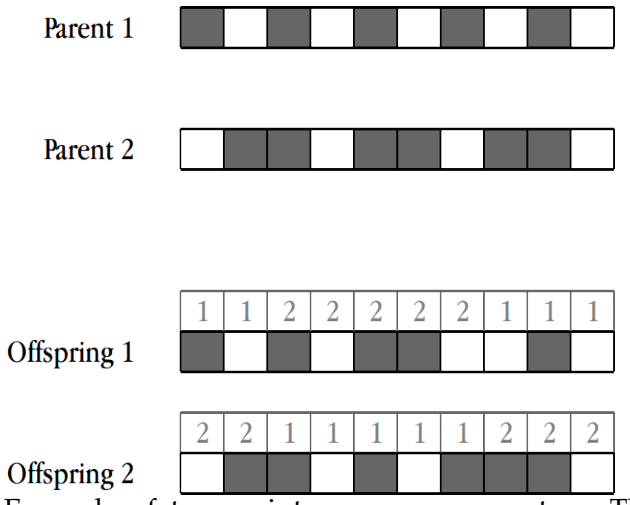


Figure 2.3: Example of two points crossover operator. The cutting points are after the second and seventh genes. The number above each offpring's gene indicates which parent provided that gene.

- Uniform: the uniform crossover brings a higher degree of randomness. In this crossover, the genes for the first offspring are

randomly drawn from the parents. The second offspring gets the genes from the other parent, hence when the first offspring takes a gene from the first parent, then the second offspring will take the same gene from the second parent and vice-versa. Figure 2.4 illustrates the uniform crossover.

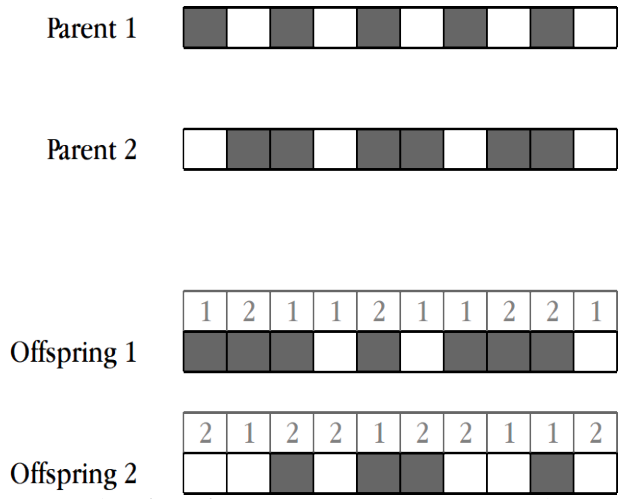


Figure 2.4: Example of uniform crossover operator. The number above each offpring's gene indicates which parent provided that gene.

Mutation Operator. As generations evolve, the best genes are kept in the population and bad genes cease to exist. This can lead to the algorithm getting stuck into local optima. Mutation operators have a probability of making small changes to the offspring individuals in order to introduce diversity into the population. Two of the most common mutation operators are:

- Flip: a random gene has its value altered to the opposite value (in case of binary encoding) or to a random value.
- Swap: two random genes have their values swapped.

Population Replacement Strategy. After the selection, crossover and mutation are performed, the new offspring individuals replace the in-

dividuals in the population taking their places. Few strategies can be taken, such as:

- Classic: the new population, consisting of the offspring, completely replaces the former population.
- Best Fit: the new population is formed only with the best fit individuals from both the current population and the offspring individuals. This leads to a faster convergence, but potentially makes the algorithm more prone to be stuck in local optima.
- Elitist: a percentage of the X% best fit individuals from the current population is kept. The remaining is replaced by the offspring.

Note that we presented only a few basic components and strategies for a classic GA. The literature presents many other operators as these can be tailored according to the problem domain. Following, Algorithm 4 presents the pseudocode of the main loop of a GA.

Algorithm 4 Genetic Algorithm

1: $P_{current} \leftarrow generateInitialPopulation()$
2: $\mathbb{X} \leftarrow getBestIndividual(P_{current})$
3: **while** stop criteria is not met **do**
4: $\quad Parents \leftarrow selectParents(P_{current})$
5: $\quad P_{offspring} \leftarrow doCrossover(Parents)$
6: $\quad P_{offspring} \leftarrow doMutation(P_{offspring}, mutation_probability)$
7: $\quad P_{current} \leftarrow updatePopulation(P_{current}, P_{offspring})$
8: $\quad \mathbb{X} \leftarrow getBestIndividual(P_{current}, \mathbb{X})$
9: **end while**
10: **return** $\mathbb{X}$

2.5 Multiobjective Optimisation Problems

Essentially, a *multiobjective optimisation problem* (*MOP*) is an optimisation problem that considers multiple objective functions to be optimised si-

multaneously.

Definition 2 (Multiobjective Optimisation Problem)**.** *Without loss of generality consider a minimisation problem. Let* $\mathbb{X}$ *be any solution for the problem,* $\mathbb{S}$ *be the feasible region delimited by the problem constraints and* $\Pi = (f_1, f_2, ..., f_m)$ *be the set of objective functions, then a multiobjective optimisation problem can be defined as:*

$$\begin{aligned} &\text{minimise} \quad \vec{f}(\mathbb{X}) = (f_1(\mathbb{X}), f_2(\mathbb{X}), ..., f_m(\mathbb{X})) \\ &\text{subject to} \quad \mathbb{X} \in \mathbb{S} \end{aligned} \tag{2.2}$$

A straightforward way to tackle MOPs is to tackle them as single-objective problems by scalarising the solution-vector $\vec{f}$ using a *utility function* $u : \mathbb{R}^n \to \mathbb{R}$ and a weight vector (or *utility vector*) $\vec{w} = (w_1, w_2, ..., w_m)$ such that $u(\vec{f}) = (f_1(\mathbb{X}) \times w_1 + f_2(\mathbb{X}) \times w_2 + ... + f_m(\mathbb{X}) \times w_m)$ and then solving the problem as a single-objective problem [22]. This is a common approach and has been widely applied in the literature [28].

2.5.1 Pareto Dominance

Knowing the weight vector $\vec{w}$ is not always possible as the weights might not be available or are not easily obtainable (specially with a large set of objectives), hence the need for other techniques. An alternative is to stipulate a relation between objective vectors such that they can be comparable. The most common is the domination concept, proposed by Pareto [171]:

Definition 3 (Pareto Dominance)**.** *Given two objective values vectors* $\vec{u} = (u_1, u_2, ..., u_m)$ *and* $\vec{v} = (v_1, v_2, ..., v_m)$*,* $\vec{u}$ *dominates* $\vec{v}$*, denoted by* $\vec{u} \succ \vec{v}$*, if and only if* $\forall i \in \{1, 2, ..., m\}$*,* $u_i \leq v_i$ *and* $\exists j \in \{1, 2, ..., m\}$ *such that* $u_j < v_j$*.*

The Pareto dominance relation allows the comparison of pairs of vectors, hence it is possible to determine when a solution is better than another. However, the dominance relation may not provide a single optimal solution, because a set of solutions may be non-dominated among themselves. Hence, when using Pareto dominance, the aim of tackling MOPs is to obtain the set of all non-dominated solutions called *optimal Pareto front*.

Obtaining the optimal Pareto front is often an NP-Hard problem, hence many works in the literature solve MOPs by heuristically obtaining an approximation set of the optimal Pareto front [63, 148, 218]. The algorithms to tackle MOPs employed in this thesis are of heuristic nature.

Additionally, when the number of objectives in a MOP increases, the number of non-dominated solutions can potentially grow exponentially [76, 80, 185]. When the number of objectives is high (more than three), search algorithms find severe difficulties to spread the non-dominated approximation set throughout the optimal Pareto front, instead, they usually converge to certain regions (possibly sub-optimal). Laumanns et al. [142] tackled this problem with the improved ϵ-dominance relation:

Definition 4 (ϵ-Dominance)**.** *Given* $\epsilon > 0$ *and two objective vectors,* $\vec{u} = (u_1, u_2, ..., u_m)$ *and* $\vec{v} = (v_1, v_2, ..., v_m)$, $\vec{u}$ *ϵ-dominates* $\vec{v}$ *if and only if* $\forall i \in \{1, 2, ..., m\}$, $(1 + \epsilon)u_i \leq v_i$.

A higher value of ϵ reduces the number of solutions in an approximation set because any two non-dominated solutions that are close enough are ϵ-dominated. That can potentially spread the approximation set throughout the objective space, however when the value of ϵ is high, so is the probability of better solutions being left out. Thus, the efficiency of using

the ϵ-dominance alone decays as the number of solutions and objectives grows because ϵ cannot be high enough to compensate the size of the optimal Pareto front.

Deb et al. [63] improved the ϵ-dominance concept and introduced the ϵ-box dominance:

Definition 5 (ϵ-box Dominance). *Given $\epsilon > 0$ and two vectors, $\vec{u} = (u_1, u_2, ..., u_m)$ and $\vec{v} = (v_1, v_2, ..., v_m)$, $\vec{u}$ ϵ-box dominates $\vec{v}$, denoted by $\vec{u} \succ_\epsilon \vec{v}$ if and only if one of the following occurs:*

$$\begin{aligned} &1. \quad \lfloor \frac{\vec{u}}{\epsilon} \rfloor \succ \lfloor \frac{\vec{v}}{\epsilon} \rfloor, \text{ or} \\ &2. \quad \lfloor \frac{\vec{u}}{\epsilon} \rfloor = \lfloor \frac{\vec{v}}{\epsilon} \rfloor \text{ and } \|\vec{u} - \epsilon \lfloor \frac{\vec{u}}{\epsilon} \rfloor\| < \|\vec{v} - \epsilon \lfloor \frac{\vec{v}}{\epsilon} \rfloor\| \end{aligned} \tag{2.3}$$

The ϵ-box dominance divides the objective space into "hyper-boxes" where at most one solution from each hyper-box can be in the approximation set. The first condition ensures that if a vector $\vec{u}$ belonging to the hyper-box H_1, ϵ-dominates the vector $\vec{v}$ belonging to the hyper-box H_2 and $H_1 \neq H_2$, then $\vec{u} \succ_\epsilon \vec{v}$. Condition 2, however, ensures that if both solutions belong to the same hyper-box, such that $H_1 = H_2$, and $\vec{u}$ is closer (using Euclidean distance) to the optimal corner of the hyper-box than $\vec{v}$, then $\vec{u} \succ_\epsilon \vec{v}$.

2.5.2 Decision Making Process

The goal when solving MOPs is to find the optimal Pareto front and not a single solution (unless the front is composed of a single solution), hence, the decision-making process of real world problems is different. Usually, a human *decision maker* (DM) is presented with an approximation set (or a representative subset of the approximation set) and choose between solutions. Clearly, when the number of solutions is high, picking the

best one is non-trivial. Three different methods of using the DM input are commonly used in the literature.

A Priori

A priori methods [110] require the DM to provide information on the preferences of objectives or criteria before the solving process. The most common a priori technique is the use of a utility function as presented earlier in the chapter. The DM presents a weight value for each objective and that weight vector is used by a single-objective optimisation algorithm using a weighed sum of objectives as the objective function. An alternative is the use of *goal programming*, in which target values are set for each criterion and single-objective algorithms attempt to find the solution closest to the target values as possible. The field of Multicriteria Decision Making (MCDM) is a well-established research area and one of its goals is to extract preferences from decision-makers in order to aid in the decision-making process. Several works try to obtain the utility function from the decision maker in order to apply a single-objective optimisation algorithm [121, 122, 147].

However, this approach might be impracticable because it is assumed that a utility function does exist and that it can be represented by a linear function (the use of a non-linear function is possible but it restricts the choice of solving techniques). Also, it is required that the DM is able to accurately classify the objectives in order to approximate the utility function.

A Posteriori

The opposite concept produced the *a posteriori* methods [110]. This technique requires the DM input after the solving process. Therefore, an ap-

proximation set of the optimal Pareto front is presented to the DM and they can choose the most appropriate solution by evaluating the available options or using any methodology they prefer.

We highlight two problems with this approach:

1. On large problems, it is possible to have much more non-dominated solutions than a human is able to evaluate. Hence, additional techniques may be required to filter the approximation set and provide the DM with a reduced set with the most interesting options. Because "most interesting" is subjective, clearly deciding which solutions falls in this category is no easy task.

2. A posteriori multiobjective optimisation algorithms are shown to present inferior performance when exploring the objective space of problems with more than three objectives [87].

Interactive

Obtaining the user preferences a priori is, at best, a difficult goal to achieve. One alternative that has gained attention recently is the *interactive* method of decision-making [29], which conveys that the user preferences should be input during the optimisation process.

In the interactive decision making, the solving process starts with minimal input from the DM (just like in the a posteriori method). When certain criteria are met, the process stops and some input is required from the DM. It may be required that the DM choose between a subset of solutions or interesting features. The optimisation is then resumed using the new information to guide the search towards new interesting solutions. This interactive process can be repeated a number of times.

The argument for this method considers that, intuitively, it is easier

to choose between a small set of solutions than to attribute individual scores [30, 93]. Giving scores to solutions imply that the DM can accurately evaluate them while picking one out of a small set can express the subjectivity of the choice process – a DM may decide one solution over another but he may not know to quantify the decision or criteria. However, interactive methods are less straightforward to implement and assess because of the constant communication with the DM, which causes the method to be highly dependent on the interface provided. Additionally, online learning mechanisms are often required to enable the search algorithms to explore the regions in the objective space that the DM finds interesting.

2.6 Many-objective Optimisation

The literature conveys that when a problem has more than three objectives, it is categorised as a *many-objective problem*. The specific research on many-objective problems is quite new in the literature with the term "many-objective" being coined around 2002 [75]. Initially, these techniques were developed to solve continuous optimisation problems [75, 76, 201, 225] or the DTLZ benchmark problems [62, 85, 213].

In recent advances, many-objective algorithms have been applied to discrete combinatorial optimisation problems such as a car controller optimisation problem [167] and water resource optimisation [188].

Even though there has been progress on solving problems with more than three objectives in recent years, still the current algorithms find it difficult to provide a good set of solutions as the number of objectives grows [87].

2.7 Objective Relationships in Multiobjective Optimisation

The understanding of objectives and their relationships represents important information on the choosing of decision-making methods and solving techniques.

Purshouse and Fleming [183] proposed the use of the Kendall correlation metric [125] to classify the relationship between pairs of objectives as conflicting, harmonious or independent. The Kendall correlation value can be calculated as follows: given an approximation solution set and any two objectives Z_i and Z_j where $i \neq j$, calculate the number of *concordant* pairs μ_c and *discordant* pairs μ_d. A pair of solutions (a, b) is concordant when $Z_i^a > Z_i^b$ and $Z_j^a > Z_j^b$ (where Z_x^y is the value of objective x for solution y), or when $Z_i^a < Z_i^b$ and $Z_j^a < Z_j^b$. A pair of solutions is discordant when $Z_i^a > Z_i^b$ and $Z_j^a < Z_j^b$, or when $Z_i^a < Z_i^b$ and $Z_j^a > Z_j^b$. If μ is the total number of solutions in the approximation set, then the Kendall correlation coefficient τ can be calculated as follows:

$$\tau = \frac{\mu_c - \mu_d}{\frac{1}{2}\mu(\mu - 1)}$$

2.7.1 Conflicting Relationships

A *conflicting relationship* between two objectives exists when the increase of one criterion across a multitude of solutions incurs the decrease of the other criterion. We say any two objectives are conflicting using the Kendall correlation when the coefficient value is close to -1. Highly conflicting problems are computationally difficult and intractable, as each pair of conflicting objectives exponentially increases the number of non-dominated solutions in the optimal Pareto Front [183].

Therefore, most benchmark problems proposed in the literature present a highly conflicting nature, such as the DTLZ benchmark [62, 85, 213] and the ZDT benchmark [50]. Additionally, most multiobjective algorithms are tailored to perform better on highly conflicting problems.

2.7.2 Harmonious Relationships

A relationship between two objectives is *harmonious* when the increase of one objective across multiple solutions incurs the increase of the second objective. The Kendall correlation coefficient for harmonious objectives is close to 1. In the case of highly harmonious relationships, one of them can potentially be removed from the set of objectives without interfering with the optimal Pareto front, hence the existence of harmonious relationships does not necessarily incur in an increase of computational difficulty or the size of the optimal Pareto front [184].

2.7.3 Independent Relationships

When the relationship between two objectives is neither harmonious or conflicting, it is said that there is no relationship, or that the objectives are *independent*. In terms of Kendall correlation, this means values close to 0.

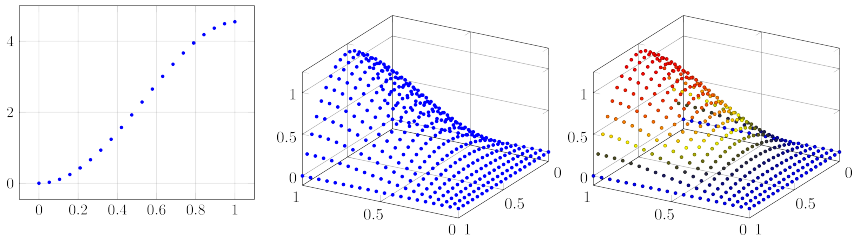
Purshouse and Fleming [184] show that if the decision variables can be split according to the objectives (or group of objectives), a divide-and-conquer strategy could be applied to improve the results.

2.8 Multiobjective Visualisation Techniques

To help understanding MOPs, several techniques are available to enable us to visualise solution sets. We list the most widely found in the literature.

2.8.1 Scatter-Plots

A scatter-plot is a plot graph of an approximation solution set, such that each solution is a point in the n-dimensional space (for a MOP with n objectives). Obviously, scatter-plots perform better when the number of objectives is two or three. Colours are often used to encode the fourth dimension.



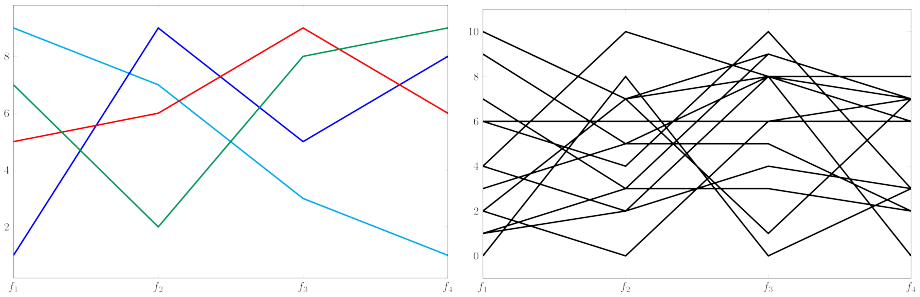
(a) 2D **(b)** 3D **(c)** 4D (using colours as the fourth dimension.

Figure 2.5: Example of scatter plots for two, three and four dimensions.

Figure 2.5 shows three scatter-plots. In (a) it shows a scatter-plot for an approximation set with two objectives represented by a cartesian graph. In (b) we have an approximation set with three objectives represented in a 3D graph. Finally, in (c) we have an approximation set for four objectives using colours as the fourth dimension. Scatterplots are possibly the visualisation technique most widely employed in the literature [84, 214, 219].

2.8.2 Parallel Coordinates

Another widely employed visualisation mechanism is the parallel coordinates graph. The graph is defined by a line chart, where each line corresponds to an objective vector. The y axis defines the objective values and the x axis defines each objective. When the objectives are incommensurable (for example, when comparing currency with distances), it is common practice to normalise the values.



(a) Four solutions **(b)** 14 solutions

Figure 2.6: Example of a parallel coordinates graph considering four objectives.

Figure 2.6 present two examples of parallel coordinates graphs. Both charts consider four objectives. On (a) we have four solutions, each one represented by a different colour. The graph allows a prompt comparison of the objective values. One notable characteristic of parallel coordinates graphs is that they scale well with increased number of objectives because all that is required is a new value in the x axis. However, the technique makes it difficult to visualise a large number of solutions. Take for example Figure 2.6 (b) where only 14 solutions turns the graph much difficult to read.

2.8.3 Heatmaps

The use of heatmaps in the multiobjective optimisation literature was proposed by Pryke et al. [182] and soon became popular. The method

enables the visualisation of large solution sets and a large number of objectives. The heatmap is usually a rectangular table, where rows correspond to solutions and columns to objectives. Each cell contains only a coloured background where the colour itself represents the data.

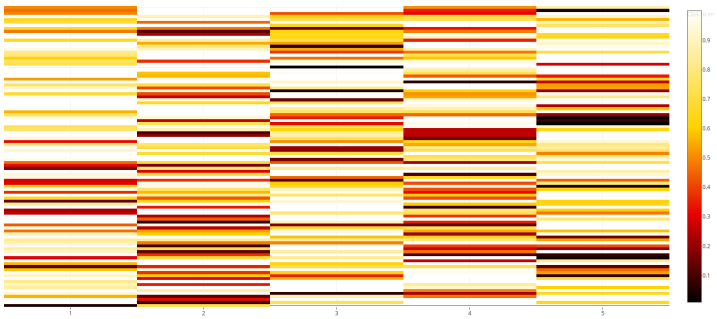


Figure 2.7: An example of heatmap for five-objective solutions where each column represent an objective and each row a solution.

Figure 2.7 presents an example of a heatmap for solutions of a five-objective problem. Although the technique allows the visualisation of large data, it is often difficult to extract relevant information just by using heatmaps. Also, the order of the solutions in the heatmap can highlight or make it difficult to spot interesting features. Nonetheless, they became popular in the recent literature [214].

2.8.4 Other Visualisation Mechanisms

Several less used visualisation techniques were proposed in the literature:

- distance and distribution charts;
- interactive decision maps;
- hyper-space diagonal counting;
- two-stage mapping;
- level diagrams;

- hyper-radial visualisation;
- Pareto shells;
- principal component analysis;
- self organising maps;
- seriated heatmaps; and
- radar charts.

It is not in the scope of this thesis to survey all visualisation techniques. For more information on the subject, we recommend the work of Korhonen and Wallenius [132], Lotov and Miettinen [150], Walker et al. [219] and Tusar and Filipic [214].

2.9 Solving Methodologies

Part of the thesis focus on a novel technique to analyse relationships between objectives in a MOP. Later, this technique is applied to analyse the objective relationships of the home healthcare scenarios presented in Chapter 3. In order to perform the analysis, approximation solution sets were required, hence we employed well known multiobjective algorithms. Because the aims of this thesis do not include new multiobjective algorithmic components, we focus the survey of multiobjective algorithms on the ones used in the following chapters.

Because typically evolutionary algorithms tackle entire populations of solutions, it is natural that in the multiobjective literature these techniques are the most widely spread. We now briefly list the ones most commonly found in the literature:

- **First Generation Algorithms:** Vector Evaluated Genetic Algorithm (VEGA) [197] MultiObjective Genetic Algorithm (MOGA) [81], Nondominated Sorting Genetic Algorithm (NSGA) [205] and the Nitched Pareto Genetic Algorithm (NPGA) [106]. These algorithms were the pioneers of the field being the first adaptations of genetic algorithms to the multiobjective context using dominance operators.

- **Second Generation Algorithms:** Zitzler et al. [230] showed that elitism helps on achieving better convergence for multiobjective evolutionary algorithms, hence several authors incorporated elitism and other features found in the state of the art single objective algorithms. We list the Strength Pareto Evolutionary Algorithm (SPEA) [229], the Simple Evolutionary Algorithm for Multiobjective Optimization (SEAMO) and the Nondominated Sorting Genetic Algorithm 2 (NSGA-II) [61].

- **Third Generation Algorithms:** the more recent multiobjective evolutionary algorithms try to handle deterioration, large problem sets, maintain diversity and spread the resulting solutions through the optimal Pareto set. For that, they employed archives, different dominance relations and other advanced mechanisms. We highlight the Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D) [227], the ϵ-NSGA-II [63] the SPEA2 [231], the SEAMO2 [165] and the EMOSA [148].

- **Indicator-Based Algorithms:** these are algorithms based on indicator-based techniques, such as the volume dominance concept. We highlight the works of Brockhoff and Zitzler [34], Emmerich et al. [71], Igel et al. [112] and Le [144].

- **Many-Objectives Algorithms:** recently more attention has been

given to problems containing more than three objectives. Among these algorithms we highlight the S Metric Selection Evolutionary MultiObjective Algorithm (SMS-EMOA) [24], the Control of the Dominance Area of Solutions (CDAS) [196], the Hypervolume Estimation Algorithm for Multiobjective Optimisation (HypE) [12], the auto-adaptive multiobjective evolutionary algorithms framework Borg [99] and the more recent NSGA-III [60].

2.9.1 Nondominated Sorting Genetic Algorithm 2

The Nondominated Sorting Genetic Algorithm 2 (*NSGA-II*) [61] is a Pareto-based genetic algorithm that keeps an archive of non-dominated solutions. Its main features include:

- a procedure to classify solutions according to their non-dominance level;
- an elitism mechanism in the form of a non-dominated archive to enhance convergence; and
- it uses crowding distance to guarantee diversity and to spread the solutions across the objective space.

The NSGA-II is possibly the most employed multiobjective algorithm in the literature. In the context of scheduling and routing problem, the NSGA-II was successful to many problems, including:

- A three-objective parallel machine scheduling problem, where a mutation algorithm based on the type of problem was included in the NSGA-II [15].
- A two-objective task scheduling for computational grids where the NSGA-II is used with tailored Fuzzy adaptive operators [194].

- A two-objective security driven scheduling for computational grids where a modified NSGA-II is proposed [123].
- A job scheduling in computational grids with three conflicting objectives [124].
- A two-objective green vehicle routing problem where an NSGA-II was implemented to reduce travel costs and CO_2 emissions [115].
- A two-objective commodity vehicle routing problem, where a modified NSGA-II with a tailored mutation operator was proposed to minimise distances and maximise the potential customers demand coverage [199].
- A two-objective vehicle routing problem where a tailored variant of the NSGA-II was used to reduce distances and the number of vehicles [48].

Note that this list is not extensive and the literature features several additional works. Our goal by providing these examples is to justify the use of the NSGA-II in the experiments presented in the following chapters.

2.9.2 Multiobjective Evolutionary Algorithm Based on Decomposition

The Multiobjective Evolutionary Algorithm Based on Decomposition (*MOEA/D*) [227] is an algorithm that decomposes the MOP into N scalar optimisation subproblems and solves them simultaneously by evolving a population of individuals. The population is composed of the best individuals found so far for each subproblem. The subproblems are defined by aggregating coefficient vectors. The MOEA/D features:

- lower computational complexity when compared to NSGA-II, especially when the number of non-dominated solutions in the objective space is too large;
- efficient handling of diversity because of the coefficient vectors; and
- the computational complexity is not largely affected by tackling many-objectives.

The MOEA/D has also been widely applied to the literature and it is one of the best-performing algorithms. In regard to the scheduling and routing problems, we highlight the following works:

- A three-objective job shop scheduling problem where a MOEA/D was used with an adaptive mechanism to select genetic operators to minimise the completion time, the total flow time and the tardiness time [228].
- A two-objective (makespan and energy consumption) job scheduling for heterogeneous computing systems was solved using a standard MOEA/D [65].
- A three-objective vehicle routing problem where a hybrid MOEA/D using local search was employed to minimise total distance travelled, maximise the route balancing between vehicles and minimise the number of vehicles [131].
- A two-objective multicast routing problem with network coding was solved by a MOEA/D that employed a tailored mechanism to update the population to address the issue of lack of feasible solutions [221].

Chapter 3

The Home Healthcare Problem

In workforce scheduling and routing problems (*WSRP*), a mobile workforce must perform tasks in scattered geographical locations. Solving these problems require defining schedules and a route plan for each worker such that all visits (when possible) are covered. These problems combine features from both scheduling and routing problems, making them very challenging optimisation problems [44, 45, 138]. Examples of practical applications of WSRP are home healthcare scheduling [2, 51], technician scheduling [56, 223] and security personnel scheduling [163]. Here we consider the problem of scheduling nurses and care workers to visit and provide care services to people in their homes.

Many researchers have tackled challenging WSRP. However, for the majority of works published in the literature, the datasets are not readily available and most of those appear to be artificially generated. For example, some benchmark scenarios have been adapted by extending vehicle routing problems with time windows (VRPTW). Castillo-Salazar et al. [42] modified the datasets proposed by Solomon [203] and Castro-Gutierrez et al. [46] in order to generate WSRP test instances. The original 56 Solomon instances are classified into six instance sets where only

routing components are defined. Each Solomon instance contains a list of locations for vehicles to deliver orders. For each location, the spatial coordinates, visit demand, time window and service time are given. In order to generate WSRP instances from Solomon's instances, Castillo-Salazar et al. [42] generated a set of workers of size roughly equal to a fifth of the number of visits. Worker-visit preferences and time-dependency between visits were also are randomly generated. However, those generated WSRP instances do not define skill requirements for the visits. The VRPTW instances proposed by Castro-Gutierrez et al. [46] are similar to Solomon's instances. Therefore, Castillo-Salazar et al. [42] also modified these instances in the same way in order to create another set of WSRP test instances. Rasmussen et al. [187] also tackled the home care crew scheduling problem where care workers have to be assigned to a number of visits at the patient's premises. They tested their algorithms by solving 60 instances generated based on two real-world scenarios. These instances have all the features of a WSRP as described later in this chapter. In these instances, the number of workers ranges from 6 to 15 and the number of visits ranges from 60 to 150. Note that to the best of our knowledge, the Rasmussen et al. [187] dataset is only available on request from the authors.

The related literature presents many ways to tackle WSRPs, including exact, heuristic and hybrid approaches. Angelis [6] modeled a HHC problem as a simple assignment problem, disconsidering the routing constraints and proposing a linear programming model to solve it and Borsani et al. [27] tackled a simplified version of the WSRP with mixed integer programming. Also, the problem has been tackled in the past with branch and price algorithms [17, 32]. Where exact methods failed to provide solutions, heuristic and metaheuristics have been applied to

solve large instances of WSRP. Such techniques include simulated annealing [23], local search [26], evolutionary approaches [2, 114], constraint programming techniques [1] and hyper-heuristics [162]. Hybrid approaches combining mixed integer programming with heuristics for either the routing or the scheduling sub-problems have been proposed [21] as well as those combining repeated matching with set partitioning model [73].

This chapter presents a set of real-world problem instances provided by our industrial partner and benchmark results obtained with heuristic and mathematical programming techniques. The intention is that other researchers can benefit from accessing these real-world WSRP instances to facilitate the development and comparison of solution methodologies. The chapter describes the datasets and the data model for the WSRP considered. A baseline constructive heuristic is also presented and its results compared to the results obtained with mathematical programming. The chapter also presents an analysis and discussion of the characteristics and difficulties of each set of instances, which reveals that each one presents unique challenges to an algorithmic solver. The datasets proposed here arise from daily planning scenarios in six distinct home healthcare companies of different sizes. A daily planning problem can contain up to 1,011 workers and 1,726 visits and involves scheduling visits and routing workers within a planning period of a single day.

3.1 The Home Healthcare Problem: Definition and Formulation

This thesis considers a real-world home healthcare problem, which is a type of WSRP [42]. Given a set of patients' requirements or visits and a

Table 3.1: Notation used in MIP model for WSRP

Sets	Definition
V	Set of all nodes denoted by $V = D \cup T \cup D'$. i, j instantiate nodes in V.
V^S	Set of source nodes and visiting nodes $V^S = D \cup T$.
V^N	Set of sink nodes and visiting nodes $V^N = D' \cup T$.
D	Set of source nodes.
D'	Set of sink nodes.
D_k	Set of source node for worker k, $D_k \subseteq D$
D'_k	Set of sink node for worker k, $D'_k \subseteq D'$
T	Set of visiting nodes.
E	Set of edges between two nodes.
K	Set of workers, k is a worker in K.
Parameters	
M	Large constant.
$\lambda_1, \ldots, \lambda_4$	Objective weights.
$t_{i,j} \in \mathbb{R}^+$	Travelling duration between node $i \in V^S$ and node $j \in V^N$.
$d_{i,j} \in \mathbb{R}^+$	Travelling distance between node $i \in V^S$ and node $j \in V^N$.
$\rho'_{k,j}, \rho''_{k,j}, \rho'''_{k,j} \in \mathbb{R}^+$	Costs of assigning worker k to node $j \in T$ relative to skills ($'$), staff($''$) and area($'''$) preferences.
$\delta_j \in \mathbb{R}^+$	Duration required at visit node $j \in T$.
$\alpha_L^k, \alpha_U^k \in \mathbb{R}^+$	Start and end of available time for worker k.
$w_j^L, w_j^U \in \mathbb{R}^+$	Lower (earlier) and upper (later) time windows for arriving at node j.
$q_j^k \in \{0, 1\}$	Suitability of worker k for visit node j, the value is 1 when the worker is suitable to work that visit, 0 otherwise.
$\gamma_j^k \in \{0, 1\}$	Region availability for worker on node j, the value is 1 when a worker is willing to work, 0 otherwise.
$p_j^k \in \mathbb{R}^+$	Incurred cost of assigning worker k to visiting node j.
$h^k \in \mathbb{R}^+$	The maximum working hours for worker k.
$r_j \in \mathbb{N}$	The number of workers required for visit node j.
Variables	
$x_{i,j}^k \in \{0, 1\}$	Decision variable, the value is 1 when a link between $i \in V^S$ and $j \in V^N$ is assigned to worker k, 0 otherwise.
$\omega_j^k \in \{0, 1\}$	Variable to indicate worker available time violation, the value is 1 when the assignment at node j is made outside available time of worker k, 0 otherwise.
$\psi_j^k \in \{0, 1\}$	Variable to indicate worker region availability, the value is 1 when the assignment at node j is made outside region availability of worker k, 0 otherwise.
$y_j \in \mathbb{N}$	Binary variable indicating that a visiting node j is unassigned.
$a_j^k \in \mathbb{R}^+$	Assigned time of worker k to work at node j.

set of care workers (also nurses and doctors), the goal is to find a work plan such that a number of suitable workers are assigned to perform all the patient visits. A worker is considered suitable if he/she has the skills and contractual conditions (working hours, coverage of locations, etc.)

needed for the tasks required in the visit. Also, any worker assigned to multiple visits must have enough time to complete all visits considering the travel time between visit locations. The length of the planning period in the real-world scenarios considered here is one day. The quality of a work plan is measured in terms of operational cost (overall distance travelled and staff costs), satisfaction of working conditions, satisfaction of clients and workers preferences and the number of unassigned visits. All the features and parameters for the healthcare planning problem investigated here come from real-world scenarios faced by Webroster's clients in the UK. We introduced this problem to the literature in previous work [139].

3.1.1 Parameters, Variables and Constraints

A mixed integer programming model (MIP) is described next for the problem outlined above. This model was developed by colleagues within our research group and presented by Laesanklang et al. [137]. Table 3.1 summarises the notations used in this model grouped in three parts: Sets, Parameters and Variables. The model is based on solutions found in the literature to map a real world problem into mathematical constraints. In this section we explain the model and its constraints. The references contained in this section refer to works that used similar formulations for the respective constraints.

Most of the models found in the literature formulate the HHC problem on a graph. We apply similar approach to the model, so that the model defines a directed graph $G = (V, E)$. The set of nodes V consists of three node types; source nodes D, sink nodes D' and visit nodes T. The source nodes D and sink nodes D' correspond to starting and ending locations, which in most cases in the scenarios presented here are

Minimise lexicographically

$$\sum_{j \in T} y_j, \tag{3.1}$$

$$\sum_{k \in K} \sum_{j \in T} (\omega_j^k + \psi_j^k), \tag{3.2}$$

$$\sum_{j \in T} (3r_j - \sum_{i \in V^S} \sum_{k \in K} (\rho'_{k,j} + \rho''_{k,j} + \rho'''_{k,j}) x_{i,j}^k), \tag{3.3}$$

$$\sum_{k \in K} \sum_{i \in V^S} \sum_{j \in V^N} (d_{i,j} + p_j^k) x_{i,j}^k \tag{3.4}$$

Subject to

$$\sum_{k \in K} \sum_{i \in V^S} x_{i,j}^k + y_j = r_j \qquad \forall j \in T \tag{3.5}$$

$$\sum_{i \in V^S} x_{i,j}^k = \sum_{n \in V^N} x_{j,n}^k \qquad \forall j \in T, \forall k \in K \tag{3.6}$$

$$\sum_{j \in V^N} x_{n,j}^k \geq \sum_{j \in V^N} x_{i,j}^k \qquad \forall k \in K, \forall i \in T, n \in D_k \tag{3.7}$$

$$\sum_{i \in V^S} x_{i,n}^k \geq \sum_{i \in V^S} x_{i,j}^k \qquad \forall k \in K, \forall j \in T, n \in D'_k \tag{3.8}$$

$$x_{i,j}^k \leq q_j^k \qquad \forall k \in K, \forall i \in V^S, \forall j \in T \tag{3.9}$$

$$a_j^k + M(1 - x_{i,j}^k) \geq a_i^k + x_{i,j}^k t_{i,j} + \delta_i \qquad \forall k \in K, \forall i \in V^S, \forall j \in V^N \tag{3.10}$$

$$w_j^L \leq a_j^k \leq w_j^U \qquad \forall j \in T, \forall k \in K \tag{3.11}$$

$$\sum_{i \in V^S} \sum_{j \in T} x_{i,j}^k \delta_j \leq h^k \qquad \forall k \in K \tag{3.12}$$

$$\sum_{i \in V^S} x_{i,j}^k - \psi_j^k \leq \gamma_j^k \qquad \forall k \in K, \forall j \in T \tag{3.13}$$

$$\alpha_L^k - a_j^k \leq M(1 - x_{i,j}^k + \omega_j^k) \qquad \forall k \in K, \forall i \in V^S, \forall j \in T \tag{3.14}$$

$$a_j^k + \delta_j - \alpha_U^k \leq M(1 - x_{i,j}^k + \omega_j^k) \qquad \forall k \in K, \forall i \in V^S, \forall j \in T \tag{3.15}$$

$$x_{i,j}^k, \gamma_j^k, \omega_j^k, \psi_j^k \text{ are binary} \qquad \forall i, j \in V, \forall k \in K \tag{3.16}$$

$$y_j \in \mathbb{N}, a_j^k \geq 0 \qquad \forall j \in T, \forall k \in K. \tag{3.17}$$

workers' homes. The starting and ending locations, denoted by $D_k \subseteq D$ and $D'_k \subseteq D'$, are defined for each worker k. The set of visit nodes T represents visits or patients' requirements. Note that several visits may share the same location. For simplicity of the model, $V^S = D \cup T$ and $V^N = D' \cup T$ are defined as sets of nodes containing outgoing edges and

incoming edges, respectively.

Each edge $e = (i, j)$ in the set E represents a directed travelling route between two nodes i and j. The target of the MIP model is to find a directed path for workers to travel from a starting node in D to a designated sink node in D', visiting patients in-between [58, 66, 149]. To build a path, several edges are assigned to a worker $k \in K$. An edge $e = (i, j)$ is assigned to worker $k \in K$ such that decision variable $x_{i,j}^k = 1$. The union of assigned paths should cover all the requirements in T. Since some visits may require multiple workers, the number of required workers at a node j is denoted by r_j. However, it is not always the case that all visit nodes can be visited, hence the total number of unassigned visits y_j must be kept to a minimum [33, 187]. These assignment rules are expressed by constraint (3.5). In addition, constraints (3.6) to (3.8) guarantee that a path of a worker k starts on a source node in D, connects visiting nodes in T and arrives to a sink node in D'. Each worker has a pre-defined source/sink node. Thus, a worker k must begin their route from a starting location in D_k and must return to the ending location in D'_k. Note that the problem instances presented here require workers to return to their start location when finishing the route, hence the start and end locations for a worker are the same even though the MIP formulation supports the case of different start and end locations for a worker.

Each worker has a set of skills and contractual conditions which indicate if the worker is eligible to perform that visit and the incurred costs. Therefore, a worker may only be assigned to visits for which the worker also has the required skills. A worker who is eligible for visit node j is defined by the binary variable $q_j^k = 1$ [67, 187]. These restrictions are expressed by constraint (3.9).

Each assignment must consider the travelling time to the patient's

location, the start time of the visit and its duration [33, 67, 187, 211], represented in constraint (3.10). Additionally, the arrival time must be in-between the provided time window [46, 52, 141, 203], as indicated by constraint (3.11). Note that in the problem instances presented here, exact arrival times are given which are expressed as $w_j^L = w_j^U$ for all $j \in T$.

Soft constraints are applied to this model to allow the violation of some non-mandatory requirements such as regions and time availability of workers. One question that might be raised is about the region and time availability being considered as soft constraints instead of hard constraints. In the real-world, it is not surprising that staff availabilities are not always up to date and hence such information might not always be accurate. Thus, a worker has a list of regions in which he/she is available to perform visits. Hence, assignments located in those geographical regions should be prioritised. In practice, assigning workers to visits outside their available regions is allowable but not desirable. A binary parameter $\gamma_j^k = 1$ specifies that a worker k is available in the region of a visiting node j. The binary variable $\psi_j = 1$ if the worker k assigned to node j is not available in that region, i.e. $\gamma_j^k = 0$. This case of region availability violation is formulated by constraint (3.13). Analogously, the same principle is applied to the worker's available time. If an assignment at node j has its start time outside the available time interval of the assigned worker, then the binary decision variable $\omega_j = 1$ (indicating a violation of the soft constraint). This is expressed by constraints (3.14) and (3.15). Time availability preference information is given for each worker [211]. The earliest and latest available times are denoted by α_L^k and α_U^k respectively. In order to avoid soft constraints violations, assignments should be made within the worker's available time. Con-

straint (3.14) states that the arrival time a_j^k to a visit node j should be later than the start of the available time. If the visit node j is the first on the worker's path, the worker may leave their starting location before the start of the available time to make the visit on time. In the same way, constraint (3.15) states that a worker k should leave a visit node j before the end of the available time α^U. The leaving time is $a_j^k + \delta_j$ as a_j^k is the arrival time of worker k to the visit node j and δ_j is the time required on visit j.

3.1.2 Objective Function

The objective function is presented in equations (3.1)–(3.4). Multiple criteria compose the objectives. Previous works found in the literature on related problems employ lexicographic approaches to define the objective function [42, 187]. Hence, the objective function is defined lexicographically. There are four tiers of objectives, sorted by their priorities. A higher priority tier has total precedence over a lower priority tier. The four objective terms are the number of unassigned visits, the number of soft constraint violations, preference violations, and the operational costs and distances.

The first and highest priority tier (Equation (3.1)) is composed of the number of unassigned visits (y). In some cases, a visit requires two workers which in this case counts as two visits. An unassigned visit is a visit that is not assigned to any worker. In general, unassigned visits must be kept to a minimum.

The second tier (Equation (3.2)) consists of the soft constraint violations. Thus, the aim is to minimise the number of two soft constraint violations: the worker's time availability (ω_j) and the worker's region availability (ψ_j). Hence, if an assignment at a time when the worker is

not available, it is counted as a worker's time availability violation and if an assignment is made outside a worker's available region, it is counted as a worker's region availability violation.

The third tier (Equation (3.3)) comprises the workforce-visit preferences. There are three types of preferences considered here: optional skills preferences (the client preferences), worker preferences (the service provider preferences), and working area preferences (worker preferences). Each preference has a value from zero to one where one indicates the most preferred option and zero indicates the least preferred option. For instance, to perform a visit, it may be required that the assigned worker has multiple skills. If all skills are mandatory (hard constraint), a worker that does not have all the required skills cannot be assigned to perform that visit. However, some of the required skills might be desirable only, i.e. a worker that has such skills is preferable instead of a worker that does not have them. The same principle applies to worker preferences and working area preferences. Then, the preference tier value is represented by $\rho' k, j$, $\rho'' k, j$ and $\rho''' k, j$; and is a summation of the three preferences. A higher preference value ρ_j^k corresponds to a preferable decision on assigning worker k to visit node j regarding skill preference, worker preference or working area preference respectively. Thus, the goal of maximising preferences is converted into a minimisation objective by penalising decisions that do not fulfil the highest preference. The highest preference value is given by $\sum_{j \in T} 3r_j$, because each assignment has three types of preferences and each preference can have a maximum value of one, r_j is the number of assignments at visit j. Therefore, from the second term of the objective function, the preferable solution is the one having the total preferences value nearer to the maximum preferences value.

The fourth and least important objective tier (Equation (3.4)) is the operational cost which consists of adding the workforce travelling distances ($d_{i,j}$) and incurred staff cost (p_j^k). The incurred staff cost arises from the fact that in all scenarios presented later here, companies give zero-hour contracts to workers hence paying hourly rates according to the assigned visits. The incurred staff cost p_j^k is zero if a worker k is employed on a full-time contract and the cost $p_j^k > 0$ when the worker is on a zero-hour contract. The cost p_j^k is pre-defined based on the individual contractual conditions which may not necessary reflect the worker's skills. Distances and costs are added to one tier because on a business perspective, distance typically means mileage cost paid to staff.

Implementation

The techniques presented in this thesis to solve the proposed HHC problem are compared with implementations of the model using the mathematical solver CPLEX or algorithms that attempt to use the solver on subproblems. The CPLEX solver, however, cannot deal with a lexicographic objective function, henceforth we employ a weighted sum approach with weight coefficients that emulate the lexicographic objective function. All experiments presented in this thesis use this implementation. Following is the revised objective function using λ coefficients in a weighted sum.

Minimise

$$\begin{aligned}
&\lambda_4 \sum_{j \in T} y_j + \\
&\lambda_3 \sum_{k \in K} \sum_{j \in T} (\omega_j^k + \psi_j^k) + \\
&\lambda_2 \sum_{j \in T} (3r_j - \sum_{i \in V^S} \sum_{k \in K} (\rho'_{k,j} + \rho''_{k,j} + \rho'''_{k,j}) x_{i,j}^k) + \\
&\lambda_1 \sum_{k \in K} \sum_{i \in V^S} \sum_{j \in V^N} (d_{i,j} + p_j^k) x_{i,j}^k
\end{aligned} \tag{3.18}$$

Weight Setups

Because the HHC problem combines features from both scheduling and routing problems, it is interesting to assess whether highlighting scheduling or routing components of the objective function causes distinct challenges for solving algorithms. Therefore, we introduce three weighting setups for the aforementioned λ coefficients. The **original** setup is the lexicographic objective function aforementioned. The **balanced** setup considers the two least important tiers as one, essentially increasing the weights of routing. The **routing** setup gives more importance to operational costs and distances than to preferences. In more detail, these weight setups are described next.

Objective Weight Setup: Original

This is the setup previously described and the one used throughout the thesis. The λ weigths emulate the hierarchical behaviour of the lexicographic objective function. The highest coefficient (λ_4) represents the highest priority tier in the lexicographic function and it is large enough as to guarantee that the tier has total precedence over lesser tiers. The same applies for the remaining coefficients and their respective priority

tiers (λ_3, λ_2 and λ_1). The λ coefficients were obtained empirically. Table 3.2 presents their values. Therefore, the objective priority values in this setup are set as $\lambda_4 \gg \lambda_3 \gg \lambda_2 \gg \lambda_1$.

Table 3.2: Values for the λ coefficients.

		λ_1	λ_2	λ_3	λ_4			λ_1	λ_2	λ_3	λ_4
	1	0.000315543017980	1.0	16.0	256.0		**1**	0.000009103162776	1.0	258.0	66564.0
	2	0.000888944200973	1.0	15.0	225.0		**2**	0.000010011638530	1.0	241.0	58081.0
	3	0.000600741314782	1.0	19.0	361.0		**3**	0.000007461110825	1.0	311.0	96721.0
A	**4**	0.001248486210470	1.0	14.0	196.0	**D**	**4**	0.000008495203056	1.0	279.0	77841.0
	5	0.002571685740003	1.0	6.0	36.0		**5**	0.000008262647759	1.0	284.0	80656.0
	6	0.001008745826314	1.0	14.0	196.0		**6**	0.000007158516910	1.0	327.0	106929.0
	7	0.002401190990731	1.0	6.0	36.0		**7**	0.000007158163341	1.0	328.0	107584.0
	1	0.000784098482769	1.0	18.0	324.0		**1**	0.000029242245687	1.0	220.0	48400.0
	2	0.002157031924072	1.0	6.0	36.0		**2**	0.000028967287243	1.0	220.0	48400.0
	3	0.000433681434965	1.0	35.0	1225.0		**3**	0.000026724748453	1.0	237.0	56169.0
B	**4**	0.001415989351760	1.0	15.0	225.0	**E**	**4**	0.000036650133791	1.0	181.0	32761.0
	5	0.000499972501512	1.0	31.0	961.0		**5**	0.000026522905446	1.0	242.0	58564.0
	6	0.000556127130662	1.0	28.0	784.0		**6**	0.000042447073805	1.0	160.0	25600.0
	7	0.000517965638160	1.0	30.0	900.0		**7**	0.000024719788834	1.0	260.0	67600.0
	1	0.000017827234488	1.0	88.0	7744.0		**1**	0.000001961791290	1.0	619.0	383161.0
	2	0.000342829716480	1.0	3.0	9.0		**2**	0.000001912973187	1.0	633.0	400689.0
	3	0.000023279304396	1.0	75.0	5625.0		**3**	0.000001533298028	1.0	782.0	611524.0
C	**4**	0.000148924319639	1.0	16.0	256.0	**F**	**4**	0.000001592549011	1.0	755.0	570025.0
	5	0.000185274048108	1.0	14.0	196.0		**5**	0.000001433640145	1.0	853.0	727609.0
	6	0.000020517092277	1.0	79.0	6241.0		**6**	0.000002270356092	1.0	845.0	714025.0
	7	0.000578730496782	1.0	3.0	9.0		**7**	0.000001277847426	1.0	923.0	851929.0

Objective Weight Setup: Balanced

This setup is solely used in this chapter. In this weight setup, operational cost and preferences are given the same importance but both still have lower absolute priority than unassigned visits and time and region availability violations. This setup seeks to establish a balance between the scheduling and the routing aspects of the problem. Hence, the objective priority values in this setup are set as $\lambda_4 \gg \lambda_3 \gg \lambda_2 \approx \lambda_1$.

Objective Weight Setup: Routing

This setup is solely used in this chapter. In this weight setup, preferences are not considered, hence the routing aspect of the problem is addressed in detriment of preferences satisfaction. This can be the case of a service

provider that does not have defined preferences in the problem instance definition. The objective priority values in this setup are set as $\lambda_4 \gg \lambda_3 \gg \lambda_1$.

HHC $\times$ VRP

Because the problem considered in this thesis considers a single day, it can be argued that it presents more similarities to a vehicle routing problem rather than to a scheduling problem. Typically, the inclusion of multiple days brings to light several working regulations that are not common to the VRP literature, such as the allocation of weekly breaks, maximum weekly working hours and minimum rest time between working shifts.

However, the HHC problem considered has two main distinctions from VRP.

- Generally, the importance of minimising distances in a VRP is high, while in the problem considered in this thesis it is low. This might look like something of minimal importance, but it completely changes the fitness landscape of the objective space to resemble one of a scheduling problem.

- Double-up visits (when visits require multiple workers) constitute a constraint not common in the VRP literature. In that context, it would mean multiple vehicles having to make deliveries at the exact same time and then each one going their own way afterwards. In the HHC problem considered here, this is common as many physical activities (such as lifting patients or bathing them) require at least two workers.

3.2 Six Real-World HHC Problem Datasets

The WSRP scenarios proposed here have been provided by Webroster, the industrial collaborator in this research. The data collected originates from six distinct home healthcare companies and, thus, each one presents a unique scenario of different size and features. These scenarios are called WSRP-A, WSRP-B, WSRP-C, WSRP-D, WSRP-E and WSRP-F. The scenario WSRP-A is on average the one with the smallest problem instances and WSRP-F is the one with the largest ones in terms of number of visits and workers. Each of the six scenarios has the data for seven problem instances, each one corresponding to a one day planning operation. Hence, in each dataset instance number 1 corresponds to a Sunday operation, instance number 2 corresponds to a Monday operation and so on. However, each problem instance was taken from a different month of the year selected at random, hence the sequence of instances in once scenario does not correspond to a particular one-week operation.

Table 3.3: Features present in the datasets.

	WSRP-A	WSRP-B	WSRP-C	WSRP-D	WSRP-E	WSRP-F
Skills Requirement (q_j^k)	✓	✓	✓	✓	✓	✓
Working Hours Restrictions(h^k)	✗	✗	✗	✓	✓	✓
Area Availability (γ_j^k)	✓	✓	✓	✓	✓	✓
Time Availability (α_L^k, α_U^k)	✓	✓	✓	✓	✓	✓
Preferred Skills ($\rho'_{k,j}$)	✗	✗	✓	✓	✓	✓
Preferred Workers ($\rho''_{k,j}$)	✓	✓	✓	✓	✓	✓
Preferred Areas ($\rho'''_{k,j}$)	✓	✓	✓	✓	✓	✓
Distances ($d_{i,j}$)	✓	✓	✗	✓	✓	✓
Costs (p_j^k)	✓	✓	✓	✓	✓	✓

Table 3.3 shows the features present in each dataset. Working hours restriction is present only in the largest sets, preferred skills are not present in the two smaller sets and notably set WSRP-C does not contain information about travelling distances. Table 3.4 also shows some basic information about the instances: total number of visits, total number of workers, number of areas, average duration of a visit in minutes and

Table 3.4: Basic information on the proposed WSRP scenarios.

	WSRP-A								WSRP-B							
	01	**02**	**03**	**04**	**05**	**06**	**07**	**mean**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**mean**
Number of Visits	31	31	38	28	13	28	13	26	36	12	69	30	61	57	61	46
Number of Workers	23	22	22	19	19	21	21	21	25	25	34	34	32	32	32	30
Number of Areas	6	4	5	4	4	8	4	5	6	5	7	5	8	8	7	7
Avg. Task Duration	144	110	102	70	57	157	104	106	160	155	112	58	106	101	103	113
Max. Simultaneous Visits †	10	11	12	7	4	10	4	8	13	4	17	5	14	17	14	12
Overall Compatibility	6.4	6.1	5.5	7.2	2.4	5.8	1.2	4.9	17.6	6.9	19.5	6.3	14.7	16.3	15.4	13.8

	WSRP-C								WSRP-D							
	01	**02**	**03**	**04**	**05**	**06**	**07**	**mean**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**mean**
Number of Visits	177	7	150	32	29	158	6	80	483	454	585	520	538	610	611	543
Number of Workers	1037	618	1077	979	821	816	349	813	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
Avg. Task Duration	440	599	456	356	344	469	484	449	62	50	67	61	58	57	56	52
Max. Simultaneous Visits †	160	6	139	24	22	150	4	72	80	78	101	85	91	101	95	91
Overall Compatibility	10.6	3.1	6.7	10.4	3.8	7.0	1.2	6.1	11.3	9.3	15.9	10.9	15.6	18.9	20.3	14.6

	WSRP-E								WSRP-F							
	01	**02**	**03**	**04**	**05**	**06**	**07**	**mean**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**mean**
Number of Visits	418	425	462	351	461	301	498	416	1211	1243	1479	1448	1599	1582	1726	1470
Number of Workers	243	244	267	266	278	278	302	268	805	769	898	789	889	783	1011	901
Number of Areas	13	14	15	13	15	13	16	14	45	46	54	47	59	44	64	51
Avg. Task Duration	99	102	101	73	100	64	98	91	79	87	83	89	72	81	92	83
Max. Simultaneous Visits †	97	102	116	61	109	54	123	94	223	245	267	233	265	224	332	257
Overall Compatibility	85.5	88.2	90.1	70.9	92.4	69.2	95.3	84.3	7.7	7.3	9.7	8.8	9.4	9.2	10.4	8.9

† maximum number of visits that are overlapping in time at any time of the day.

maximum number of simultaneous visits at any given time of the day. Finally, the last line presents the computation of the overall compatibility, which means the average number of skilled workers that are available both in time and area to perform a visit. A larger overall compatibility means that more skilled workers are fully available to perform the tasks, hence the solving algorithms have more candidates for each task, potentially increasing the computational difficulty to solve the problem. Also, the larger the overall compatibility, the less likely the solutions are to present time or area availability violations. On the contrary, a low compatibility value not only indicates that time and area availability violations are likely to occur, but also that fulfilling all visits is unlikely.

Dataset WSRP-A. This is the scenario with the smallest problem instances, all of which can be solved to optimality using a MIP solver. Hence this set is mainly used to compare the capability of heuristic techniques to reach the optimal solution. This scenario does not consider working hours restriction and skill preferences. Although the instances in this dataset have an average of 21 workers for 26 visits, there are only

4.9 fully compatible workers (skilled workers available both in the area and at the time of the visit) for each visit. Considering that the number of maximum average simultaneous visits is roughly eight, it is likely that the solution will require workers not fully available to perform some visits, hence this scenario can be considered to have a conflicting aspect regarding staff availabilities.

Dataset WSRP-B. In this set all instances can also be solved to optimality by a MIP solver. Like scenario WSRP-A, this set does not consider working hours restrictions and skill preferences. However, the overall compatibility is higher than the maximum number of simultaneous visits, which means that the likelihood of area or time availability violations is much smaller. Hence it should be easier to find feasible solutions without this type of violations. On the other hand, the difficulties of this scenario are the preferences matching and the routing aspects, as each visit has many fully compatible workers.

Dataset WSRP-C. This is a scenario with unique features. Like WSRP-A and WSRP-B, it does not considers working hours restrictions, however, it does considers skills preferences. Additionally, this is the only set where the travelling distances are not available. However, all regions in this scenario belong to the same city, hence the time travel between locations is estimated to be 20 minutes. Moreover, the problem instances in this scenario have the highest time spent on a visit, averaging almost eight hours. That means that the majority of the visits overlap in time, hence cannot be performed by the same worker, making the sequencing of workers of less importance. Finally, these are the instances where the number of workers is the highest compared to the other datasets, but

few workers are fully compatible for each visit. This large pool of workers exists in this scenario because most workers are hired on 'zero-hour' contracts. In such contracts, workers are paid only for the hours they work, hence the company keeps a large workforce available. Usually, these workers have other jobs and finding an available one can be tricky. When planning, the coordinator sends automated messages to the qualified workers and the first to reply gets the job.

Dataset WSRP-D. This is one of the large scenarios. In terms of the number of visits alone, scenario set WSRP-D is larger than WSRP-E. This dataset includes instances with short visits, averaging less than an hour and maximum simultaneous visits averaging 16% of total visits. Taking into account that the total number of workers is less than half the number of visits, it is safe to assume that the sequencing of visits plays an important role in these scenarios. Therefore, the routing aspect of these instances can be challenging. Additionally, the low number of fully compatible workers per visit makes it likely that the solutions will have a fairly high number of time or area availability violations.

Dataset WSRP-E. The instances in this set are among the most difficult ones. Although the average number of visits in these instances is smaller than in set WSRP-D, there is a higher number of workers and, specifically, fully compatible workers in these instances. Thus, for every visit, there is a large number of fully compatible workers to choose from. Also, the average task duration in these instances is approximately 90 minutes hence many tasks can be sequenced with a single worker. It can be safely concluded that in these instances the challenging aspect of the routing is attenuated.

Dataset WSRP-F. This set contains the largest instances with the average number of visits being roughly 1,500. Also, the number of regions in this scenario is large, meaning that it involves several smaller problems that share resources. It resembles a larger version of dataset WSRP-D as the average number of compatible workers is not too high. Hence, it can be expected that many time or region availability violations arise in the solutions to these problem instances.

Although all these datasets are scenarios of the same WSRP, they constitute distinct challenges to solution methods. As noted above, all instances in datasets WSRP-A and WSRP-B as well as the smaller instances in dataset WSRP-C can be solved to optimality with an MIP. All the other instances require alternative solving methods.

3.2.1 Description of Data Model

This subsection outlines the data files provided for the WSRP datasets described above. Each problem instance is named according to its set and identification number within the set, thus the first instance of set WSRP-A is named WSRP-A-01.zip. The zip file contains all the problem files. The file structure follows the model proposed in Chapter A. The data model uses a set of XML files, semantically divided, each one describing a part of the whole problem.

Within the files, all monetary costs are expressed in British Pounds (£), distances in miles and time values in minutes. Absolute times, such as the start time of a visit, are given as the number of minutes since the previous midnight. All preference levels are given as a value between 0.0 and 1.0, where 0.0 means complete indifference and 0.99 means strong preference. Values equal to 1.0 are used only when a feature is absolutely

required, such as skills. For instance, a skill with preference level 0.99 is strongly preferred (soft constraint), but a skill with preference level 1.0 is required (hard constraint).

The datasets and corresponding documentation are currently available online[1]. For a complete description of the data model and all the fields, please refer to the available documentation.

3.3 Experiments and Results

In order to evaluate the difficulty of the proposed problem instances, baseline solutions were generated using two algorithms: a baseline greedy constructive heuristic and the decomposition technique proposed by Laesanklang and Landa-Silva [136].

3.3.1 Greedy Constructive Heuristic

Algorithm 5 Greedy Constructive Heuristic

Ensure: Feasible solution s'
1: $s' \leftarrow$ new empty solution
2: **for** $i \in T$ (following a random order) **do**
3: $\quad best \leftarrow \varnothing$
4: $\quad$ **for** $k \in K$ **do**
5: $\quad\quad$ **if** $f_o(s' + (k,i)) < f_o(s' + best)$ **then**
6: $\quad\quad\quad best \leftarrow (k,i)$ //(k,i) is the assignment of worker k to visit i
7: $\quad\quad$ **end if**
8: $\quad$ **end for**
9: **end for**
10: **return** s'

Greedy heuristics are relatively simple to implement but are capable of producing competitive results on scheduling problems. For instance, Azab and Naderi [9] present three greedy heuristics for a job scheduling

[1]https://drive.google.com/open?id=0B2OtHr1VocuSNGVOT2VSYmp6a2M

problem where exact methods are impractical and Cai et al. [40] propose a fast greedy heuristic for real-time scheduling of trains on a single-track. On both examples the choice of greedy heuristics was made because the authors required a fast processing technique, however the empirical results presented showed that the chosen algorithms managed to achieve acceptable results.

Algorithm 5 presents the pseudocode for the baseline greedy constructive heuristic used on the preliminary studies of the datasets. This algorithm starts with an empty solution (line 1). Then, for each possible visit, following a random visit order (line 2), it chooses the best worker such that the costs of the partial solution are minimal (lines 4–8). This process is repeated until all visits are processed. It should be noted that due to the random visiting order, running the algorithm multiple times may result in different solutions.

3.3.2 MIP Solver and Decomposition Method

Optimal solutions were obtained with the MIP solver CPLEX on datasets A, B and instances 2, 4, 5 and 7 of dataset C. The decomposition technique proposed by Laesanklang and Landa-Silva [136] was used to find solutions for the remaining problem instances. That decomposition method works by splitting the problem into smaller sub-problems that the MIP solver can solve to optimality. In summary, the decomposition works as a three-step process: geographical decomposition, conflict repair and heuristic assignment. These steps are described below.

1. The method decomposes the problem into sub-problems according to visits' geographical areas. A sub-problem is solved to optimality. Note that the sub-problems' solutions may have conflicts between

them because a worker may be assigned to visits on different sub-problems that overlap in time.

2. Conflict repair tackles conflicting visits on different sub-problems by defining a variant of the main MIP aimed to solve conflicts. It considers the entire route of conflicting workers and solves them such that some of the conflicting tasks are left unassigned. The conflict repair is solved to optimality. The result from this step will be a feasible solution of the main WSRP with some unassigned visits as a result from the conflict repair process.

3. Heuristic assignment greedily selects the best available workers for visits that were left unassigned in the conflicts repair step. If an assignment in this step would result in a time conflict, then the visit is left unassigned in the final solution.

3.3.3 Scheduling and Routing Balance Evaluation

These experiments look at the balance between the scheduling and routing aspects of the problem and how changing their importance affects the performance of the algorithms. The impact of running the algorithms with different weight setups for the objective function on the same problem instance is investigated here. For the problem instances in dataset WSRP-A only, the algorithms were run using the **balanced**, **routing** and **original** weight setups described earlier in this chapter.

Table 3.5 presents the results including the optimal value obtained by the solver, the value from the decomposition method and the average value for 10 runs of the greedy constructive heuristic. Figure 3.1 presents relative gaps to the optimal solution for both the greedy constructive heuristic (dashed lines) and the decomposition method (solid lines) for

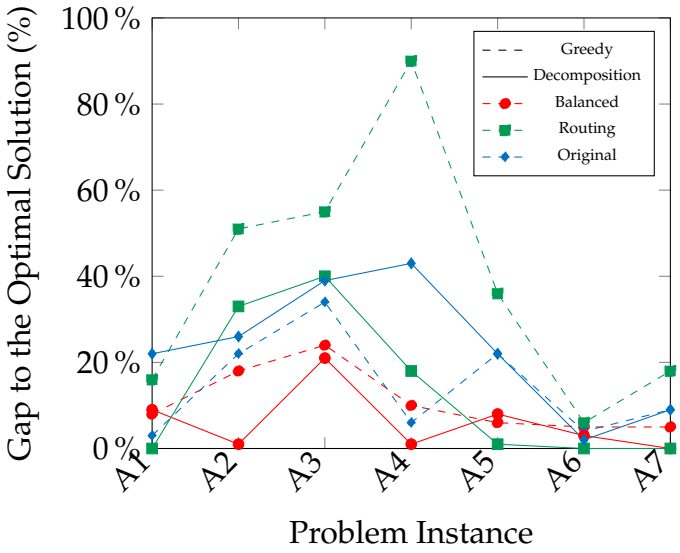


Figure 3.1: Relative gap to the optimal solution with different objective function setups.

the three objective function weight setups. It can be seen that for the **original** setup, the greedy constructive heuristic clearly presents better results with a smaller gap to the optimal solutions. For the **balanced** setup, the decomposition technique clearly surpasses the performance of the heuristic. For the **routing** setup, the decomposition method is far better than the greedy constructive heuristic.

Given the design of the greedy heuristic, results corroborate that it basically considers the problem as an assignment problem and restricts itself to matching the best worker to the visit in turn. In this heuristic, there is no procedure that attempts to improve the quality of the routing aspect in the solution. Hence it is no surprise that it exhibits good per-

Table 3.5: Comparing the decomposition technique against the greedy constructive heuristic for the different weight setups in the objective function.

		Balanced			Routing			Original		
		Optimal	Greedy	Decomp.	Optimal	Greedy	Decomp.	Optimal	Greedy	Decomp.
A	1	5.413	**5.907**	5.981	2.004	2.38	**2.006**	3.487	**3.586**	4.48
	2	4.697	5.703	**4.757**	0.98	2.009	**1.463**	2.491	**3.193**	3.362
	3	5.059	6.647	**6.43**	1.508	3.383	**2.508**	2.995	**4.564**	4.932
	4	3.397	3.757	**3.428**	0.014	0.142	**0.017**	1.421	**1.515**	2.493
	5	4.913	**5.213**	5.364	0.504	0.793	**0.507**	2.42	**3.095**	3.12
	6	5.704	6.029	**5.895**	2.014	2.14	**2.017**	3.549	3.694	**3.617**
	7	5.845	6.165	**5.845**	1.876	2.282	**1.879**	3.714	**4.064**	4.065

Table 3.6: Detailed information for the solutions found for each instance.

		A								B							
		1*	2*	3*	4*	5*	6*	7*	Avg.	1*	2*	3*	4*	5*	6*	7*	Avg.
	Total Visits	32	31	38	28	13	28	13	26	36	12	71	30	62	57	61	47
	Total Workers	23	22	22	19	19	21	21	21	25	25	34	34	32	32	32	31
	Assig. Workers	13	12	12	9	4	14	5	10	18	7	22	8	19	17	18	16
	Distances Sum	355.1	316.2	336.8	225.1	131.7	348.5	147.5	266	327.7	78.2	461.2	197.4	449.0	321.4	436.3	324
	Costs Sum	181.7	221.0	307.2	179.0	105.5	161.9	92.6	178	609.7	258.7	791.1	228.6	689.6	592.6	565.4	534
Pref.	Skills	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0
Pref.	Workers	22.6	20.3	25.6	19.7	4.2	21.1	4.3	17	17.4	4.7	44.6	16.4	31.5	31.7	29.9	25
Pref.	Area	26.0	26.0	32.0	25.0	10.0	20.0	11.0	21	30.0	11.0	47.0	12.0	42.0	47.0	44.0	33
Vio.	Area	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Vio.	Time	4	2	3	0	1	4	4	3	0	0	0	0	0	0	0	0
	Unassigned	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

		C								D							
		1	2*	3	4*	5*	6	7*	Avg.	1	2	3	4	5	6	7	Avg.
	Total Visits	177	7	150	32	29	158	6	80	516	482	623	558	568	655	657	580
	Total Workers	1037	618	1077	979	821	816	349	814	164	166	174	174	173	174	173	171
	Assig. Workers	164	6	143	27	25	150	6	74	94	78	118	91	103	114	114	102
	Distances Sum	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	3055.6	2884.6	3624.7	3402.6	2859.8	2599.7	2554.5	2997
	Costs Sum	44144.3	2375.5	34779.8	4567.0	3304.1	38767.7	1408.6	18478	6720.6	5446.7	8419.9	7257.0	7586.0	7969.8	8360.2	7394
Pref.	Skills	55.5	4.0	34.5	4.5	5.0	34.5	1.5	20	70.0	81.5	119.5	103.5	111.0	123.0	131.0	106
Pref.	Workers	56.0	2.5	40.3	6.4	5.7	40.3	1.5	22	258.3	211.2	319.2	229.1	323.2	394.4	409.6	306
Pref.	Area	79.0	6.0	54.0	17.0	13.0	78.0	5.0	36	515.0	481.0	620.0	541.0	563.0	653.0	656.0	576
Vio.	Area	96	1	94	15	16	75	1	43	0	0	0	0	0	1	0	0
Vio.	Time	45	3	38	3	5	44	4	20	89	108	57	74	42	29	31	61
	Unassigned	1	0	1	0	0	3	0	1	1	1	1	1	1	1	1	1

		E								F							
		1	2	3	4	5	6	7	Avg.	1	2	3	4	5	6	7	Avg.
	Total Visits	440	440	475	363	484	321	521	435	1238	1267	1564	1510	1706	1690	1847	1546
	Total Workers	243	244	267	266	278	278	302	268	805	769	898	789	889	783	1011	849
	Assig. Workers	140	142	152	91	146	74	162	130	250	281	308	265	301	286	368	294
	Distances Sum	1859.5	1953.1	2119.5	1437.9	2306.6	1238.8	2562.6	1925	12342.3	13636.5	14937.3	6255.2	8077.3	5972.5	16369.0	11084
	Costs Sum	3871.7	3848.2	4637.1	2692.7	4232.7	1723.3	4419.1	3632	20496.0	24365.3	24131.7	20837.1	18503.3	18612.8	31131.5	22583
Pref.	Skills	95.5	94.5	94.5	92.0	92.0	70.0	69.0	87	364.0	284.5	333.0	249.5	260.5	248.5	270.5	287
Pref.	Workers	253.3	236.1	262.8	150.5	246.8	142.5	320.9	230	545.4	604.5	908.0	786.8	899.5	861.3	1056.1	809
Pref.	Area	436.0	436.0	469.0	360.0	477.0	318.0	516.0	430	1089.0	1087.0	1392.0	1377.0	1583.0	1585.0	1654.0	1395
Vio.	Area	0	0	0	0	1	0	0	0	123	139	162	115	101	83	184	130
Vio.	Time	0	0	0	0	1	0	1	0	231	254	296	312	329	348	381	307
	Unassigned	0	0	0	0	0	0	0	0	7	6	1	3	0	2	7	4

* Optimal solution.

formance on scenarios where the minimisation of distances has low priority. The decomposition method achieved better solution quality when the routing aspect of the problem has higher priority, as seen on the **routing** weight setup of the objective function. Therefore, the difficulty to explore the search space varies as the balance between the priorities of the scheduling and routing aspects of the problem changes.

3.4 Detailed Solution Analysis

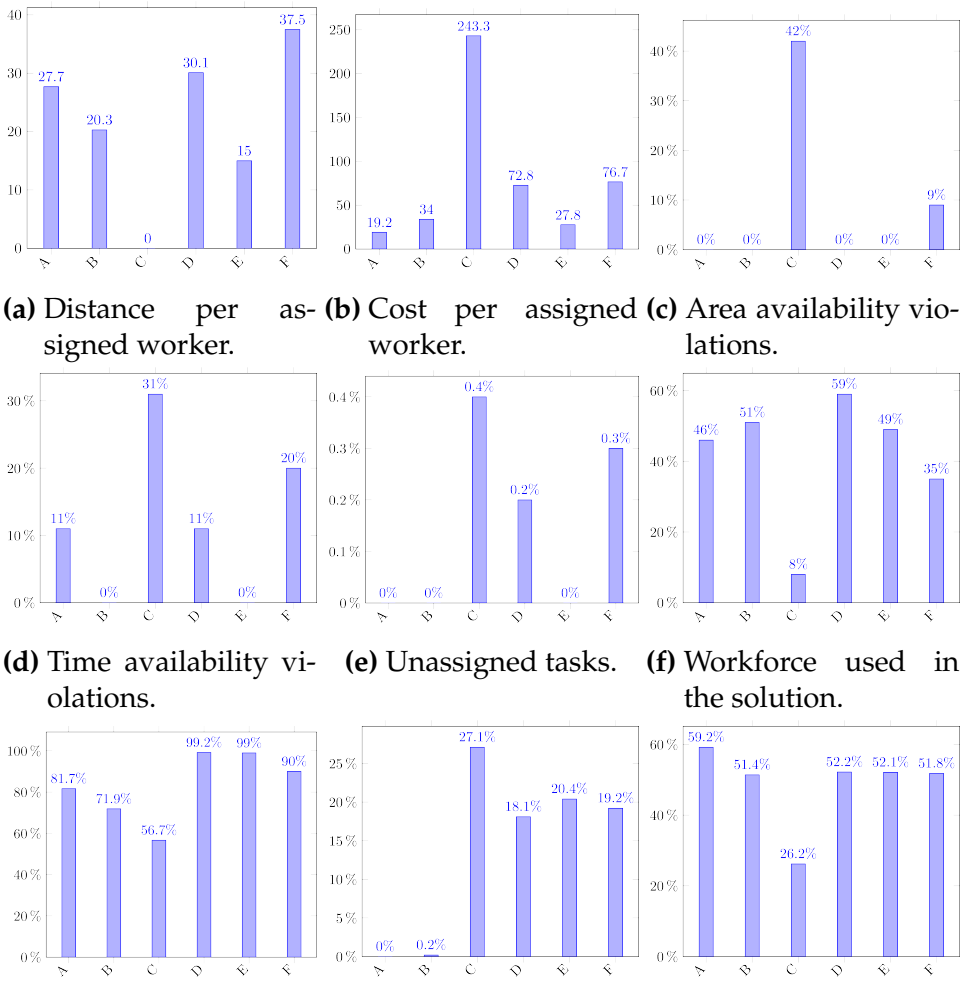
For the next experiments, only the **original** weight setup for the objective function was used because it is the one commonly used in the literature and also by the majority of our industrial partner customers. However, that might not be the case for other applications or for all the home healthcare industry, hence we introduced other weighting setups with different scheduling/routing balance. Nonetheless, the goal here

is to provide insights on the characteristics of the proposed problem instances, hence the solutions chosen are the best between those provided by the decomposition algorithm, 100 runs of the greedy constructive heuristic and the optimal solution (when available).

Table 3.6 presents the detailed results for the best solution found for each instance in each dataset. The table shows the total number of visits (considering the minimum number of workers required by each visit); total number of workers; number of workers assigned in the solution; the total sum of distances and costs; the sum of preferences coefficients for skills, workers and area; the sum of time and area violations; and the number of visits left unassigned. Figure 3.2 presents nine charts derived from the data in Table 3.6. This information is then used to draw some conclusions about each of the instance sets.

Dataset WSRP-A. Figure 8.9a shows that this set is among the top three on highest average travel distance per assigned worker, thus, on average each worker must travel 27.7 miles per day. In Section 3.2, it was stated that compatibility could be an issue in these instances. Area availability violations (3.2c) are not a problem in this scenario, however it is not possible to get rid of time availability violations (3.2d). One difficulty on this scenario lies in finding the minimum number of time availabilities while maximising the preferences (3.2g and 3.2i). Also, the high average distances hint that given a distances-biased objective function the problem can present a strong routing aspect (as we could see when comparing the results of different objective functions).

Dataset WSRP-B. This scenario was identified as a highly compatible one. It can now be seen that both time and region availabilities present



(a) Distance per assigned worker. **(b)** Cost per assigned worker. **(c)** Area availability violations.

(d) Time availability violations. **(e)** Unassigned tasks. **(f)** Workforce used in the solution.

(g) Areas preference satisfaction. **(h)** Skills preference satisfaction. **(i)** Preferred staff satisfaction.

Figure 3.2: Comparative results from the average solution of each set.

no violations (3.2c and 3.2d). The average travelling distance is the second lowest among the scenarios (not considering dataset C), hence the routing aspect is less important than in other scenarios. Operational costs also do not play a major role here (8.9b), it can be seen that the average cost per worker is relatively low. Finally, with a highly compatible staff, the difficulty of this scenario lies in finding the best workers that maximise the preferences satisfaction (3.2g and 3.2i).

Dataset WSRP-C. This is the scenario where the average operational costs are higher (8.9b). This means that minimisation of costs plays an

important part. Also, aside from having the largest worker pool (3.2f), this is the most conflicting scenario, it can be seen that both time and region availability conflicts present higher values among all sets (3.2c and 3.2d). Finally, because of the highly conflicting nature, it can be observed that the preference satisfaction for staff and areas is among the lowest in the datasets (3.2g and 3.2i). The difficulty of this set is to find a solution with the minimum number of conflicts and only then maximise the preferences. There is no routing aspect aside from timing the visits but the optimisation of costs can be detrimental.

Dataset WSRP-D. As observed in the previous section, the sequencing of visits plays an important role in these instances as the travelled distances (8.9a) are the second highest among all sets. Region availability is not a problem (3.2c), however a considerable number of time availability violations is present in the solution (3.2d). Hence, this appears to be a well-balanced set of instances where the routing, preferences attendance and the reduction in the number of time violations are equally important.

Dataset WSRP-E. As observed in the previous section, the high compatibility of this set can be verified by no time and region availability violations (3.2c and 3.2d). Routing overall plays a small role in this set since it presents the smallest overall travel distance (8.9a). Additionally, this is the only one of the larger sets in which all visits can be assigned (3.2e). The difficulty of this scenario basically lies in maximising the preferences (3.2g, 3.2h and 3.2i) as many distinct workers are fully available for each visit and many fulfil different preferences requirements.

Dataset WSRP-F. As expected, the instances in this set present a highly constrained scenario with the second highest region and time availability violations (3.2c and 3.2d). Routing can play an important role in this set as it has the highest average travelled distance per worker. The number of tasks that could not be assigned is also high (3.2e) providing further evidence of the conflicting nature of these problem instances. Finally, it is clear that preferences are very important in this scenario because the values obtained are among the highest (3.2g, 3.2h and 3.2i). This is a very balanced scenario regarding its features because every aspect of the problem looks important and meaningful: the routing is relevant with its high travel distances and sequencing importance, the preferences are there to be explored and the conflicting nature of availability is there as well. In addition, the very large size of these problem instances results is this being the most challenging instance set.

In the results obtained for many of the problem instances tackled here there are time and area availability violations (3.2c and 3.2d). In the instances of set C, if area and time availabilities were hard constraints, considering the best case where these violations happen simultaneously on the same visits (when possible), at least 31% of them would be left unassigned, which cannot happen in real planning operations, hence both area and time availabilities are tackled as soft constraints in practice.

From the experimental results presented here it can be concluded that each scenario represented by a dataset is unique in regard to the challenges it poses to a solving algorithm. There are scenarios with a strong conflicting nature where finding fully available workers is difficult (WSRP-A, WSRP-C, WSRP-D, WSRP-F). There are scenarios where routing plays a major role (WSRP-D, WSRP-F). There are scenarios where the high compatibility of workers move the focus of the search towards

finding the worker that maximises the satisfaction of the second tier of soft constraints (WSRP-B, WSRP-E).

Although all scenarios are instances of the same type of WSRP problem, with the same formulation and data model, they present distinct degrees of difficulty and challenge. The design of algorithms that can perform well on all the facets of the problem is an interesting research direction.

3.5 Conclusion

In this chapter a set of problem instances for the combined optimisation problem of scheduling and routing workforce is proposed. A detailed explanation of all constraints and objectives were provided, as well as a MIP formulation for the problem. Finally, three objective functions were introduced with different weightings, to enable us to assess the efficiency of solving algorithms when the routing and scheduling constraints of the problem have different weigthing. The main contributions of this work are: 1) to make these challenging WSRP problem instances available to the research community, 2) to provide benchmark solutions obtained with a baseline heuristic algorithm and a mixed-integer programming (MIP) solver, and 3) to provide a better understanding of this type of workforce scheduling and routing (WSRP) problem arising in home healthcare.

The analysis of experimental results showed that the heuristic approach performs better when the scheduling aspects are given priority. The MIP decomposition approach produces better results when the routing aspects of the problem are given priority. Hence, it might be difficult to design an algorithm that performs well on different weight setups of

the objective function. Then, exploring multi-objective approaches could be a way forward if solving HHC instances under different weight setups for the objective function is required.

Overall, all six home healthcare scenarios proposed here show different features and problem sizes, hence presenting different challenges to an algorithmic solver. These proposed datasets have the potential to be a suitable benchmark for testing and evaluating solution methods to tackle this type of WSRP.

Chapter 4

Lower Bounds for Workforce Scheduling and Routing Problems

In Chapter 3, a home healthcare benchmark dataset was presented alongside with results from exact and heuristic methods. We presented 42 instances grouped into six datasets, each representing a scenario from a distinct home healthcare provider. However, using the formulation presented in Chapter 3 and the IBM CPLEX solver, the optimal solution could be obtained on only 18 of the 42 instances. On the remaining instances, the solver run out of memory, even on a High Performance Computing environment with over 100GB of RAM available. This was because of the three-dimensional x_{ij}^k matrix. Therefore, solutions for the 24 largest instances are only empirically evaluated, where the optimal solution is not known. Hence, this chapter presents an algorithm to provide lower bound values for the HHC problem. The lower bounds represent an estimation of how much a feasible solution can be improved.

Knowing the lower bounds for the HHC problem is important. First,

because lower bounds represent good benchmark values when the optimal solution is unknown, hence they can provide information about the quality of solutions obtained by heuristic methods. Second, because lower bounds can help HHC companies to understand their operational capabilities, again, in the case where optimal solutions are not available. The lower bounds provide the decision maker with a guarantee of maximum visits coverage, if recruitment is needed in specific areas or if there is a shortage of workers on specific times of the day.

For the minimisation problem, a lower bound value can be obtained by several approaches. For a linear problem, a lower bound can be provided by finding a feasible solution of the dual problem. However, finding the tightest lower bound is to find the optimal solution for the duality which is also the optimal solution of the primal problem (by the duality theory). An integer (or mixed-integer) programming problem may have a high computational complexity such that the solver takes an impractical computational time to reach a tight bound. Therefore, the Lagrangian relaxation is an approach where constraint violations are allowed with violation costs, known as Lagrange multipliers. Thus, the complex constraints, which cause long computational time, can be relaxed and the lower bound can be obtained quicker.

The most popular approach to obtain a lower bound for the MIP problem is the Linear Programming (LP) relaxation, which consists of relaxing the integer constraints. The relaxed problem becomes a linear programming problem that takes significantly less computational time. The literature presents examples of LP relaxation to find a lower bound value for the vehicle routing problem (VRP) [207] and the capacitated vehicle routing problem [14]. Also, the lower bound can be improved by adding cutting planes which are mathematical constraints that re-

duce the feasible regions but do not eliminate feasible solutions [95]. The lower bound value improves iteratively when additional planes are added. LP relaxation is not only used to find a lower bound but also is a part of algorithms to find integer solutions such as branch-and-bound algorithms [186], branch-and-cut algorithms [170] and branch-and-price algorithms [18].

Constraint relaxation is not only limited to the integer constraints. The other constraints may be relaxed so that computing an optimal solution for the relaxed problem is possible. For example, a lower bound for the VRP can be obtained from the optimal solution of the multiple travelling salesmen problem [52]. Also, the lower bound value can be provided by heuristic algorithm. Augerat et al. [8] apply a capacity separation heuristic to find the best bound for the capacited vehicle routing problem. Three heuristic algorithms were employed to find lower bounds for the crew scheduling problem [160] and for the capacitated vehicle routing problem [14]. These three bounding heuristics consist of: 1. an extension of the bounding method proposed by Christofides et al. [52], 2. a column and cut generation procedure using Lagrangean relaxation, and 3. a column and cut generation procedure that attempts to close the duality gap of the near optimal dual solution. Lower bounds from the three procedures are summed to achieve a tighter lower bound value. For more detail, see also Baldacci et al. [14].

Lower bounds can also be computed without using mathematical models. Haouari and M'Hallah [101] tackled a hybrid flowshop problem to find an operation procedure of a flowshop with machine duplication. The lower bound was determined from a function involving the summation of completion time and the process time. Note that the objective function of this problem is to minimise the maximum completion time.

For this case, the lower bound value is not an outcome of the mathematical programming model. Instead, the lower bound is calculated from the job's minimum summation of completion times, idle times, and processing times. A similar estimation was also proposed by Lee and Vairaktarakis [145].

Albareda-Sambola et al. [3] calculated lower bounds for the stochastic location-routing problem using both mathematical models and derived equations by separating the problem's objective function into two parts. The first part is the cost associated with production plants and their fixed costs. This part of the problem was formulated as a mixed integer programming problem and the optimal solution to this problem is solved by a MIP solver. The second part tackled the routing problem where the lower bound was calculated from a priori route procedure.

4.1 Methodology

Early experiments showed that the datasets studied in this thesis present high computational challenge and few datasets can be solved to optimality by exact methods. The exact methods failed to provide solutions on the larger datasets because the MIP model requires too much memory from the computational resources, and even using the high performance computing facilities at the University of Nottingham was not enough to provide results. Hence, we can not rely on other exact techniques to obtain lower bounds using the current MIP model for the HHC datasets, such as column generation. Therefore, we propose a novel procedure to find lower bounds for the HHC problem.

Figure 4.1 illustrates the proposed method. We first relax the HHC problem into a relaxed scheduling subproblem RSS (illustrated on the

left-hand side of the branching tree) and a relaxed routing subproblem RRS (illustrated on the right-hand side of the branching tree). An optimal solution for the RSS is obtained by a procedure that greedily assigns workers to visits. Then, a second procedure calculates the number of unassigned visits in the optimal solution of the HHC problem and updates the optimal solution of the RSS problem by unnasigning that number of highest cost visits. The same procedure is also used to find the number of constraint violations for staff time and area availability. The number of unassigned visits, the number of time availability violations and the number of area availability violations are added together to provide the minimum assignment costs.

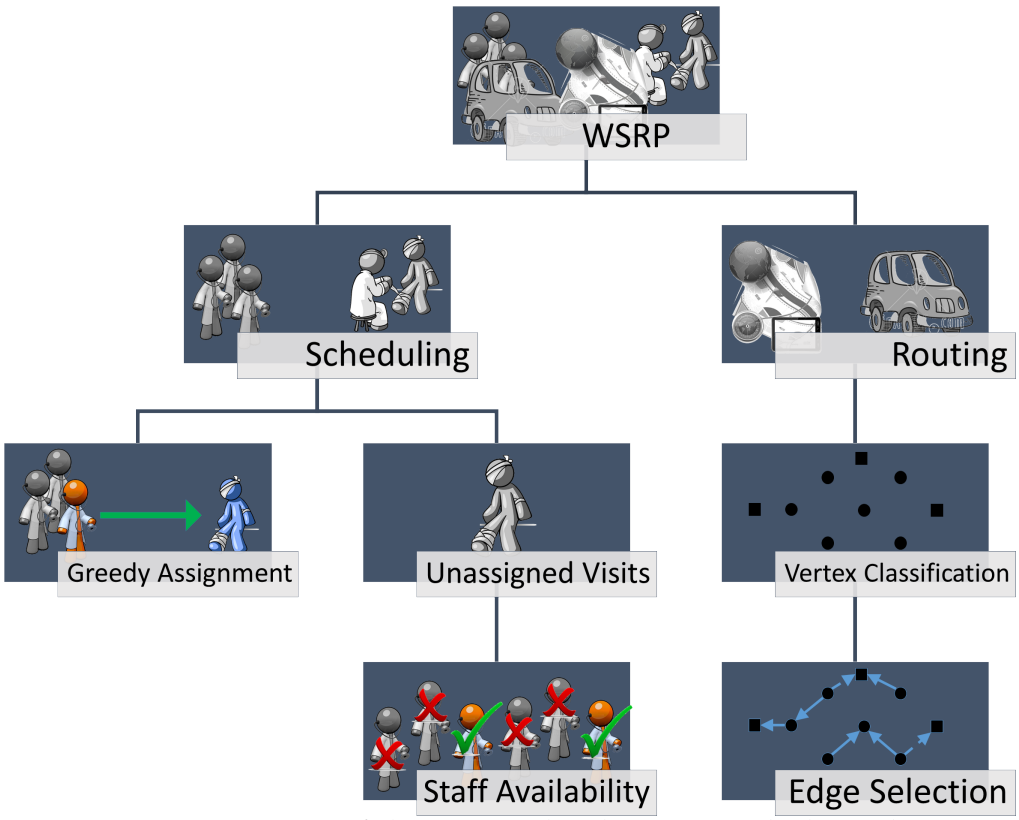


Figure 4.1: Overview of the proposed technique to calculate a lower bound for the WSRP.

The algorithm to find the lower bounds for the RRS starts by defining a graph such that visits are vertices and edges exist between two

vertices when both visits can be performed by the same worker. The algorithm builds a subgraph with all vertices and the edges with the lowest distances in the graph, following some building rules. The final subgraph provides the minimum travelling distances to be made for the whole plan. Finally, the minimum cost of assignments and the minimum travelling distances are added together to provide a lower bound to the HHC problem. The definition of each subproblem is formally provided next. Also, we prove that the combined lower bounds constitute a lower bound for the original problem.

The relaxed scheduling subproblem consists of the scheduling constraints only and the relaxed routing subproblem consists of the routing constraints only. Note that the constraint to minimise the number of visits left unassigned is kept on both subproblems. For simplicity, from now on we refer to the HHC problem $W = (V, K)$ such that $V = D \cup T \cup D'$ is the set of source vertices, visits and return vertices respectively and K is the set of workers.

Definition 6. *Given the HHC problem W, its **relaxed scheduling subproblem (RSS)** S can be obtained from W by removing time conflicts (constraint 7), maximum working hours (constraint 10) and updating the distance matrix such that $\forall i \in V^S, \forall j \in V^N, d_{i,j} = 0$.*

Definition 7. *Analogously, the **relaxed routing subproblem (RRS)** R of the HHC problem W can be obtained from W by relaxing skills, workers and region preferences ($\lambda_2 = 0$), time and area availabilities ($\lambda_3 = 0$), and removing all incurred staff costs ($\forall k \in K, \forall j \in T, p_j^k = 0$).*

Next, Definition 8 presents a solution for the HHC problem. Figure 4.2 illustrates the concept of the solution. Each column corresponds to a pair (visit, worker) representing the assignment of that worker to that

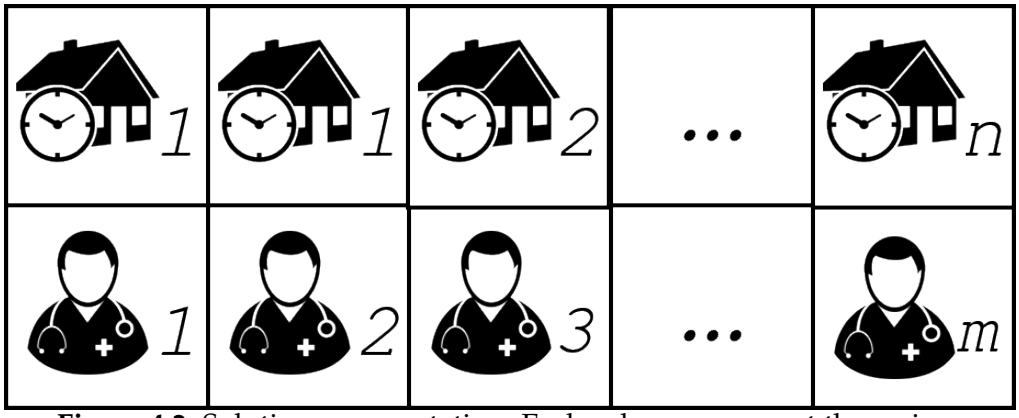


Figure 4.2: Solution representation. Each column represent the assignment of a visit to a worker. Note that visit 1 requires two workers.

visit. Without loss of generality, the solution representation can be extended to the RSS and the RRS.

Definition 8. *Let* $\vec{x} = [x_{i,j}^k]$ *be a decision vector for the MIP model of the HHC problem presented in Chapter 3 where* $x_{i,j}^k$ *is a decision variable representing the assignment of worker* k *to travel from node* i *to node* j*. We define an assignment solution set as* $X = \{(v_1, k_1), (v_2, k_2), \ldots, (v_n, k_m)\}$ *indicating that the visit* v_1 *will be served by the worker* k_1*, and so on; where* $v_1, v_2, \ldots, v_n \in T$ *and* $k_1, k_2, \ldots, k_m \in K$*. X is derived from the decision vector such that* $X = \{(v, k) | \exists x_{i,v}^k = 1, [x_{i,v}^k] = \vec{x}, i \in T\}$

We next show how to calculate the lower bounds for each separate subproblem, starting with the RSS and expanding it to include the number of unassigned visits and time and area availability violations; then presenting the lower bounds for the RRS; and finally joining the two techniques to present the lower bounds for the main HHC problem.

4.2 Lower Bounds for the Relaxed Scheduling Subproblem

The technique to calculate lower bounds for the RSS uses a greedy algorithm to provide the local best assignments. Because the RSS ignores time conflicts and working hours restrictions, the greedy algorithm is capable of obtaining the optimal solution for the problem. However, the optimal solution of the HHC problem may have a higher number of unassigned visits due to time conflicts. Thus, we propose an algorithm on a conflict graph to identify the unassigned visits. Later, we extend that concept to time and area availability violations.

In order to show that a greedy algorithm can provide an optimal solution for the RSS, we must demonstrate the greedy choice property of the problem [57]. To achieve that, Lemma 1 states that changing an assignment in a solution for the RSS does not cause the entire solution to become infeasible as long as the assignment itself is feasible (i.e. the visit is assigned to a fully skilled worker).

Lemma 1. *Let* $X_S = ((v_1, k_1), (v_2, k_2), \ldots, (v_n, k_n))$ *be a feasible solution for the RSS S. Let* $X'_S \subseteq X_S$ *and* $X''_S \subseteq X_S$ *be feasible partitions of* X_S *such that* $X'_S \cup X''_S = X_S$ *and* $X'_S \cap X''_S = \varnothing$*. Given any assignment* $(v_i, k_i) \in X'_S$*, and a new feasible assignment* (v_i, k_j) *such that* $k_i \neq k_j$*. Then,*

$$X'_S = (X'_S \setminus \{(v_i, k_i)\}) \cup \{(v_i, k_j)\}$$

will not cause X''_S *to become infeasible.*

When re-assigning a visit for the HHC problem, the solution may become infeasible if the new assignment causes a time conflict (constraint 7) or if the assignment causes the total working time to exceed

the maximum permitted (constraint 10). Since the RSS is derived from the HHC problem by removing constraints 7 and 10, then by Definition 6 no changes in X'_S will cause X''_S to become infeasible as long as the assignment (v_i, k_j) itself is feasible.

Next, Lemma 2 shows that if we split the set of visits of the RSS into two separate subsets and consider each one as a subproblem, the sum of objective-evaluation of the parts is equal to the objective-evaluation of the whole problem.

Lemma 2. *Given the RSS S and its objective function (note that the λ weights emulate the lexicographic objective function)*

$$f_S(X_S) = \sum_{k\in K}\sum_{i\in V^S}\sum_{j\in V^N}(\lambda_1 p_j^k + \lambda_2\rho_j^k)x_{i,j}^k + \sum_{j\in T}\sum_{k\in K}\lambda_3(\omega_j^k + \psi_j^k) + \sum_{j\in T}\lambda_4 y_j$$

where $X_S = \{(v,k)|\forall v \in V^N, \exists k \in K\}$ and $\rho_j^k = (\rho'_{k,j} + \rho''_{k,j} + \rho'''_{k,j})$.

The objective function $f_S(X_{S'} \cup X_{S''}) = f_S(X_{S'}) + f_S(X_{S''})$ when $X'_S \cup X''_S = X_S$, $X'_S \cap X''_S = \varnothing$.

Proof. From the mathematical model, we have $\omega_j^k, \psi_j^k, y_j = 0, \forall j \in D'$. Thus

$$\begin{aligned} f_S(X_S) &= \sum_{j\in V^N}\left(\sum_{k\in K}\sum_{i\in V^S}(\lambda_1 p_j^k + \lambda_2\rho_j^k)x_{i,j}^k + \sum_{k\in K}\lambda_3(\omega_j^k + \psi_j^k) + \lambda_4 y_j\right) \\ &= \sum_{j\in V^N} f_{S_j}(\mathbf{x_j}) \end{aligned}$$

where

$$f_{S_j}(\mathbf{x_j}) = \sum_{k\in K}\sum_{i\in V^S}(\lambda_1 p_j^k + \lambda_2\rho_j^k)x_{i,j}^k + \sum_{k\in K}\lambda_3(\omega_j^k + \psi_j^k) + \lambda_4 y_j.$$

And $\mathbf{x_j}$ is the assignment corresponding to visit j.

As $X'_S \cup X''_S = X_S$, $X'_S \cap X''_S = \varnothing$ and $X'_S = \{(v,k)|\forall v \in V', \exists k \in K\}$ and $X''_S = \{(v,k)|\forall v \in V'', \exists k \in K\}$ where $V' \cup V'' = V^N$ and $V' \cap V'' = \varnothing$. Therefore,

$$\begin{aligned} f_S(X_S) &= \sum_{j \in V'} f_{S_j}(\mathbf{x_j}) + \sum_{j \in V''} f_{S_j}(\mathbf{x_j}) \\ &= f_S(X_{S'}) + f_S(X_{S''}) \end{aligned}$$

□

Definition 9 (Greedy choice)**.** *Given the RSS S and a visit $v_i \in S$, an assignment (v_i, k_i) is obtained by a **greedy choice** if k_i presents the minimum costs when assigned to v_i.*

To show that the RSS can be solved by a greedy algorithm, Lemma 5 demonstrates that the RSS abide by the greedy choice property, i.e. an optimal solution is composed by greedy choices. However, it is intuitive to understand that the RSS possess this property given Lemma 2 because when we consider splitting the visits into a set with one visit only and another set with the remaining visits, given that both sets are optimal, the combined solution is optimal.

Lemma 3 (Greedy Choice Property)**.** *Let S be a RSS, $S' \subseteq S$ be a subproblem of S such that $S' \neq \varnothing$. Let $X^*_{S'}$ be an optimal solution of S' and let $a^* = (v_i, k_i)$ be an assignment in $X^*_{S'}$. Given an assignment $a' = (v_i, k_x)$ obtained by a greedy choice such that $v_i \in S'$. Then, $a' \in X^*_{S'}$ or a^* was obtained by a greedy choice.*

Proof. Let us consider.

1. If $k_i = k_x$, then $(v_i, k_x) \in X^*_{S'}$.

2. If $k_i \neq k_x$, still $f_{S'}(v_i, k_x) \leq f_{S'}(v_i, k_i)$ because of the greedy choice. Let us substitute (v_i, k_i) by (v_i, k_x) to obtain a new solution $X'_{S'} = (X^*_{S'} \setminus \{(v_i, k_i)\}) \cup \{(v_i, k_x)\}$. Note that by Lemma 1 $X'_{S'}$ is feasible. Now, by Lemma 2 we have that:

$$f_S(X^*_{S'}) = f_S(X^*_{S'} \setminus \{(v_i, k_i)\}) + f_S(v_i, k_i)$$

and

$$f_S(X'_{S'}) = f_S(X^*_{S'} \setminus \{(v_i, k_i)\}) + f_S(v_i, k_x)$$

hence since $f_S(v_i, k_x) \leq f_S(v_i, k_i)$ (Definition 9), then $f_S(X'_{S'}) \leq f_S(X^*_{S'})$. Therefore, because $X^*_{S'}$ is optimal, $f_S(v_i, k_x) < f_S(v_i, k_i)$ leads to a contradiction, hence $f_S(v_i, k_x) = f_S(v_i, k_i)$, meaning that $\{(v_i, k_i)\}$ was also obtained by a greedy choice.

□

Algorithm 6 RSSGreedySolver

Require: Partial solution X_S, Subproblem $S' = (T', K)$
Ensure: Optimal solution X^*_S

```
if T' = ∅ then
    return X_S
end if
v ← first visit in T'
k_x ← ∅
for k_i ∈ K do
    if X_S ∪ (v, k_i) is feasible and f_{S_v}(X_S ∪ (v, k_i)) < f_{S_v}(X_S ∪ (v, k_x)) then
        k_x ← k_i
    end if
end for
T' ← T' \ {v}
return RSSGreedySolver(X_S ∪ (v, k_x), S')
```

Algorithm 6 presents the pseudo-code for the procedure to solve the RSS by recursively performing the best choice of worker to be assigned

to the visit in turn until all visits have been assigned. Theorem 1 demonstrates the correctness of the proposed algorithm followed by the Corollary 1 that proves its optimality.

Theorem 1. *Let $S^t = (T^t, K)$ be a subset of the RSS $S = (T, K)$ obtained at any recursive call $0 \leq t \leq |T|$ during the RSSGreedySolver procedure such that T^t is composed of all visits presented in the partial solution X_S^t at time t. Then, X_S^t is optimal for S^t.*

Proof by induction.

- **Basis:** consider $X_S^0 = \varnothing$ before the first call of the algorithm and the subproblem $S^0 = (T^0, K)$ such that $T^0 = \varnothing$. Then the empty solution X_S^0 is optimal for the subproblem S^0 with no visits.

- **Induction:** Let X_S^t be an optimal solution for S^t. Line 5 selects a visit v and lines 6–11 picks the locally best worker such that the assignment (v, k_x) is feasible and the cost is minimum. Now let us consider the subproblem $S' = (T^t \cup \{v\}, K)$ and a solution $X_{S'} = X_S^t \cup \{(v, k_x)\}$. The greedy choice property (Lemma 5) assures us that $X_{S'}$ is optimal for the subproblem S'. Finally, line 12 updates T^t to T^{t+1} by inserting v into T^t and line 13 updates X_S^t to X_S^{t+1} such that:

$$S^{t+1} = S^t \cup \{v\}$$

$$X_S^{t+1} = X_S^t \cup \{(v, k_x)\}$$

  Therefore, because $S^{t+1} = S'$ and $X_S^{t+1} = X_{S'}$, X_S^{t+1} is an optimal solution for S^{t+1}.

- **Conclusion:** From the basis and the induction step, we conclude that X_S^t is optimal for S^t for any recursive call t. □

Corollary 1 (Optimality of RSSGreedySolver)**.** *Given the greedy algorithm presented in Algorithm 6 and a RSS S, then the solution*

$$X_S = \textbf{RSSGreedySolver}(\varnothing, S')$$

is optimal.

Proof. Line 3 returns the current solution for the current subproblem if $T' = \varnothing$. That only happens when there is no visits left to be assigned, hence $T^t = T$ and $S^t = (T, K)$. Since the current solution X_S^t is optimal for S^t (Lemma 1), then X_S^t is optimal for S. □

4.2.1 Unassigned Visits

We showed that the RSSGreedySolver algorithm is capable of providing an optimal solution for the RSS. However, because the RSS excludes time conflicts, in the case that their appearance would force visits to be left unassigned in the optimal solution of the HHC problem, in the optimal solution of the RSS these visits would still be assigned, and because unassigned visits (λ_4) represent the highest priority objective in the lexicographic objective function, the outcoming solution is not a good lower bound for the HHC problem. Hence, we now present an extension to the RSSGreedySolver so that it can identify the number of visits to be left unassigned and choose the visits to be left uncovered such that the resulting solution is not an optimal solution of the RSS anymore, but is a lower bound for the HHC problem.

Therefore, first, Lemma 4 shows that the number of unassigned visits in any optimal solution of the HHC problem is not smaller than the number of unassigned visits in any optimal solution of the RSS. Then, Lemma 5 shows that, given the optimal solution to the RSS, when removing the

assignment with the highest cost, if the total of unassigned visits on the resulting solution is equal to the optimal solution of the HHC problem, the resulting solution is a lower bound for the HHC problem. Later, Corollary 2 extends that idea for the removal of multiple assignments.

Lemma 4. *Let W be an HHC problem and S its RSS. Denote Y_X as the set of unassigned visits by any solution X. Let X_W^* be an optimal solution of W and X_S^* be an optimal solution of S, then $|Y_{X_W^*}| \geq |Y_{X_S^*}|$.*

Proof by contradiction. Suppose $|Y_{X_W^*}| < |Y_{X_S^*}|$. Then,

$$f_S(X_W^*) < f_S(X_S^*).$$

Therefore, $f_S(X_S^*)$ does not present minimum costs, thus contradicting our assumption that X_S^* is an optimal solution of S. □

Lemma 5. *Let W be a HHC problem and S its RSS. Let X_W^* be an optimal solution of W and X_S^* be an optimal solution of S. Denote Y_X a set of unassigned visits by the solution X. Given $X'_S = X_S^* \setminus \{(v,k)\}$, then,*

1. *If $|Y_{X'_S}| > |Y_{X_W^*}|$, then from Lemma 4 $|Y_{X_W^*}| = |Y_{X_S^*}|$.*
2. *If $|Y_{X'_S}| < |Y_{X_W^*}|$, then $f_S(X'_S) < f_S(X_W^*)$.*
3. *If $|Y_{X'_S}| = |Y_{X_W^*}|$ and $f_S(v,k) \geq \max_{(v_i,k_i)\in X'_S} f_S(v_i,k_i)$, then $f_S(X'_S) \leq f_S(X_W^*)$.*

Proof. Because the lexicographic objective function presents the highest priority for unassigned visits (λ_4), cases 1 and 2 are trivial.

Case 3: Let v_j be the visit that is unassigned in X_W^* and assigned in X_S^*. Let $S' = (S \setminus \{v_j\})$. It is trivial that $f_S(X_S^*) \leq f_S(X_W^*)$. By the greedy choice property of the RSS (Lemma 5), $f_{S'}(X_S^* \setminus \{(v_j,k_j)\}) \leq f_{S'}(X_W^*)$. Thus, $f_S(X_S^* \setminus \{(v_j,k_j)\}) \leq f_S(X_W^*)$.

Therefore,

1. If $v = v_j$, then $(X_S^* \setminus \{(v_j, k_j)\}) = X'_S$. Hence, $f_S(X'_S) \leq f_S(X_W^*)$.

2. If $v \neq v_j$ and $f_S(v,k) \geq f_S(v_j,k_j)$ (hypothesis), then $f_S(X_S^* \setminus \{(v,k)\}) \leq f_S(X_S^* \setminus \{(v_j,k_j)\}) \leq f_S(X_W^*)$.

□

Definition 10. *Given X is a solution to the HHC problem and V' is a subset of visits.*

$$X \oslash V' = X \setminus \{(v,k) | \forall v \in V', \forall k \in K\}.$$

Corollary 2. *Let W be a HHC problem and S its RSS. Let X_W^* be an optimal solution of W and X_S^* be an optimal solution of S. Let $Y_{X_W^*}$ be the set of unassigned visits by X_W^*. Let Y be the set containing all possible sets of $|Y_{X_W^*}|$ visits in S. Then,*

$$\min_{Y_S \in Y} f_S(X_S^* \oslash Y_S) \leq f_S(X_W^*).$$

Proof. This is an extension of Lemma 5. Let $S' = S \oslash Y_{X_W^*}$. Let Y_S^* be the set of visits such that $\min_{Y_S \in Y} f_S(X_S^* \oslash Y_S) = f_S(X_S^* \oslash Y_S^*)$. Knowing that $Y_{X_W^*} \in Y$,

1. If $Y_{X_W^*} = Y_S^*$, then the greedy choice property of the RSS (Lemma 9) ensures that $f_{S'}(X_S^*) \leq f_{S'}(X_W^*)$, hence $f_S(X_S^* \oslash Y_{X_W^*}) \leq f_S(X_W^*)$.

2. If $Y_{X_W^*} \neq Y_S^*$, then it means that we have a Y_S^* such that $f_S(X_S^* \oslash Y_S^*) \leq f_S(X_S^* \oslash Y_{X_W^*}) \leq f_S(X_W^*)$.

□

We showed that the removal of the highest cost assignments from the RSS optimal solution such that the resulting number of unassigned visits matches the optimal solution of the HHC problem can potentially outcome a better lower bound for the HHC problem. Hence, with the

knowledge of the number of unassigned visits existing in the optimal solution of the HHC problem, we can derive an algorithm to take the produced RSS optimal solution and remove the highest cost assignments.

Therefore, next we propose a technique to approximate the number of unassigned visits in the optimal solution of the HHC problem. This technique first creates a graph containing the time conflicts information such that the vertices represent the visits and the edges indicates that two visits cannot be performed by the same worker without causing a time conflict. Then, it identifies cliques in that graph and counts how many workers are fully available to perform all visits in each clique. If the number of available workers is smaller than the number of visits in the clique, we define a *conflicting clique*. Finally, we remove the minimum number of maximum cost vertices in the graph such that no conflicting cliques remain in the graph.

Definition 11. *Given the HHC problem W, we define a* ***conflict graph*** $\Gamma_W = (T, E)$ *of W such that T is the set of visiting nodes of W and $E = \{(v_i, v_j) | v_i, v_j \in T$ and the assignment of v_i and v_j to a same worker generates a time conflict$\}$.*

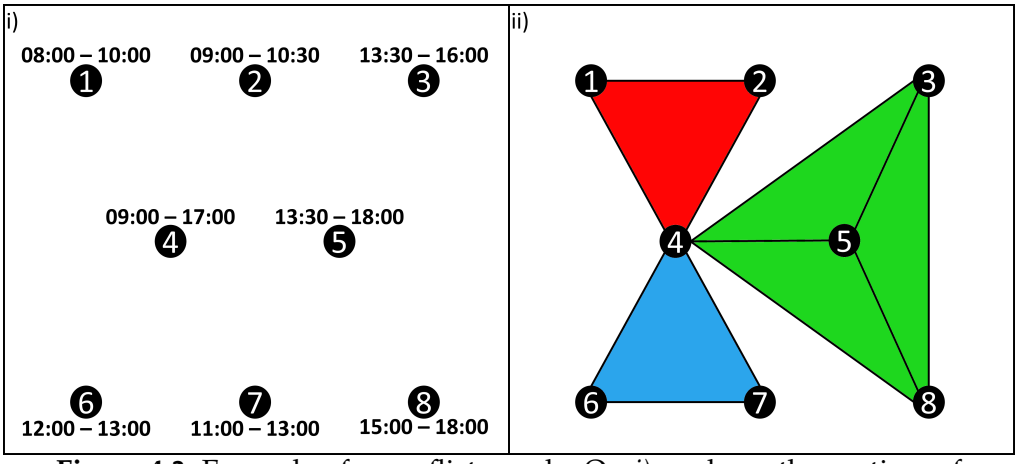


Figure 4.3: Example of a conflict graph. On *i)* we have the vertices of the graph, representing the visits and the times in which they occur. On *ii)* we have the full conflict graph with the maximal cliques expressed in different colours.

Figure 4.3 present on *i)* eight vertices corresponding to the eight visits of the HHC problem. On top of each vertex are the start and end times of the respective visit. On *ii)* we have the corresponding conflict graph. Each two vertices are connected by an edge only if no single worker can perform both visits without causing a time conflict. Additionally, the three coloured regions represent the maximal cliques in that graph.

Next, Lemma 6 states that given a clique, if the number of available workers for the visits of that clique is smaller than the number of visits, then the difference will be the number of unassigned visits between those visits in the optimal solution of the HHC problem.

Lemma 6. *Let W be the HHC problem and X^*_W be an optimal solution of W. Let Γ_W be the conflict graph of W. Given a clique C_k of Γ_W, and the number σ of unique available workers of all visits of C_k. If $\sigma < |C_k|$, then $|C_k| - \sigma$ visits in C_k are unassigned in X^*_W.*

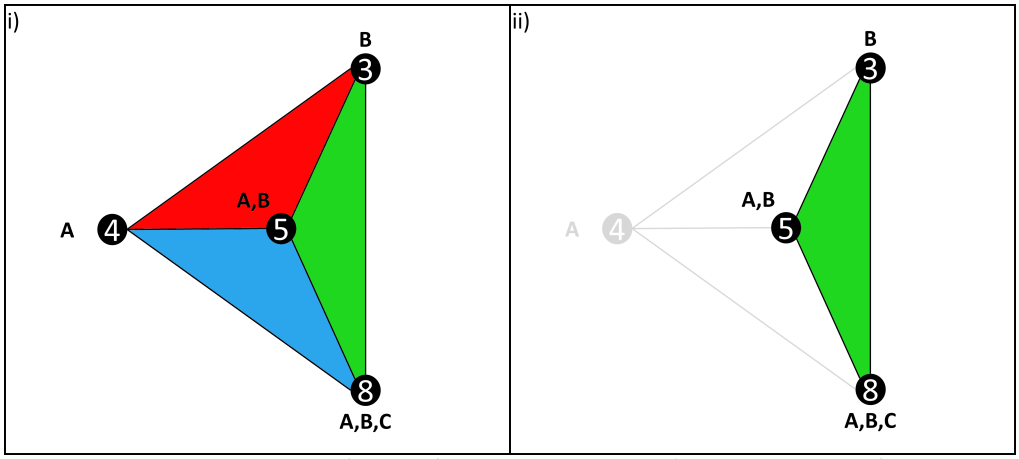


Figure 4.4: Example of a conflicting clique. In *i)*a maximal conflicting clique consisting of four vertices is presented. Near each vertex is the list of available workers (namely A, B and C). On *ii)* we have the resulting graph after removing the vertex 4, hence eliminating all conflicts.

Example: Figure 4.4 focuses on the green maximal clique of the previous figure. In *i)* we also have the information of the available workers for each visit, hence vertex ❹ has only worker A available, vertex ❺

has workers A and B, and so on. Note that we have four vertices but only three available workers in total, namely A, B and C. Thus, according to Lemma 6, necessarily one visit is left unassigned. In *ii)*, visit ❹ was unassgined. Moreover, note that in *i)*, the red region constitutes a smaller conflicting clique since we have three vertices and only two available workers, hence by removing vertex ❹ we eliminated two conflicting cliques.

Next, Lemma 7 links the number of removed vertices from the conflict graphs with the number of visits unassigned in the optimal solution of the HHC problem. Then, Corollary 3 includes the number of visits already unassigned in the optimal solution of the RSS. Finally, Theorem 2 tells us which vertices must be removed from the solution such that the resulting solution will be a lower bound for the HHC problem.

Lemma 7. *Let W be an HHC problem, Γ_W be the conflict graph of W and X^*_W be an optimal solution of W. Let C be the set of all cliques of Γ_W. Let $Y_k \in Y$ be a set of unique vertices removed from the cliques in C such that $|Y_k|$ is minimum and $|C_k \setminus Y_k| \leq \sigma, \forall C_k \in C$. Then,*

$$|Y_k| \leq |Y_{X^*_W}|.$$

Proof. by contradiction.

Assume $|Y| > |Y_{X^*_W}|$. Let Y_W be the set of unassigned vertices in X^*_W. Since X^*_W is an optimal solution of W, $|Y_{X^*_W}|$ vertices could be removed from C such that $\forall C_k \in C, |C_k \setminus Y_{X^*_W}| \leq \sigma$. Therefore $|Y_k|$ is not minimum, hence a contradiction. □

Corollary 3. *Let W be an HHC problem, Γ_W be the conflict graph of W and X^*_W be an optimal solution of W. Let S be the RSS of W and X^*_S be an optimal solution of S. Let Y_S be the set of unassigned vertices in X^*_S. Let C be the set of*

all cliques of Γ_W. Let Y_k be a set of unique vertices removed from the cliques in C such that $|Y_k|$ is minimum and $\forall C_k \in C, |C_k \setminus Y_k| \leq \sigma$. Then,

$$|Y_k \cup Y_S| \leq |Y_{X^*_W}|.$$

Proof. Algorithm 6 only leaves a visit unassigned when there are no feasible workers to perform that visit (line 8). Without loss of generality, let v be an unassigned visit in X^*_S. Then, v represents a clique C_v (with a single vertex) in Γ_W such that its value $\sigma_v = 0$, thus $v \in Y_k$. Therefore $Y_S \subset Y_k$, thus by Lemma 7 $|Y_k \cup Y_S| = |Y_k| \leq |Y_{X^*_W}|$. □

Theorem 2. *Let W be a HHC problem, Γ_W be the conflict graph of W and X^*_W be an optimal solution of W. Let S be a RSS of W and X^*_S be an optimal solution of S. Let Y_S be the set of unassigned vertices in X^*_S. Let C be the set of all cliques of Γ_W. Let Y be the set of k sets Y_k of unique vertices removed from the cliques in C such that $|Y_k|$ is minimum and $\forall C_k \in C, |C_k \setminus Y_k| \leq \sigma$. Then,*

$$\min_{Y_k \in Y} f_S(X^*_S \setminus Y_k) \leq f_S(X^*_W)$$

Proof. Consider the cases:

1. If $|Y_k| < |Y_{X^*_W}|$, then it is trivial (Lemma 5 and Corollary 3).
2. If $|Y_k| = |Y_{X^*_W}|$, then it is trivial (Corollary 2 and Corollary 3).

□

The Proposed Algorithm:

We now present the proposed algorithm that takes the optimal solution of the RSS and identifies which visits are left unassigned in order to improve the lower bounds for the corresponding HHC problem. The algorithm is composed of three sequential steps:

Step 1 – Build Conflict Graph: initially the algorithm takes the HHC problem and builds a conflict graph (Definition 11) as presented in Figure 4.3. Visits are vertices and every two vertices are linked by an edge if having their corresponding visits assigned to a same worker would cause a time conflict. Also, the algorithm lists, for each vertex, the workers that can perform the corresponding visit.

Step 2 – Cliques Identification: next, the algorithm identifies and lists cliques in the conflict graph. Identifying all cliques in a graph is an NP-Hard problem [120]. Thus, we limit ourselves to cliques of a maximum given size `C_SIZE` using a simple backtracking algorithm.

Step 3 – Identify Maximum Cost Vertices: the algorithm uses the list of cliques generated in the previous step to feed a mixed integer programming model to identify the the minimum number of vertices with maximal costs such that the removal eliminates the conflicting cliques in the list (Theorem 2).

Set	**Description**
T	a set of visiting nodes
C	a set of conflicting cliques
Parameter	
$\zeta_{i,c}$	a parameter indicates node i is a part of clique c.
η_i	a minimum cost of assigning visit i.
ξ_c	a parameter indicates maximum number of nodes for clique c.
Variable	
x_i	a binary variable indicates node i is selected for assignment.

$$\text{Maximize} \quad \sum_{i=1}^{|T|} \frac{\eta_i + 1}{\eta_i} x_i \tag{4.1}$$

Subject to

$$\sum_{i=1}^{|T|} \zeta_{i,c} x_i \leq \xi_c \qquad \forall c \in C \tag{4.2}$$

$$x_i \in 0, 1 \qquad \forall i = 1, \ldots, |T| \tag{4.3}$$

We consider the problem as a maximum set packing. The set of vertices T and the set of conflicting cliques C are given. A conflicting clique $c \in C$ is a subset of T. We introduce a binary parameter $\zeta_{i,c} = 1$ to present that vertex i is a member of clique c, and $\zeta_{i,c} = 0$ otherwise. The size of the clique indicates the number of required available workers. That is, if a number of available worker ξ_c for the clique c is more than the number of nodes in the clique, all visits can potentially be assigned. However, it might be the case that the number of available workers is less than the clique size. Therefore, the MIP model, presented in equations (4.1) - (4.3), is subject to find the largest number of vertices to which conflicting cliques does not have more vertices than the number of available workers. A decision variable $x_i = 1$ decides that the vertex i should be selected for inclusion, $x_i = 0$ otherwise. If there are multiple solutions with the same number of selected vertices, the model chooses the cheapest assigned vertices where the cost of vertex i is presented by η_i.

Step 4 – Remove Assignments: finally, the algorithm takes the optimal solution of the RSS and removes the assignments for the visits identified in the previous step. The new solution X' potentially provides a tighter lower bound for the HHC problem.

Note that limiting the clique size in step 2 does not invalidate the

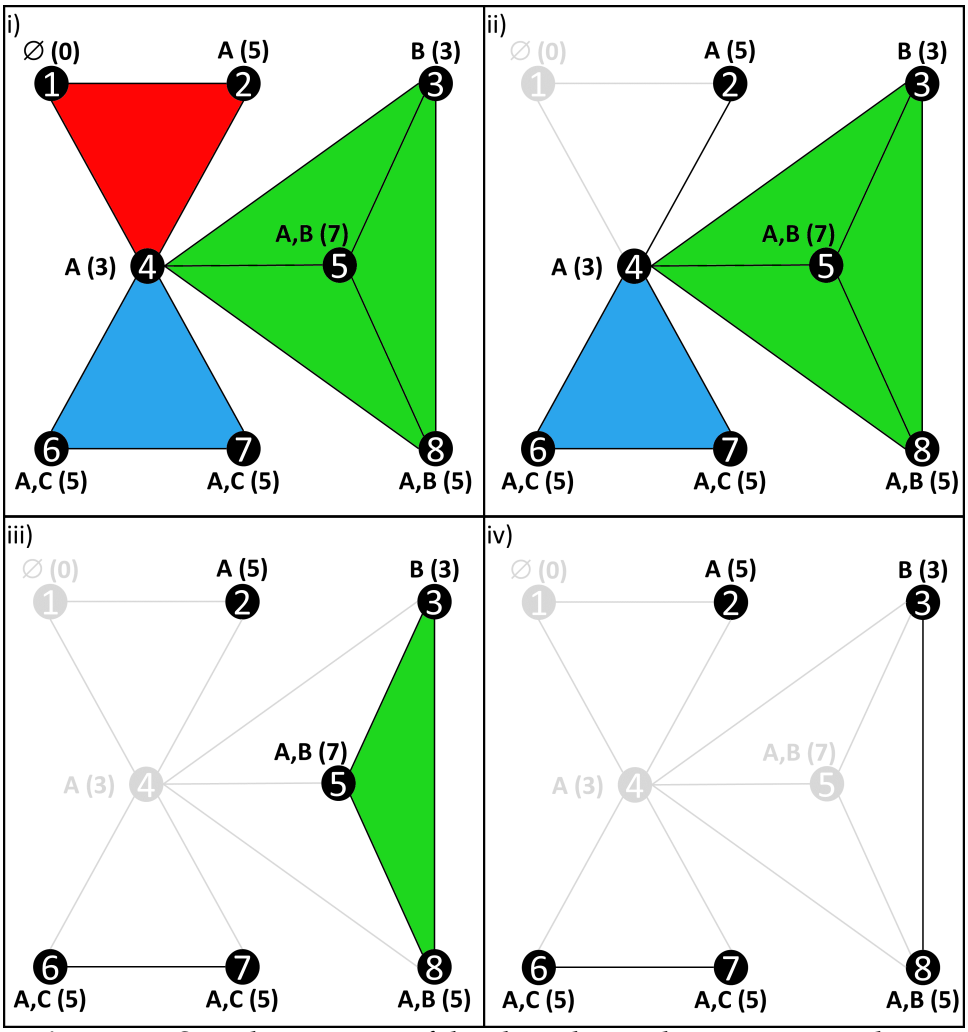


Figure 4.5: Sample execution of the algorithm to detect unassigned visits. In *i)* the problem is presented with the three maximal cliques presented in different colours. Each subsequent part of the figure presents the removal of a vertex according to the algorithm criteria.

proposed algorithm. In the worst case, the algorithm is not capable of identifying the cliques larger than `C_SIZE`, hence the number of vertices that is later removed from the graph to eliminate conflicting cliques is potentially smaller than the optimal number, but never larger, therefore the final solution constitute a valid lower bound for the HHC problem.

Example: Figure 4.5 presents a conceptual example of how the algorithm works. In *i)* we have the conflict graph corresponding to the visits presented in Figure 4.4. The coloured regions represent the maximal

conflicting cliques. The number of workers available for each visit can be found next to each corresponding vertex (A, B and C). The number corresponding to the minimum assignment cost is inside parentheses. Initially, we see that the red clique is made of three vertices and it has only one worker available, the blue clique has three vertices and two workers available, and the green clique has four vertices and two workers available. Also, we see that vertex ❶ does not have any available worker. Because that incurs a clique of size one with zero workers available, that vertex must necessarily be removed. The resulting graph is displayed in *ii)*. Now the red clique has two vertices and one worker available, still a conflicting case. Because vertex ❹ is common to all three cliques, its removal can eliminate the red and blue conflicting cliques while improving the green clique. In *iii)* we see the resulting graph after removing that vertex. The green clique now has three vertices and two workers. Removing any vertex can eliminate the conflicting situation. We know that in the optimal solution of the HHC problem, one of these three vertices is unassigned. Nonetheless, to be able to ensure the lower bound we must remove the highest cost vertex, which in this case is ❺. In *iv)* we see the final graph with no conflicts. The final lower bound can be expressed by:

$$f_S(X) = 5 + 5 + 5 + 5 + 3 + (3\lambda_4)$$

4.2.2 Time and Area Availabilities

We extend the concept of using the conflict graph to time and area availabilities. We use the same algorithm that estimates unassigned visits to identify which visits have time and area availabilities violations. The η_c of each vertex in the conflict graph is not the number of skilled workers, but the number of skilled **and** time/area available workers. It is easy to

see that the visits removed from C necessarily have a time or area availability violation. In this case, before running the RSS greedy algorithm, we consider that for those visits, no available worker can be assigned, hence instead of removing the assignment, the algorithm selects the minimum cost worker that violates time or area availability. Again, this is only possible because we exploit the lexicographic objective function.

4.3 Lower Bounds for the Relaxed Routing Subproblem

We now introduce the algorithm to calculate the lower bounds for the RRS. The concept is to build a graph where vertices represent visits and staff start/end locations. Edges are possible routes between two places, where the cost of an edge relates to the distance between the geographical places that the incident vertices represent.

The algorithm uses the graph to classify the vertices and select edges according to a set of rules. The lower bound is obtained because the edges selected individually possess minimum costs and the number of edges selected is $|T| + 1$, hence we are considering the worst case when a single worker can perform all visits sequentially. The key points are:

- the resulting selection of vertices and edges is not a solution for the RSS nor the HHC problem;
- the sum of the costs of the edges of the resulting selection is guaranteed to be smaller than the sum of distances in the optimal solution of the HHC problem.

Following we define the minimum directed graph employed by the algorithm.

Definition 12. *Given the RRS R, we define the* minimum directed subgraph $G_R = (V, E', \omega)$ *such that E is the set containing all two shortest (considering distances only) directed edges $e_{u,v}$ leaving from each vertex in V and*

$$\omega : E \to \mathbb{R}, \omega(e_{u,v}) = d_{u,v}$$

, where $u, v \in V, d_{u,v} \in R$.

Definition 13. *Given the RRS R. Let X_R be a solution of R. We can represent X_R as a* ***directed graph*** *$G_X = (V, E', \omega)$ such that*

$$E' = \{e_{u,v} = (u, v) | x_{u,v} = 1, x_{u,v} \in X_R\},$$

$$\omega : E' \to \mathbb{R}, \omega(e_{u,v}) = d_{u,v}$$

where $u, v \in V, d_{u,v} \in R$.

Definition 14. *Given the RRS R. Let X_R be a solution of R and $G_X = (V, E', \omega)$ its graph representation. Let $E_v = \{(v, u) | (v, u) \in E', \forall u \in V\}$ be the set of outgoing edges of v. Then,*

$$f_R(X_R) = f_R(G_X) = \sum_{e \in E'} \omega(e) = \sum_{v \in V} \sum_{e \in E_v} \omega(e).$$

The proposed algorithm distinguishes the vertices of a minimum directed subgraph G_R as of two types: ● vertices representing the visit locations and ■ vertices representing workers' start/end locations. On the following illustrations, solid lines represent edges with minimum cost and dashed lines represent the edge with the next higher cost. The algorithm derives a subgraph G'_R of G_R such that all vertices are kept in G'_R and no more than $|T| + 1$ edges are present in G'_R. The algorithm selects the edges to be included by classifying the vertices individually.

Consider the following vertex classification for ● vertices, as displayed on Figure 4.6:

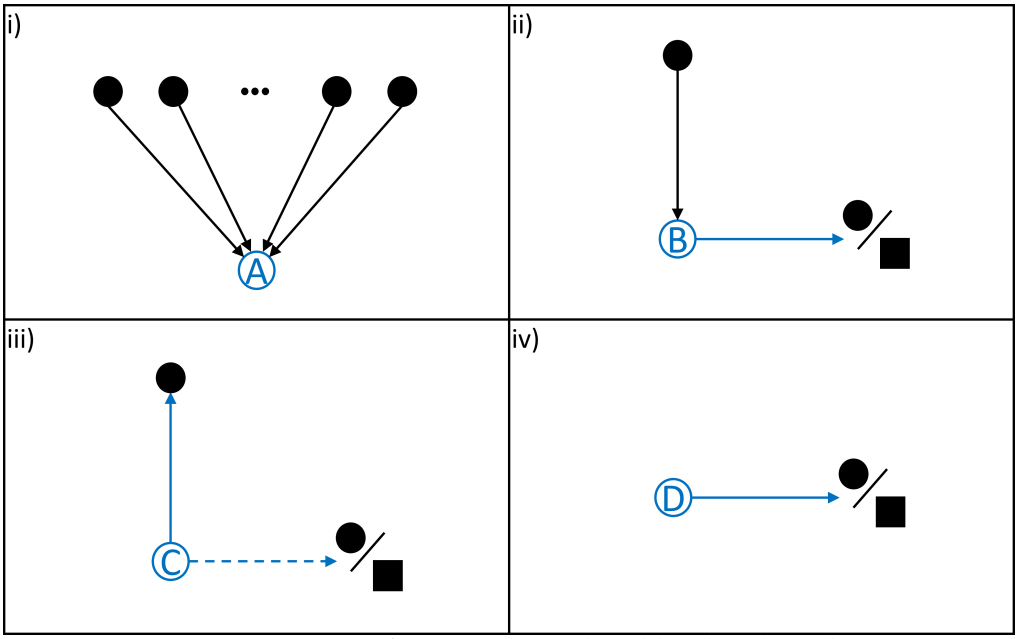


Figure 4.6: Types of vertices considered by the algorithm.

- **Type** Ⓐ: Vertices with no outgoing edges and any number of incoming edges. This case is shown in Figure 4.6 *i*).

  $f_R(\text{Ⓐ}) = 0$. Because f_R only computes outgoing edges, the values of the incoming edges are computed on the adjacent vertices. Hence, clearly $f_R(\text{Ⓐ})$ is a lower bound for the visit that the vertex represents.

- **Type** Ⓑ: Vertices with one incoming edge and one outgoing edge to either ● or ■. Figure 4.6 *ii)* present this vertex type.

  $f_R(\text{Ⓑ}) = \textit{shortest edge cost}$. This represent the case where a worker comes from another visit (distance not computed in this vertex) and leaves to the closest point. Because only the shortest edge is computed, clearly f_R represent a lower bound for that visit.

- **Type** Ⓒ: Vertices with two outgoing edges, where the shortest

edge is adjacent to a ● vertex and the second shortest is adjacent to either ● or ■. Figure 4.6 *iii)* illustrates this case.

$f_R(\text{C}) =$ *summation of the costs of the two shortest edges*. Because the shortest edge leads to a ●, it is guaranteed that another edge (the second smallest) must be selected because a worker cannot come from and leave to the same place unless it is a ■ vertex (note that multiple visits in a same location will be represented as multiple vertices). Because the two outgoing edges present minimum costs, f_R constitute a lower bound for that vertex.

- **Type** (D): Vertices with no incoming edges and the shortest outgoing edge to either ● or ■. Figure 4.6 *iv)* present this vertex type. $f_R(\text{D}) =$ *shortest edge cost*. Analogously to type (B), f_R represents a lower bound for this type of vertices as well.

Note that although we state that the f_R applied to individual vertices constitute individual lower bounds for the respective visits, it is not true that simply having a subgraph with only vertices of these types will outcome a lower bound for the RRS. Take for example the graphs in Figure 4.7. In *i)* it shows an RSS graph. All edges shown are either the shortest or the second shortest (note that there may be ties). The problem has only one worker and let us assume that this worker is qualified to perform all visits. On *ii)* we see the optimal route considering only distances (provided all visits can be successfully sequenced without time conflicts). On *iii)* we have all ● vertices classified as either (A) or (C), but in total we have 12 edges in the graph (more than the 9 present in the optimal solution) and even though they all have minimum costs, the sum of its costs is still higher than the optimal ($17 > 15$), hence that graph is not a lower bound. On *iv)* we have a different classification of vertices such

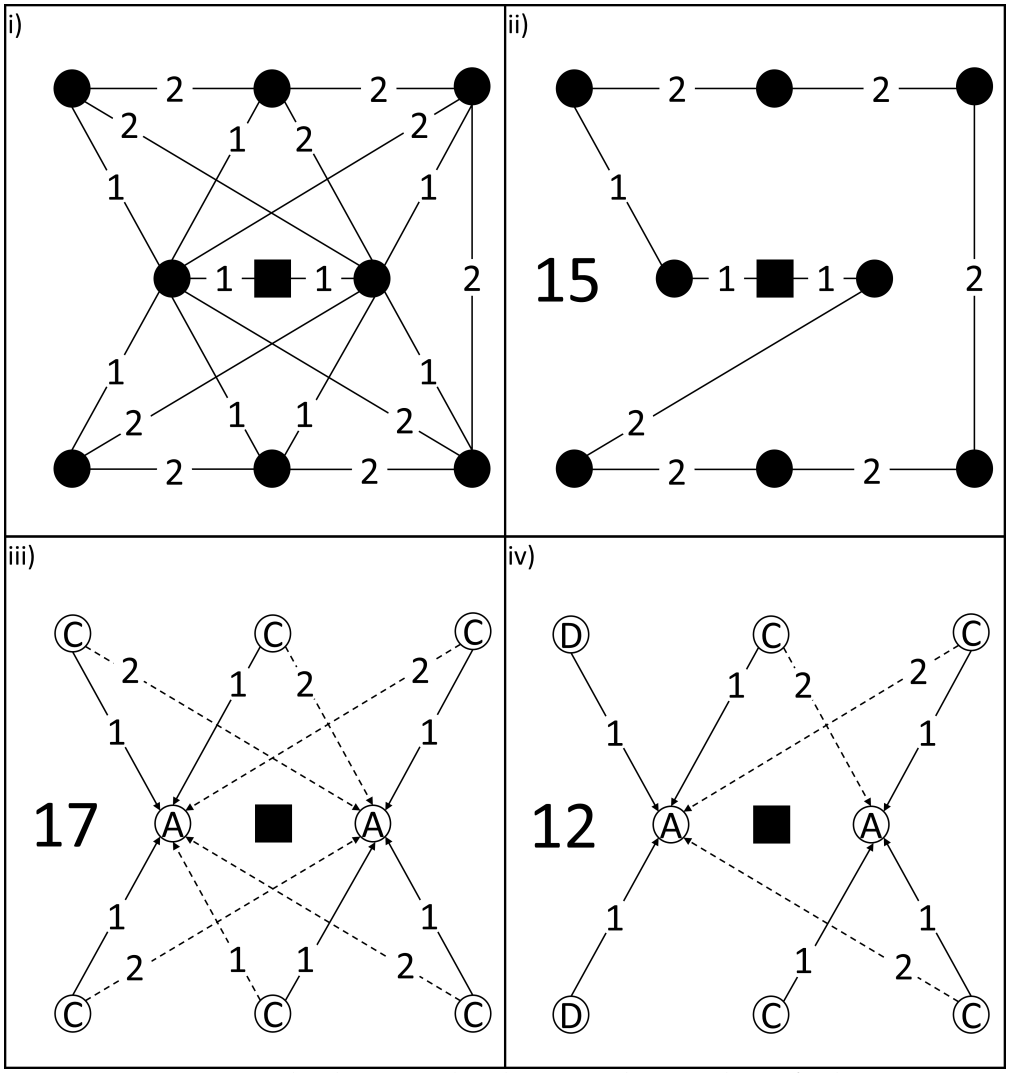


Figure 4.7: Example showing that limiting the number of edges is required. On *ii)* an optimal solution is shown. On *iii)* a valid classification of the vertices that incurs an invalid lower bound is shown. On *iv)* a valid classification where the number of vertices is retricted presents a lower bound.

that the number of edges in the subgraph is equal to $|T| + 1$, which is the exact number of edge in the case a single worker can perform all visits. Because all selected edges have minimum costs, the resulting graph is a lower bound for the optimal solution. Moreover, on real world problems, it is likely that multiple workers will be required to cover all visits, hence $|T| + 1$ is likely to be lower than the number of edges selected in the optimal solution.

Algorithm 7 presents the pseudocode for the proposed lower bound

Algorithm 7 RRSDistancesLowerBounds

Require: RSS R
Ensure: Lower bound value for the distances of R
1: $G_R = (V, E, \omega) \leftarrow$ minimum directed subgraph of R
2: $G'_R = (V' = V, E' = \emptyset, \omega)$
3: **for** $v \in V'$ **do**
4: **if** $|E'| < |T| - 1$ **and** $v =$ Ⓒ **then**
5: $E' \leftarrow E' \cup \{$any two minimum cost edges sourcing on $v\}$
6: **else if** $|E'| < |T|$ **and** $v =$ Ⓑ **then**
7: $E' \leftarrow E' \cup \{$any minimum cost edge sourcing on $v\}$
8: **else if** $|E'| < |T|$ **and** $v =$ Ⓓ **then**
9: $E' \leftarrow E' \cup \{$any minimum cost edge sourcing on $v\}$
10: **else**
11: $v =$ Ⓐ
12: **end if**
13: **end for**
14: **return** $f_R(G'_R)$

calculation technique. The algorithm starts by building the minimum directed subgraph G_R of R (line 2) and subgraph G'_R containing all vertices of G_R with an empty set of edges (line 3). Note that at this stage, G'_R already represents a lower bound because $|E'| = 0 < |T| + 1$ and all ● vertices are of type Ⓐ. Next, for every vertex $v \in V'$, the algorithm checks if two more edges can be added to E' and if the current vertex v is of the type Ⓒ, i.e. it has no incoming edges in E' and if its smallest cost edge connects to a ● vertex (line 5). In case the conditional is satisfied, the two smallest cost edges are added to E' (line 6). Note that if more than two edges present minimum (or second-minimum) costs, the algorithm arbitrarily decides which ones to include. If the condition is not satisfied, then the algorithm proceeds to test if a single edge can be added to E' and if the current vertex is of type Ⓑ, i.e. it has a single incoming edge (line 7). In case the condition is met, a minimum cost edge outgoing from v is added to E' (line 8). If the previous condition was not met, then the algorithm proceeds to test if an edge can still be added to E'

and if the current vertex is of type Ⓓ, i.e. it has no incoming edges (line 9). If that is the case, a minimum cost edge outgoing from v is added to E' (line 10). Finally, if no previous conditions were met, then the current vertex is of type Ⓐ, i.e. it might have any number of incoming edges but it doesn't possess any outgoing edge. The algorithm finishes after assigning types to all vertices. At the end, it returns the sum of costs of all edges in E', $f_R(G'_R)$.

Examples:

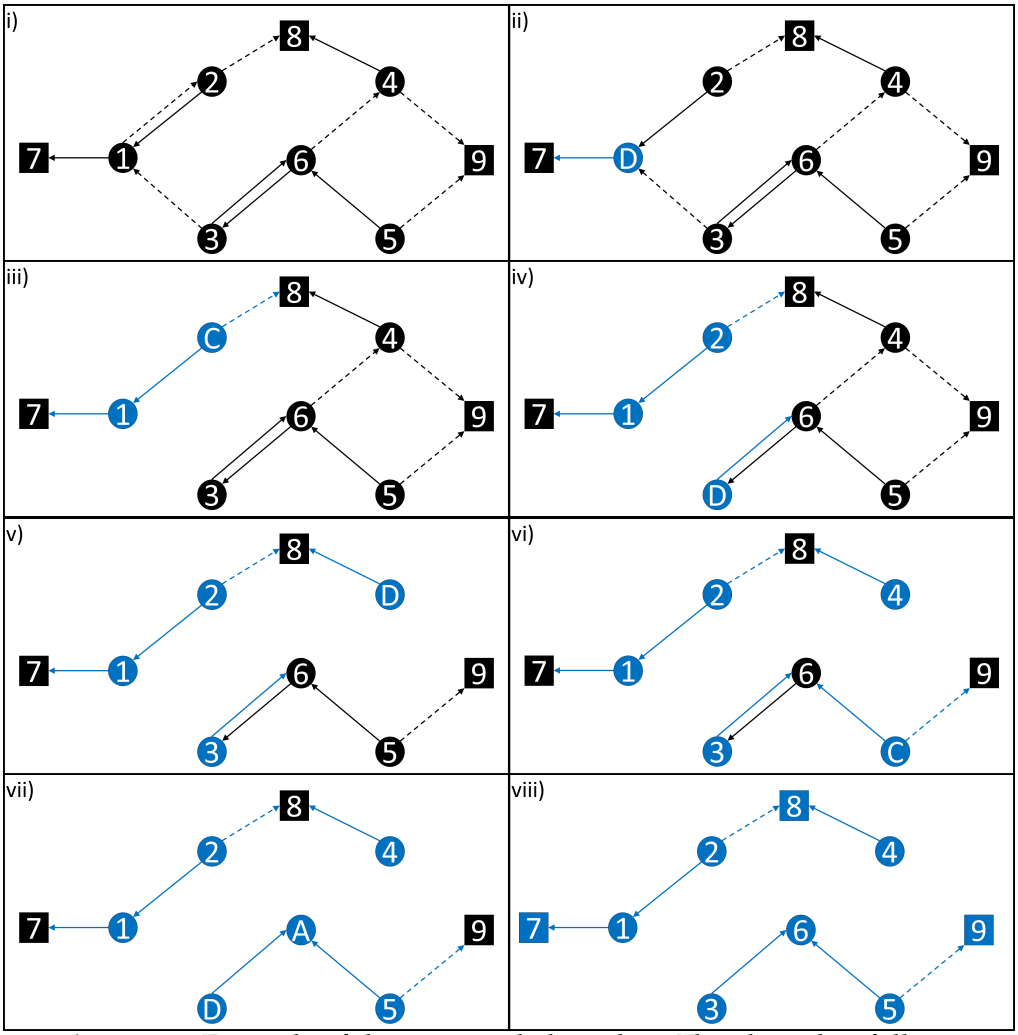


Figure 4.8: Example of the proposed algorithm. The algorithm follows the vertex numbering to iterate and classify all vertices.

Figure 4.8 presents an example of the execution of the algorithm. In

i) we have the graph G_R with six ● vertices and three ■ vertices. Solid edges represent the edges with minimum cost and dashed edges represent the edges with second minimum cost. Following the vertex numbering order, in *ii)* vertex ❶ is classified as type Ⓓ. Note that ❶ cannot be Ⓒ because its minimum cost outgoing edge is adjacent to a ■ vertex and it cannot be classified as Ⓑ because it has no incoming edges. In *iii)*, vertex ❷ is set as of type Ⓒ. Note that ❶ is now of type Ⓑ because it 'received' an incoming edge from ❷. In *iv)*, vertex ❸ has only one outgoing edge and no incoming, hence it is of Ⓓ type. In *v)*, vertex ❹ cannot be of type Ⓒ because its minimum cost edge is connected to a ■ vertex. Also, ❹ has no incoming edge, hence it is of type Ⓓ. In *vi)*, ❺ is of type Ⓒ. In *vii)*, because ❻ has two incoming edges already, it is of type Ⓐ. Finally, *viii)* presents the final graph with the selected edges.

The algorithm, however, is sensitive to the vertex ordering. Take for instance the example presented in Figure 4.9, presenting the same graph of Figure 4.8, but now running in inverse vertex order. In *ii)*, vertex ❻ is assigned type Ⓒ. Next (*iii)*), ❺ does not have its smallest edge available, hence it can only be of type Ⓐ. In *iv)*, ❹ cannot be of type Ⓒ because it has an incoming edge, hence it is assigned type Ⓑ. In *v)*, vertex ❸ has only its secondary edge available, hence it can only be of type Ⓐ. In *vi)*, ❷ is assigned type Ⓒ and in *vii)*, ❶ is assigned type Ⓑ because it has an incoming edge from ❷.

The ordering presented in Figure 4.8 induce a graph with seven edges, while the ordering used in Figure 4.9 induce a graph with six edges. Also, a single ● vertex can be classified as different types, which, at the end can provide different lower bounds. The investigation of the impact of vertex ordering runs out of the scope of this work, but we acknowledge that search algorithms could provide alternatives to explore the

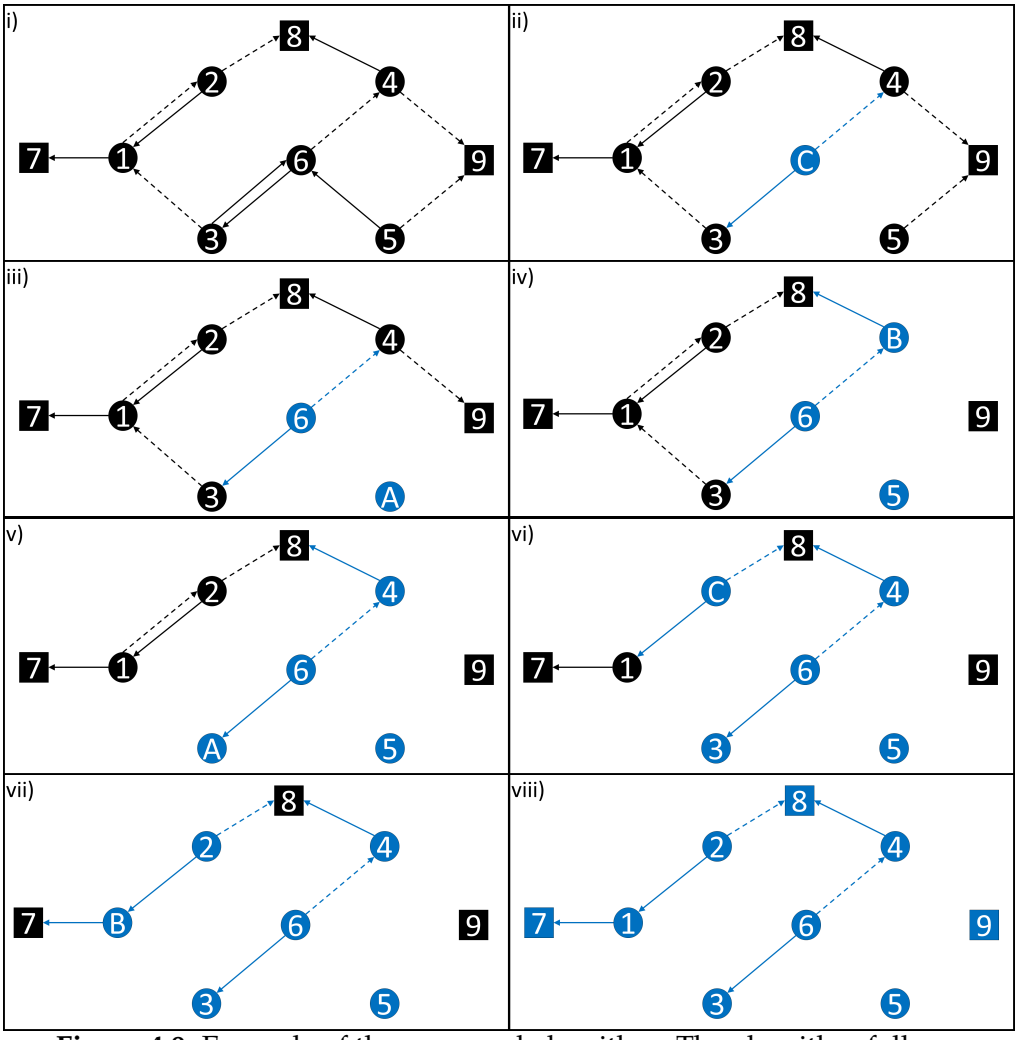


Figure 4.9: Example of the proposed algorithm. The algorithm follows the reverse vertex numbering to iterate and classify all vertices.

different possibilities and find an improved lower bound for the RRS.

Unassigned Visits

The proposed technique to calculate the lower bounds for the RRS when used to provide a lower bound for the distances of the HHC problem requires the set of unassigned visits in the optimal solution of the HHC problem to be known beforehand. Take the example of Figure 4.10. In *i)* we have a graph consisting of a problem with four visits and a single worker. The distances between locations are presented. Now, suppose

that we know that one visit will be left unassigned. In *ii)* the unassigned visit is ❹, hence the worker can visit all locations and back home with a total travelled distance of four units. In *iii)*, the visit ❶ is left unassigned, hence the worker can visit all remaining locations and go back home travelling a total of six units. Therefore, if the proposed algorithm does not know beforehand which visits are left unassigned, it cannot provide a valid lower bound.

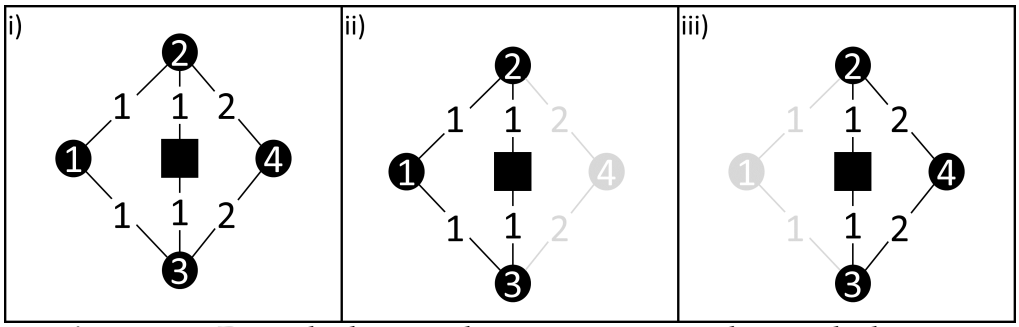


Figure 4.10: Example showing that it is necessary to know which visits are left unassigned in the optimal solution of the HHC.

Nonetheless, in most cases it is possible to know beforehand which visits are left unassigned. The algorithm provided in Section 4.2.1 is able to identify the number of unassigned visits in the optimal solution of the HHC problem, but in most cases it cannot pinpoint all visits. However, in cases where a conflicting clique consisting of one vertex is identified, then the corresponding visit is necessarily left unassigned. If only conflicting cliques of one vertex are found, then we know exactly which visits are unassigned. That algorithm, however, has exponential time complexity if run for cliques of all sizes.

Another possibility is to identify the conflicting cliques of size one and then check the unassigned visits identified against an upper bound solution provided by any other mean [4, 139, 176]. If the unassigned visits in the upper bound solution match the ones identified by the size one cliques, we know that all other visits will be assigned in the optimal

solution.

4.4 Lower Bounds for the HHC problem

In order to validate our approach of obtaining a lower bound for the HHC problem from the lower bounds of a scheduling and a routing subproblem, we show that the sum of lower bounds for each subproblem is a lower bound for the main HHC problem. Lemmas 8 and 9 define the objective function of both the RSS and RRS as a function of the main HHC problem.

Lemma 8. *Let X be a solution feasible for both the HHC problem W and its RRS S. Let $f_W(X)$ be the objective function of W applied to X and $f_S(X)$ be the objective function of S applied to X. Let $f_d(X)$ be the function of the weighted distances of X in $f_W(X)$ such that,*

$$f_d(X) = \sum_{k \in K} \sum_{i \in V^S} \sum_{j \in V^N} \lambda_1 x_{i,j}^k d_{i,j}.$$

Then,

$$f_S(X) = f_W(X) - f_d(X)$$

Lemma 9. *Let X be a solution feasible on both HHC problem W and its relaxed routing subproblem R. Let $f_W(X)$ be the objective function of W applied to X and $f_R(X)$ be the objective function of R applied to X. Let Y_X be the set of unassigned visits by the solution X. Let $f_d(X)$ be the function of the weighted distances of X in $f_W(X)$ and $f_R(X)$ such that,*

$$f_d(X) = \sum_{k \in K} \sum_{i \in V^S} \sum_{j \in V^N} \lambda_1 x_{i,j}^k d_{i,j}.$$

Then,

$$f_R(X) = f_d(X) + \lambda_4|Y_X|$$

Next, Lemma 10 joins the objective-function of both problems applied to their respective optimal solutions.

Lemma 10. *Let X_W^* be an optimal solution of the HHC problem W, X_S^* be an optimal solution of the relaxed scheduling subproblem S of W and X_R^* be an optimal solution of relaxed routing subproblem R of W. Let $Y_{X_R^*}$ be the set of unassigned visits of X_R^*, then,*

$$f_S(X_S^*) + f_R(X_R^*) - \lambda_4|Y_{X^R}| \leq f_W(X_W^*)$$

Proof. By Lemma 8 we have that $f_S(X_W^*) + f_d(X_W^*) = f_W(X_W^*)$ and by Lemma 9 we have that $f_R(X_W^*) - \lambda_4|Y_{X_W^*}| = f_d(X_W^*)$. Additionally, $f_S(X_S^*) \leq f_S(X_W^*)$ and $f_R(X_R^*) \leq f_R(X_W^*)$, hence $f_S(X_S^*) + f_R(X_R^*) - \lambda_4|Y_R| \leq f_W(X_W^*)$. □

Note that, because $f_S(X_S^*)$ already contains the number of unassigned visits for any other reason than time conflicts, $f_S(X_S^*) + f_R(X_R^*)$ may count unassigned visits twice, hence the need to remove the $\lambda_4|Y_{X^R}|$ term.

Finally, Theorem 3 states that the sum of the lower bounds of the subproblems constitutes a lower bound for the combined problem.

Theorem 3. *Let Z_S^L, Z_R^L be lower bounds for the relaxed scheduling subproblem S and for the relaxed routing subproblem R of HHC problem W respectively. Suppose X_W^*, X_S^* and X_R^* are the optimal solution of W, S and R respectively, and $Y_{X_W^*}$ is the set of unassigned visits in X_W^*. Then,*

$$Z_S^L + Z_R^L - \lambda_4|Y_{X_W^*}| \leq f_W(X_W^*)$$

Proof. By the definition of lower bound, $Z_S^L \leq f_S(X_S^*)$ and $Z_R^L \leq f_R(X_R^*)$. Hence, by Lemma 10, $Z_S^L + Z_R^L - \lambda_4 |Y_{X_W^*}| \leq f_S(X_S^*) + f_R(X_R^*) - \lambda_4 |Y_{X_W^*}| \leq f_W(X_W^*)$. □

Now that we defined the subproblems and showed that the sum of lower bounds represent a lower bound for the combined problem, we present the experimental results with the proposed algorithms.

4.5 Experimental Results

In order to evaluate the proposed technique, we performed experiments using the datasets presented in Chapter 3. Note that scenarios A, B and instances C2, C4, C5 and C7 are small enough to be solved to optimality using MIP. Also, to assess the efficiency of each component, we run the experiments using several configurations of the proposed algorithms. Moreover, all of the following algorithms include the lower bounds obtained for the RRS subproblem. We now list each algorithm:

- RSS**:** the basic RSSGreedySolver algorithm without further checking for unassigned visits or time and area conflicts.
- LB_U**:** the RSSGreedySolver algorithm using the conflict graph to detect unassigned visits.
- $LB_{UA/T}$**:** the RSSGreedySolver algorithm using the conflict graph to detect unassigned visits, a conflict graph to detect area availability violations and a third conflict graph to detect time availability violations.
- LB_{UAT}**:** the RSSGreedySolver algorithm using the conflict graph to detect unassigned visits and a second conflict graph combining

both time and area availabilities such that the listed workers for a vertex only include workers that are both available in time and area.

Finally, in order to evaluate the quality of the lower bounds provided by the algorithms, we compare the results against the optimal solution for the scenarios where that solution could be obtained. For the remaining instances we compare the results against the solution provided by a tailored metaheuristic that will be described later in this thesis.

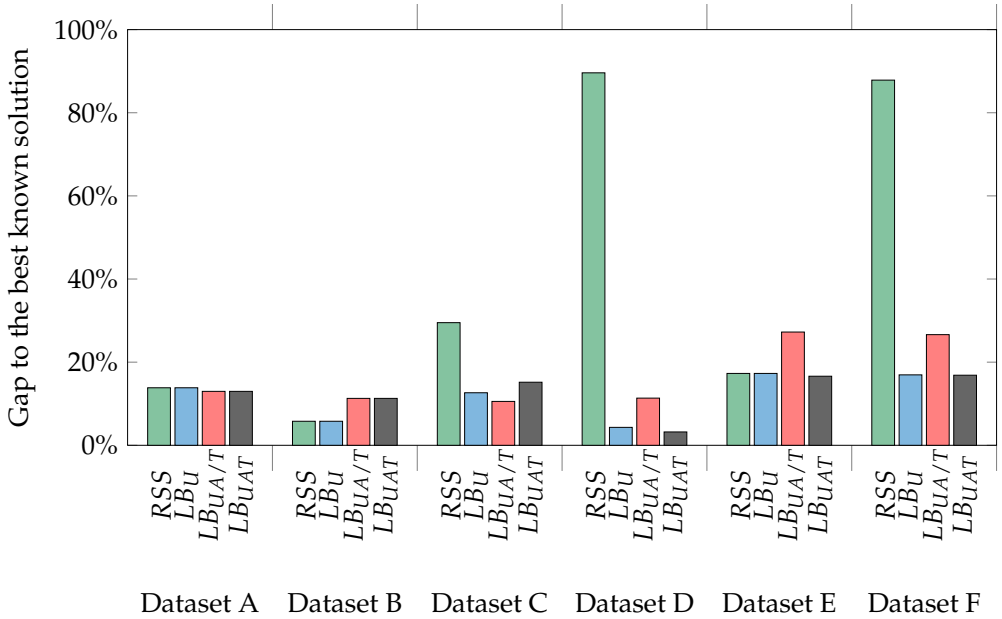


Figure 4.11: Summary of the comparison between all algorithms on all datasets.

Figure 4.11 presents the summary of the results (a detailed table of results is shown later in the chapter). On the y axis we have the gap to the best-known solution, whilst on the x axis we have the grouped bars of all algorithms for all datasets. Each data point represents the average gap of the instances in that dataset for that algorithm. We see that the quality of the results slightly variates among datasets, highlighting that they each possess different characteristics and that these characteristics have an impact on the proposed algorithms. We see that no variant of

the algorithm is the best in all scenarios.

On dataset A, all algorithms present similar performance with a slight edge to the composite algorithms LB_{UAT} and $LB_{UA/T}$. Dataset B presents the opposite scenario where using the composite algorithms outcomes worse lower bounds. This happens because all solutions in dataset A show time availability violations, whilst in dataset B no such violations occur. Hence, in dataset B, when time availability is ignored on the RSS-GreedySolver for the LB_{UAT} and $LB_{UA/T}$ algorithms, workers that are not available are selected for some tasks (but the penalty for time availability violations not computed) and later the conflict graph cannot spot any violations, hence resulting in a worse lower bound.

Datasets C, D and F present best-known solutions when using the conflict graph to identify unassigned visits, hence the RSS alone provides the worst results due to the algorithm inability to identify these visits. On datasets D, E and F, the LB_{UAT} presented the best results, showing that combining time and area availability into a single conflict graph is beneficial in most cases, however, that is not the case on dataset C. On that dataset, $LB_{UA/T}$ presents the best results. This happens because most workers are unavailable at the time and in the area of the visit. Hence, because the computation of the combined conflict graph counts an assignment that violates both area and time only once, counting them in two different conflict graphs provides improved results.

When we presented the conflict graph technique to identify unassigned visits and availability violations, we proposed the use of a parameter to limit the maximum size of the investigated cliques because finding cliques in a graph is an NP-hard problem. Figure 4.12 presents the results of increasing this limit on the $LB_{UA/T}$ algorithm. The y axis present the percentage of cumulative improvements in the lower bound

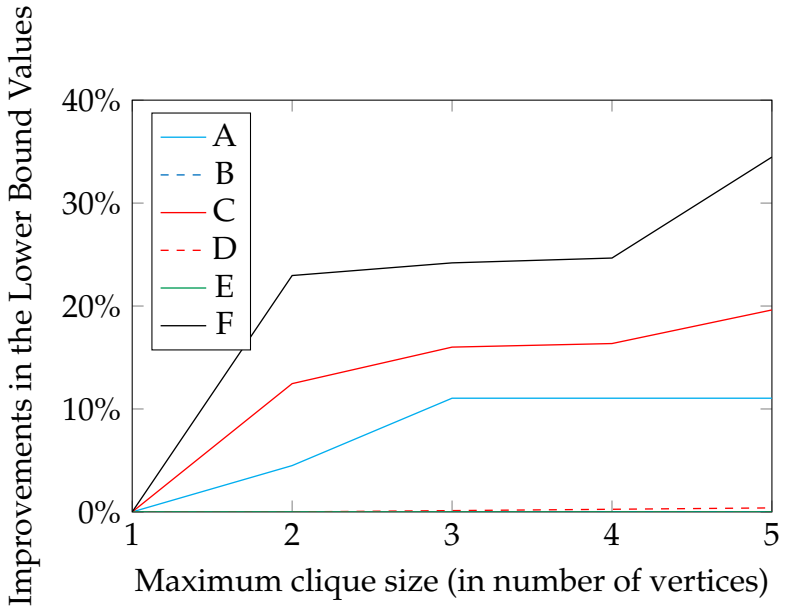


Figure 4.12: Cumulative improvements on the lower bounds obtained by varying the clique size of the $LB_{UA/T}$ algorithm in relation to using a size 1 clique.

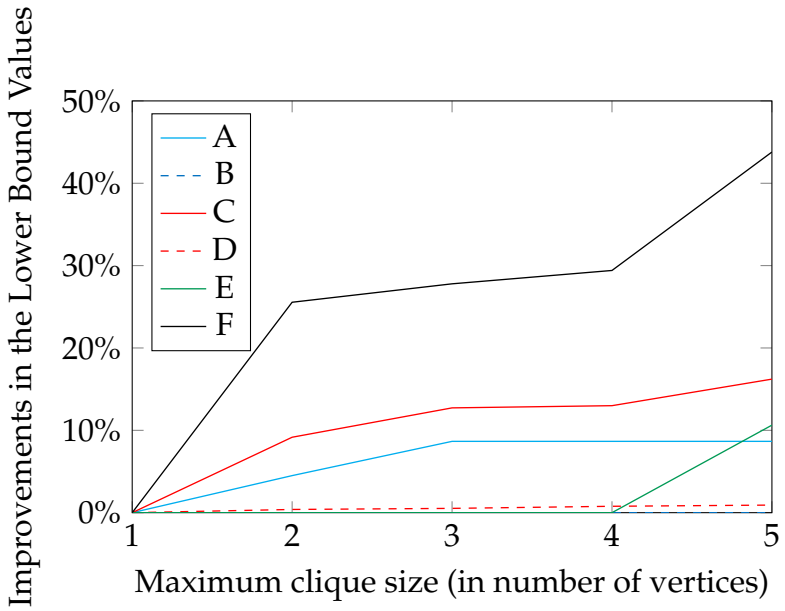


Figure 4.13: Cumulative improvements on the lower bounds obtained by varying the clique size of the LB_{UAT} algorithm in relation to using a size 1 clique.

and the x axis present the maximum size of the investigated cliques. On datasets B, D, E, increasing the clique size from two to five made no difference. However, on datasets A, C and F, an increased clique size provided better results.

Regarding runtime, for a maxumim clique of five vertices, datasets A, B and C (except C6) run in less than a minute. The instance C6 required 37 minutes. On average, datasets D and E required 4 minutes on each instance and dataset F required from 15 minutes for F1 up to 123 minutes for F7. Therefore, we could not run experiments for larger clique sizes because of the computational time demanded, however, the chart indicates that further increasing the size of the cliques on datasets C and F can potentially improve the results obtained.

Analogously, Figure 4.13 presents the results on the clique size limit for the LB_{UAT} algorithm. In this case, datasets A, C, E and F benefited of a higher clique size.

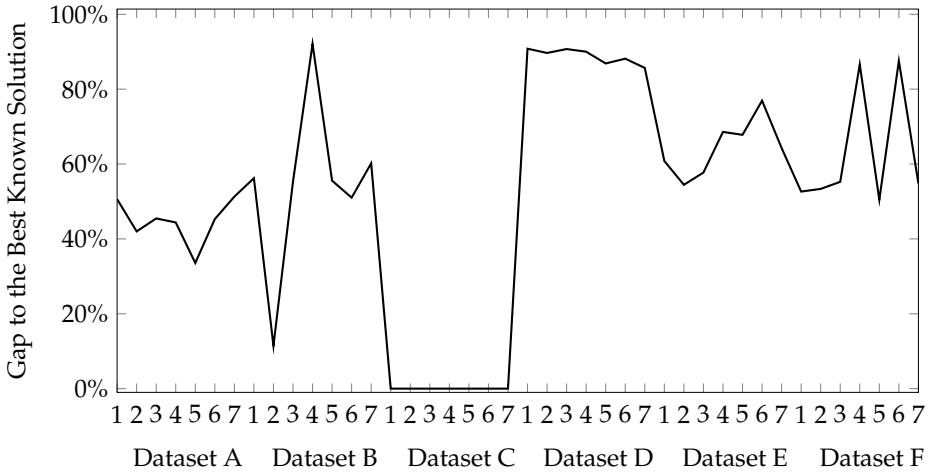


Figure 4.14: Gap from the distances lowerbounds obtained by the RRS algorithm against the distances presented in the best known solution.

Finally, Figure 4.14 presents the results of the proposed algorithm to calculate the lower bounds for the RRS, compared against the distances presented in the best-known solution. Clearly, the algorithm struggles to perform well because it does not consider many of the scheduling constraints. Note that on dataset C the gap is zero only because that dataset does not contain distances information. Still, due to the nature of the objective function employed, which has distances on the lowest cost tier,

the high gap found by this technique does not have a significant impact on the overall lower bound calculation. However, when the weight of the sum of distances is increased, the weakness of the method become evident. Nonetheless, in future work, we aim to investigate search methods to explore the search space in order to obtain improved distances lower bounds.

Table 4.1 presents the detailed results of the lower bounds obtained. The gap presented in the last column compares the best-known solution against the tightest lower bound found. Overall the best results were found for dataset D and the worst for dataset A. Only seven of the 42 instances present results worse than 10%, while a third of the instances present a gap smaller than 3% and the majority of the instances settles below 6%. The average gap found is of 6.8%, evidencing that the better technique for each instance presents a good mechanism to calculate the lower bounds for the HHC problem.

4.6 Conclusion

In this chapter we proposed a technique to calculate the lower bounds for the combined problem of scheduling and routing. The method splits the problem into two relaxed subproblems and tackles them separately. The scheduling subproblem is solved by a greedy algorithm. Later, an algorithm that uses a conflict graph and cliques identifies unassigned visits and availability violations. The routing subproblem is solved by building a graph and selecting edges following a set of rules and vertices classification.

We presented the results of the technique applied to the 42 instances presented in Chapter 3. Results showed that the proposed technique

Table 4.1: Detailed results for all algorithms. The distances are included in the results.

Instance		Lower Bounds				Best Known Solution	Gap†		
		RSS	LB_U	$LB_{UA/T}$	LB_{UAT}				
A	1	**3.39**	**3.39**	3.34	3.34	*3.49**	2.8%	10.3%	6.8%
	2	**2.28**	**2.28**	2.23	2.23	*2.49**	8.4%		
	3	2.28	2.28	**2.71**	**2.71**	*3.00**	9.6%		
	4	**1.29**	**1.29**	1.14	1.14	*1.42**	9.3%		
	5	**1.76**	**1.76**	1.70	1.70	*2.42**	27.2%		
	6	**3.51**	**3.51**	3.41	3.41	*3.55**	1.2%		
	7	2.81	2.81	**3.22**	**3.22**	*3.71**	13.3%		
B	1	**1.63**	**1.63**	1.59	1.59	*1.70**	4.2%	5.8%	
	2	**1.72**	**1.72**	1.51	1.51	*1.75**	1.9%		
	3	**1.62**	**1.62**	1.57	1.57	*1.72**	5.5%		
	4	**1.91**	**1.91**	1.70	1.70	*2.07**	8.1%		
	5	**1.68**	**1.68**	1.61	1.61	*1.82**	7.9%		
	6	**1.55**	**1.55**	1.53	1.53	*1.62**	4.3%		
	7	**1.64**	**1.64**	1.54	1.54	*1.79**	8.5%		
C	1	54.99	98.74	**99.56**	98.56	*114.71*	13.2%	7.4%	
	2	**3.05**	**3.05**	2.74	2.31	*3.15**	3.3%		
	3	49.98	87.48	**89.27**	88.27	*104.53*	14.6%		
	4	**11.12**	**11.12**	10.33	10.83	*11.15**	0.2%		
	5	11.87	11.87	**12.12**	11.15	*12.34**	1.8%		
	6	9.17	88.17	**146.07**	142.57	*179.44*	18.6%		
	7	**4.30**	**4.30**	4.09	3.59	*4.30**	0.0%		
D	1	27.71	156.71	137.49	**158.49**	*167.32*	5.3%	3.2%	
	2	20.76	141.26	125.97	**142.97**	*159.87*	10.6%		
	3	16.57	171.82	161.19	**174.17**	*177.45*	1.8%		
	4	15.75	155.25	145.44	**158.44**	*165.37*	4.2%		
	5	15.57	157.57	147.91	**160.61**	*160.70*	0.1%		
	6	13.02	176.27	167.17	**176.66**	*177.38*	0.4%		
	7	13.02	176.77	167.67	**177.15**	*177.39*	0.1%		
E	1	1.07	**1.07**	0.95	0.95	*1.16*	7.7%	8.0%	
	2	1.08	**1.08**	0.97	0.97	*1.17*	8.1%		
	3	1.10	**1.10**	1.00	1.00	*1.19*	7.3%		
	4	1.17	**1.17**	0.99	0.99	*1.27*	7.7%		
	5	1.13	1.13	1.02	**2.02**	*2.22*	9.3%		
	6	1.20	**1.20**	0.98	0.98	*1.29*	6.4%		
	7	1.11	1.11	1.03	**1.53**	*1.70*	9.8%		
F	1	77.17	1934.17	1965.96	**1982.96**	*2009.84*	1.3%	5.9%	
	2	67.67	1965.18	2008.89	**2035.87**	*2070.68*	1.7%		
	3	129.20	520.20	512.04	**575.54**	*598.79*	3.9%		
	4	121.24	1253.74	1232.53	**1315.53**	*1318.37*	0.2%		
	5	50.29	50.29	73.04	**127.04**	*181.34*	29.9%		
	6	94.27	**516.77**	507.03	612.53	*617.36*	0.8%		
	7	145.67	**3374.42**	606.73	690.19	*3484.44*	3.2%		

* *Optimal solution provided by a mathematical solver*

† *Gap of the best lower bound found in relation to the best known solution*

manages to find good lower bounds with a gap averaging only 6.8%, with a couple of solutions presenting gaps as high as 29.9% . The technique, however, presents one limitation on each subproblem: regarding the scheduling subproblem the method is dependent on finding cliques in a conflict graph, a well-known NP-hard problem. Regarding the routing subproblem the method could not provide better results because it does not consider most scheduling constraints. Nonetheless, on the majority of the instances, the results obtained were satisfactory.

Therefore, the algorithms presented in this work can be utilised to provide lower bound values for unknown HHC instances. The technique proposed was able to find results even for the largest scenarios where using a mathematical solver is not possible because the problem is too big to be loaded into memory.

The methodology and algorithms proposed in this chapter can be adapted and applied to obtain lower bounds for different optimisation problems. In fact, many WSRPs that deal with fixed visit times can benefit from this approach, such as scheduling of technicians and security workers. Moreover, the algorithms to calculate the lower bounds for the RSS can be applied to scheduling problems with fixed times such as job scheduling [7]. Analogously, the algorithms for the RRS can be adapted to provide lower bounds for routing problems.

Chapter 5

Understanding Relationships Between Objectives in Optimisation Problems

The development of solution techniques for multiobjective optimisation problems (MOPs) has witnessed many improvements in recent years, partly prompted by real-world applications, but also due to the increasing computational power available; affordable computers are now much more capable of performing the complex computations which are required to solve such problems. Within this field, the study of problems which possess many-objectives (more than three) calls for special attention: when the number of objectives grows, the number of trade-off solutions may grow exponentially, hence designing algorithms to effectively solve these problems can be very challenging [80, 185]. One option to tackle this problem is found in the single-objective problem literature: the design of algorithms that include domain knowledge and are tailored to the specific problem. This approach requires a deeper understanding of the optimisation problem under consideration. In the pres-

ence of multiple objectives, it is important to understand the relationships between objectives and the fitness landscape of a given problem. Knowles and Corne [130] discussed this in the context of the multiobjective quadratic assignment problem, and Castro-Gutierrez et al. [46] studied the objectives' relationships in the multiobjective vehicle routing problem with time windows.

The multiobjective optimisation literature often focuses on problems in which the multiple objectives exhibit strongly conflicting nature, as these present an increased challenge for many algorithms, hence being a motivation for applying multiobjective techniques. However, when the number of objectives grows, conflicts could be presented locally rather than globally. A *global relationship* holds throughout the majority of the search space and is often easy to identify. A *local relationship* exists in a restricted region of the search space and can be difficult to spot. Understanding local relationships is a useful tool to design tailored algorithms [83]. Additionally, with an increased number of objectives, composite relationships may emerge, i.e. relationships between three or more objectives that occur locally.

It is difficult to assess the multiobjective nature of MOPs and identifying trade-offs is very important for this assessment. However, the existence of multiple objectives may not guarantee the occurrence of trade-offs. Additionally, when tackling multiobjective scenarios, information about trade-offs and relationships between objectives can impact on the choice of solution techniques and algorithmic design – objectives can be grouped or removed, trade-offs can be systematically explored, or the search could be biased towards regions of interest. Moreover, most common benchmark MOPs found in the literature are generated to emulate some sort of behaviour (usually conflicting), hence the fitness land-

scape is often known beforehand. Examples of this are the ZTD family of functions [230] and DTLZ test suite [62]. However, for real world applications, especially combinatorial problems, the fitness landscape is usually unknown and learning its nature may potentially give designers substantial advantages for developing specialised algorithm.

Purshouse and Fleming [183] discussed some common techniques for analysing and visualising relationships between objectives in MOPs, including scatter plots and parallel coordinates (graphical representations), Kendall correlation [125] (a quantitative metric), amongst others. Khabzaoui et al. [126] presented statistical measures. Previous work has also applied some of these techniques to improve the understanding of relationships in specific problems, such as Castro-Gutierrez et al. [46] who used the Kendall correlation metric to assess the multiobjective vehicle routing problem with time windows and Ishibuchi et al. [113] who assessed many-objective problems with correlated objectives. However, these techniques are best suited for problems with two objectives and fail to discover insights into problems with a larger number of objectives.

In this chapter, a technique to analyse and visualise relationships (both local and global) between objectives in multiobjective problems is introduced. This technique aims to improve the understanding of the fitness landscape of MOPs by assessing different perspectives of the problem objectives. In order to be applied, the technique requires an approximation set of non-dominated solutions to be supplied by some means. A novel visualisation tool based on Karnaugh maps [119] is proposed to visualise the relationships between the many objectives. The analysis technique is performed in four steps: first the global pairwise relationships are analysed using the Kendall correlation method; then the ranges of the values found on the given Pareto front are estimated

and assessed; next these ranges are used to plot a map using Gray code, similar to Karnaugh maps, that has the ability to highlight the trade-offs between multiple objectives; and finally local relationships are identified using scatter-plots.

5.1 Motivation and Related Work

The topic of analysing fitness landscapes of MOPs is not new to the literature and important progress has been made in the past. Most previous work focuses on adapting single-objective techniques and methods to multiobjective scenarios. Some of the main contributions in this field are outlined here.

Huband et al. [108] identified the need for properly evaluating MOPs to validate algorithm development. In their work, they extend the knowledge of several benchmark problems. They assess the problems regarding multiple criteria, including the need of external or medial parameters, the scalable number of parameters and objectives, the problems being dissimilar with regard to parameter domains and trade-off ranges, the knowledge of the Pareto optima, the optimal geometry, the parameter dependencies, the bias, the mappings and the modality. Although they provide a comprehensive assessment of the problems studied, the evaluation is specifically useful for the generation of benchmarking scenarios.

Korhonen and Wallenius [132] surveyed the most common methods used in multiobjective decision-making frameworks, including multivariate statistical methods. They suggest the use of charts and graphs as visual aids in the decision-making process, including bar and line charts to visualise multiple solution vectors. In their work they also raise

the importance of the development of advanced techniques to further aid the decision-making process. In the same book, Lotov and Miettinen [150] described techniques for visualising the Pareto optimal set of many-objective problems, including scatterplots and heatmaps.

Garrett [84] and Garrett and Dasgupta [83] adapted single-objective landscape analysis techniques such as analysis of the distribution of Pareto optima, fitness distance correlation, ruggedness, random walk and the geometry of the objective space to obtain insights useful to the tailoring of multiobjective evolutionary algorithms. They apply the proposed techniques to two-objective quadratic and generalised assignment problems. They conclude that using hybrid algorithms that explore knowledge about the landscape provides performance gains compared to more general approaches.

A graphical approach to evaluate the quality of a Pareto front based on ranks of objectives was proposed by Brownlee and Wright [35]. According to their assessment, the technique may not be suitable for large solution sets as it is based on an individual analysis of solutions.

Verel et al. [215] adapted single-objective landscape analysis techniques to set-based multiobjective problems with objective correlation. Later, Verel et al. [216] conducted a study of the landscape of local optima in such problems. Verel et al. [217] proposed to carry out *a priori* analysis of a problem by evaluating the problem size, its epistasis, the number of objectives and the correlation values between objectives, to suggest the best way to tackle it. They concluded that, depending on the problem features, different types of algorithms (scalar or Pareto approaches) and sizes of the solution archive should be employed.

Recently, different methods to visualise solutions, such as scatter plots, parallel coordinates and heat maps were reviewed by Walker et al. [219].

They proposed two techniques: a data mining visualisation tool to plot a convex graph, and a new similarity measure of solutions to plot them in a two-dimensional space. Moreover, multiple visualisation techniques for many-objective approximation sets were surveyed by Tusar and Filipic [214]. They also proposed a visualisation method based on orthogonal projections of a section and they further employ it on four-dimensional approximation sets. Other visualisation techniques include objective wheels, bar graphs and colour stacks, as explored by Anderson and Dror [5].

Giagkiozis and Fleming [88] proposed a technique to estimate the Pareto front of continuous optimisation problems, and then used the estimated front to obtain values for the decision variables of interesting solutions. They applied it to convex benchmark problems. The concept involves obtaining an initial solution set (using any multiobjective algorithm) and then calculating a projection matrix of the optimal Pareto set.

Finally, Castro-Gutierrez et al. [46] assessed the well known Solomon [203] dataset of the Multiobjective Vehicle Routing Problem with Time-Windows (MVRPTW) using the objectives pairwise correlation analysis proposed by Purshouse and Fleming [183] and concluded that those instances are not suited for multiobjective benchmarking purposes. They then proposed a new set of benchmark instances for the MVRPTW and showed that those instances exhibit an interesting multiobjective nature. These instances are further evaluated in the next chapter, using the new analysis technique proposed here, showing that the results are consistent with their claim and providing further insights into those datasets.

It can, therefore, be observed that an improved understanding of fitness landscapes of MOPs can promote the development of better solution methods. Additionally, analysis and visualisation of fitness land-

scapes and relationships in MOPs is a topic of significant interest for researchers. The technique proposed in this chapter seeks to make a contribution in this area.

5.2 Objective Relationships in Multiobjective Optimisation

This research focuses on the investigation of relationships between objectives in MOPs by analysing non-dominated approximation solution sets and their coverage of the objective space. Additionally, the concepts proposed by Purshouse and Fleming [183], of conflicting, harmonious and independent objective relationships, are used.

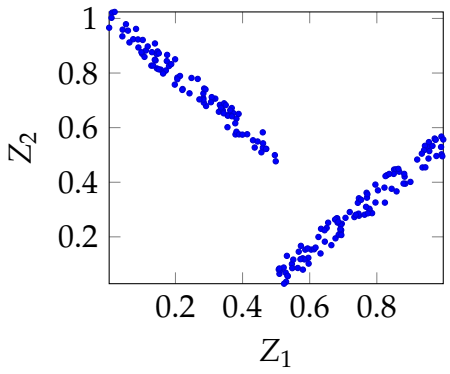


Figure 5.1: Example of complex relationship between two objectives Z_1 and Z_2 in a 3-objectives optimisation problem.

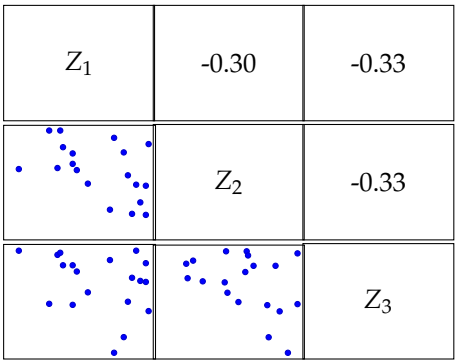
Much of the work in the literature relies on the study of pairwise relationships between objectives [46, 97, 113]. However, results obtained solely from this technique can be misleading. Take for instance Figure 5.1, which shows the scatter plot of two minimisation objectives Z_1 and Z_2 for a scenario with three objectives (Z_3 is omitted for simplicity).

Considering the global objective space, using techniques such as the Kendall correlation technique, it would be easy to conclude that the objectives Z_1 and Z_2 are independent. However, it is clear that, in fact, the

objectives are conflicting when $Z_1 < 0.5$ and harmonious when $Z_1 \geq 0.5$. This misleading loss of information is a consequence of considering the global rather than local objective space, thus identifying only *global relationships* between objectives.

(45, 68, 85), (6, 63, 99), (34, 64, 95),
(28, 100, 48), (98, 47, 69), (48, 62, 79),
(72, 24, 90), (82, 79, 16), (36, 100, 97),
(100, 87, 41), (98, 19, 87), (85, 57, 50),
(88, 20, 73), (91, 48, 99), (94, 31, 70),
(56, 49, 59), (75, 93, 1), (38, 84, 85),
(45, 78, 47)

(a) Non-dominated points.



(b) Scatter-plot matrix and correlation coefficients.

Figure 5.2: Three-way conflicting objectives

Additionally, common representations fail to highlight *composite relationships* between more than two objectives. Figure 5.2 shows evidence of that. Figure 5.2a shows a set of 19 non-dominated points where green values are above 50 and red values are equal to or below 50. It is clear that no point has three values simultaneously above 50. Figure 5.2b shows the scatter plot and the Kendall correlation coefficients for each pair of objectives. The scatter plot and correlation values do not help to identify the three-way trade-off. Likewise, it is clear that the correlation values do not indicate any strong pairwise correlation.

To better analyse and visualise the multiobjective nature of optimisation problems, techniques are needed that will help in identifying global, local and composite relationships between objectives as well as interesting trade-offs in the fitness landscape.

5.3 A Four-Step Analysis and Visualisation Technique

This is a four-step method to analyse and visualise objectives relationships in MOPs. The following requirements must be met in order to apply the technique:

1. Obtain an approximation solution set or subset of the instances of the tackled problem. This can be obtained by any multiobjective algorithm (*MOA*), provided that the quality of the solutions are good enough (i.e. solutions are well spread, numerous and close-enough to the optimal Pareto front). A combination of different methods is suggested here, to increase the probability of good solutions. Multiple MOAs are publicly available and frameworks like JMetal [69], ParadiseEO [39] and MOEA Framework [98] provide efficient implementations of several state of the art algorithms that may be applied to different problem domains.

2. Obtain knowledge of the problem domain and the importance of different objective values. Steps two and three particularly require this, when defining meaningful ranges, quality thresholds and regions of interest.

The aim of the proposed technique is to aid the study of subsets of problem instances, to help the design of more effective tailored solutions for solving other instances of the same problem. The focus of this work is to investigate the suitability of the technique for that purpose, hence it is applied to scenarios for three distinct MOPs.

Although the need to solve the problem first could be seen as a weakness, there are often similarities between problem instances from the

same source, especially for real world problems, thus insights into one set of instances can give insights into many other instances. Solving many-objective problems is computationally expensive, thus any help in tailoring fast techniques that can provide 'good-enough' results is important, and this technique may allow a user to identify similarities between instances which could be exploited. Moreover, the increased understanding of the problem could help to identify strengths and weaknesses for novel algorithms.

Each step in the analysis brings to light more information regarding any relationships between objectives, the coverage of the feasible region and the trade-off landscapes. The details of the four steps are presented in the remainder of this chapter, the results of the analysis of three distinct problems are presented in the next chapter: a multiobjective multidimensional knapsack problem, a multiobjective nurse scheduling problem and a multiobjective vehicle routing problem with time windows.

5.3.1 Step 1 – Global Pairwise Relationship Analysis:

The available approximation set of non-dominated solutions is first analysed using Kendall correlation coefficients [125], to identify global pairwise relationships, as proposed by [183]. The existence of strongly conflicting correlations (values < -0.5) immediately indicates that a trade-off surface exists. Analogously, the existence of strongly harmonious correlations (values > 0.5) indicates that objectives could be aggregated. Objectives identified as independent only prove that the objectives are not globally dependent, but do not imply the absence of local trade-offs in the fitness landscape.

If independence is detected, the problem could be assessed for the possibility of decomposing the decision variables according to the objec-

tives in order to solve each objective (or groups of objectives) separately, as such an approach has been observed to improve performance [184].

5.3.2 Step 2 – Objective Range Analysis:

Next, the range (the difference between best and worst values) of each objective in the approximation set is calculated. The problem domain knowledge or expertise of the decision-maker is then used to assess how to approach each objective; e.g. assign it to a cluster, remove it, etc. This approach assumes that there is some way to categorise solutions into high vs low quality, or acceptable vs unacceptable to the decision maker. As such the presence of a domain expert, or at least of sufficient domain knowledge to be able to determine this, is important.

A *meaningful objective* is an objective in which the range is high enough such that solutions can be classified into multiple quality categories (good to bad, high to low, acceptable to unacceptable, etc) depending on the problem domain. The existence of that type of objective indicates the presence of a trade-off in the fitness landscape. Analogously, a *non-meaningful objective* is an objective in which the range is low to the point that its variability can be considered negligible according to the problem domain, thus not worth exploring further.

Again, the categorisation of solution quality is important, to avoid issues of scaling factors, for example. An objective with a high absolute range of values across the non-dominated solutions, but where all such values are considered by decision makers to be good (if such is possible), would be a non-meaningful objective. Conversely, even a binary objective where it is important for the objective to have the value of 1 would be meaningful if some of the non-dominated solutions had a value of 0 for the objective.

One way to tackle non-meaningful objectives is to ignore them. With a low enough variability across the approximation set, it is possible that optimising for the other objectives will still result in acceptable values for the non-meaningful objectives. This may be the result of composite relationships or of the characteristics of the data in the scenario. Note that having a low range across the initial solution set does not imply that the range is negligible for the entire objective space. To assess whether this is the case, remove this objective from the MOA, determine a new approximation set, and calculate the ranges of the excluded objective in the new solution set. If the new range is also negligible, then it is likely that the objective can be ignored.

Another option to tackle non-meaningful objectives is to cluster them [97]. Clustering the objectives will reduce the size of the objective space, but the MOA will still keep some pressure towards optimising these objectives. Arguably the overall results for these objectives may not be as good as solving them separately, however considering that they all presented low ranges, it is possible that the new solutions will present similar ranges as before.

5.3.3 Step 3 – Trade-off Regions Analysis:

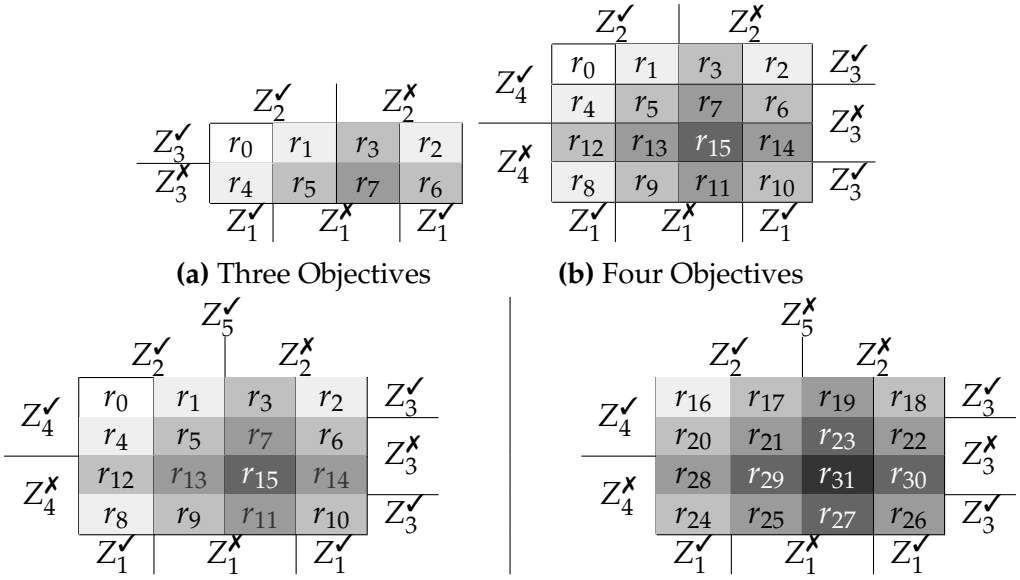
The third step consists of a quantitative method based on Karnaugh maps [119] to classify the objective space into regions of interest to aid in identifying trade-offs and complex relationships between objectives. A Karnaugh map is a method to visualise and simplify boolean algebra expressions using a truth table. A map for i variables possesses 2^i cells. The cells labelling follows the Gray code, therefore any two adjacent cells differ in only one bit. For example, in a three variables scenario, the cells adjacent to cell 0 (000_2) are cells 1(001_2), 2(010_2) and 4(100_2). Karnaugh

maps are used to group variables because they make it easy to visualise patterns.

Given an approximation set, to build its region map it is necessary to first define, for each objective Z_i where $(i = 1, 2, \ldots m)$, a threshold t_i such that values above (maximisation problem) t_i are considered good or acceptable, and values below t_i are considered bad or inadequate. The threshold can be obtained using domain knowledge or empirically – for example, using the average value for each objective as its threshold. Since the threshold position may be important for problems, it can be worth considering alternative positions for the threshold, as discussed later.

Next, each objective value in each solution is classified as good (✓) or bad (✗) according to the objective's threshold. For example, a solution $(Z_1 = ✓, Z_2 = ✗, Z_3 = ✓)$ in a three-objective scenario indicates that for Z_1 and Z_3 the objective values are better than their respective thresholds, while Z_2 presents a value which is worse than its threshold.

A *region map* is then drawn. This is similar to a Karnaugh map, but instead of boolean variables, the objectives of the MOP are used. These are arranged such that the cells of the map follow the Gray code (replacing 0's and 1's for ✓and ✗). The map is built with 2^m regions where each region represents a single combination of good and/or bad objectives. Each cell in the map represents a region r_k using a binary encoding such that the least significant digit represents Z_1 and the most significant digit is Z_m. Finally, the region is identified for each solution (considering ✓ $= 0$ and ✗ $= 1$) and the number of solutions in each region is determined. The solution of the previous example, $(Z_1 = ✓, Z_2 = ✗, Z_3 = ✓)$, falls into region r_2 because $2_{10} = 010_2$. Region r_0 represents solutions with good values in all objectives



(a) Three Objectives **(b)** Four Objectives

(c) Five Objectives

Figure 5.3: Region map schematics.

Figure 5.3 presents the region map schematics for 3, 4 and 5 objectives. Each region is labelled according to the Gray code. Take for instance Figure 5.3a. Suppose that the decision maker is interested in solutions where $(Z_1 = \checkmark)$ regardless of the other objectives. Then, they just need to look at the first (r_0, r_4) and last (r_2, r_6) columns, indicated by $Z_1^{\checkmark}$. Alternatively, if the decision maker is interested in $(Z_2 = \checkmark)$ and $(Z_3 = \checkmark)$, they need to look where the columns indicated by $Z_2^{\checkmark}$ (r_0, r_4 and r_1, r_5) overlap the line $Z_3^{\checkmark}$ (r_0, r_1, r_3, r_2), which in this case are the cells (r_0, r_1) . Additionally, a shading scheme is used to display the differences between regions, such that regions with lighter tones possess a higher number of good objectives than regions with darker tones, thus, the lightest coloured region (r_0) has only good objective-values and the darkest coloured region (r_7, r_{15} and r_{31} in Figures 5.3a, 5.3b and 5.3c respectively) has no good objective-values.

The use of these maps makes it easy to spot where solutions are concentrated in the objective space and the existence of trade-offs. When region r_0 is not empty, then there are solutions with good values in all ob-

jectives, meaning that the problem could possibly be tackled with single-objective algorithms. Alternatively, that could mean that the threshold could be increased until trade-offs appear. Performing a range analysis in this region could provide further insights into the best approach to be taken. Conversely, if most solutions fall into region r_{2^m-1}, then it is possible that the threshold is set too high and should be lowered for more accurate results. If there are no solutions in r_0, but they are scattered throughout the map, then trade-offs have been identified and the map can be used to visualise them.

Threshold Analysis

As previously mentioned, moving the threshold for objectives may sometimes have value in clarifying and further investigating the trade-offs. Region maps from modified threshold values can be used to determine additional information about the multiobjective nature of the problem, including:

- The maximum threshold such that all solutions are considered good in all objectives. This information can be useful for assessing whether the problem could be tackled as a single-objective problem.

- The minimum threshold to ensure the existence of trade-offs. This information could be used to empirically define a threshold in cases where domain knowledge is not available, replacing the expert knowledge required for that.

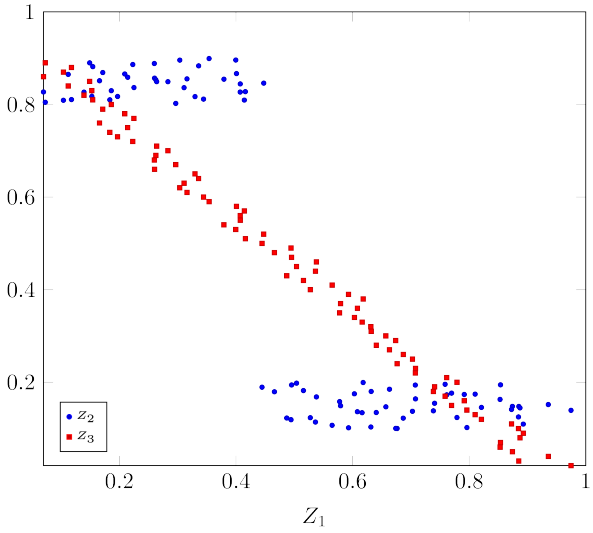
- The steepness of the decrease in the number of solutions for all objectives across the thresholds. A higher steepness indicates that the instances evaluated are more alike in their value for that objective.

In this chapter, the following method is used for this analysis: First determine the maximum and minimum values for each objective, then determine $\alpha - 1$ equally spaced threshold values for the objective which will divide this range into α equally ranged parts, where α is a value chosen by the user. A region map is then generated for each of these threshold values.

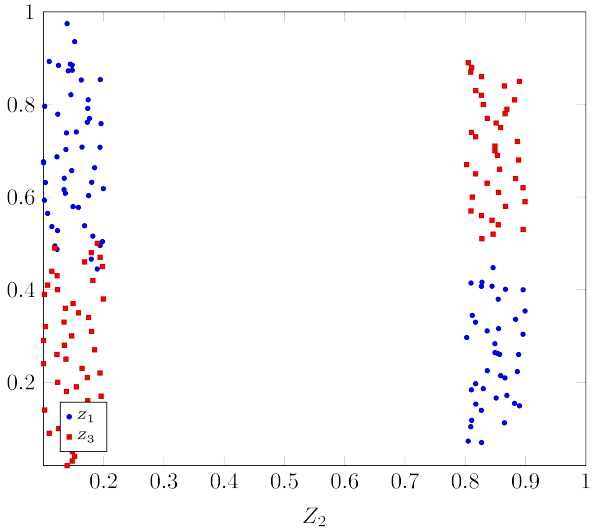
5.3.4 Step 4 – Multiobjective Scatter Plot Analysis:

Finally, a scatter plot graph of all objectives of the normalised approximation set is assessed. The values are normalised and a scatter plot is drawn of all of the solutions. One objective is selected as a *pivot* and plotted on the x axis. The remaining objectives are drawn distinctively on the y axis. Ideally, the pivot objective is well spread throughout its range in order to avoid gaps in the graph and the solutions being too concentrated in a small region, which would make the resulting graph difficult to read. The previous steps can be used to identify good objectives to be used as pivot: highly conflicting objectives (step 1) are potentially good candidates as well as objectives with large ranges (step 2) or objectives with good solutions in a large number of regions (step 3) as these are indications of spread out values. It may be interesting to test different objectives in order to identify which objective provides more useful information, or whether multiple objectives provide different insights.

Figure 5.4 illustrates the impact of choosing the wrong objective as the pivot. In a) Z_1 is chosen and the conflicting relationship between Z_1–Z_3 can be clearly visualised, as well as the small range of values of Z_2. In b) Z_2 was chosen as the pivot and the conflicting relationship between Z_1–Z_3 is not prominent anymore. Additionally, in b) all values are situated within a small region of the plot space because Z_2 has values



(a) Scatter plot of a three-objective approximation set using Z_1 as pivot.



(b) Scatter plot of a three-objective approximation set using Z_2 as pivot

Figure 5.4: Importance of choosing the right pivot on a three-objective approximation set: in a) Z_1–Z_3 conflicting relationship is easily spotted, while in b) the same relationship can not be visualised.

ranging only from [0.1,0.2] and [0.8,0.9].

The pattern recognition capacities of the human brain can now be utilised with a visual inspection of the resulting graph(s) to identify local relationships (conflicts and harmony), interesting patterns, gaps in the objective space, and well-spread trade-offs or isolated regions. This

analysis can provide useful knowledge that can be used to tailor algorithms for the problem. Additionally, if multiple instances of a problem present consistent landscapes, then the information obtained and the algorithm designed to solve these instances could potentially be of increased use for solving other unknown instances of this problem as well.

5.3.5 The Reliability of the Approximation Set

For the proposed technique to work, it is important to highlight that when the quality of the obtained approximation set is poor, the evidence raised by the analysis might be the indication of biased algorithms rather than of intrinsic characteristics of the problem and its fitness landscape. For example, this means when algorithms fail to provide solutions in certain regions of the solution space, one must be careful not to imply that these gaps are inherent to the problem. The same can be said about peculiar local relationships as they might be artefacts of the solving algorithms.

In order to minimise these issues, in this thesis, they are addressed as following:

- The approximation sets attained by different algorithms and multiple runs are combined to provide a single archive of non-dominated solutions. In particular, solutions are obtained with the MOEA/D algorithm, which decomposes the problem into several weight vectors and the NSGA-II algorithm, which uses ϵ-dominance to create an archive of non-dominated solutions. There is no guarantee that both algorithms will not be biased as the solution representation is the same. However, it is reasonable to assume that the probability

of that happening is reduced.

- The conclusions drawn using the proposed technique are taken by assessing not a single instance, but entire datasets. This measure brings more confidence in the findings because it is less likely that erratic or biased search mechanisms will present the same results on distinct instances. Therefore, if an odd fitness landscape is repeated across several instances, the probability of that being caused by the algorithm is smaller.

5.4 Case Studies

5.4.1 Multiobjective Multidimensional Knapsack Problem (MMKP)

In the multiobjective multidimensional knapsack problem [152], there are n items $(i = 1, \ldots, n)$ with p weights w_j^i $(j = 1, \ldots, m)$ and m profits c_k^i $(k = 1, \ldots, p)$. A set of items must be selected to maximise the m profits while not exceeding the capacities W_j of the knapsack. This problem can be formulated as follows:

$$\begin{aligned} \text{maximise} \quad & \sum_{i=1}^{n} c_k^i x_i && k = 1, \ldots, m \\ \text{subject to} \quad & \sum_{i=1}^{n} w_j^i x_i \leq W_j && j = 1, \ldots, p \\ & x_i \in 0, 1 && i = 1, \ldots, n \end{aligned}$$

Five MMKP datasets are considered here, each of which has five instances, all with $p = 4$, $m = 4$, $n = 1000$ and $W_j = 50000$. The first four datasets were generated following the guidelines in Bazgan et al. [20] and are as follows:

Set A: Independent random instances where $w_k^i \in_N [1, 1000]$ and $c_k^i \in_N [1, 1000]$.

Set B: Uncorrelated harmonious instances where $w_k^i \in_N [1, 1000]$, $c_1^i \in_N [1, 1000]$ and $c_k^i \in_N [max\{c_{k-1}^i - 100, 1\}, min\{c_{k-1}^i + 100, 1000\}]$ for $k = (2, 3, 4)$.

Set C: Uncorrelated conflicting instances where $w_k^i \in_N [1, 1000]$, $c_1^i \in_N [1, 1000]$ and $c_k^i \in_N [max\{900 - c_{k-1}^i, 1\}, min\{1100 - c_{k-1}^i, 1000\}]$ for $k = (2, 3, 4)$.

Set D: Correlated conflicting instances where $w_1^i \in_N [max\{900 - |c_1^i - c_4^i|, 1\}, min\{1100 - |c_1^i - c_4^i|, 1000\}]$, $w_k^i \in_N [max\{900 - |c_k^i - c_{k-1}^i|, 1\}, min\{1100 - |c_k^i - c_{k-1}^i|, 1000\}]$, $c_1^i \in_N [1, 1000]$ and $c_k^i \in_N [max\{900 - c_{k-1}^i, 1\}, min\{1100 - c_{k-1}^i, 1000\}]$ for $k = (2, 3, 4)$.

Set X: Correlated special set where:

$$r^i \in_R [0,1]$$

$$c_k^i \begin{cases} c_1^i \in_N [900,1000], c_2^i \in_N [c_1^i,1000], c_k^i \in_N [0,100] \text{ for } k=(3,4). & \text{if } r^i \leq 0.1 \\ c_3^i \in_N [900,1000], c_4^i \in_N [c_1^3,1000], c_k^i \in_N [0,100] \text{ for } k=(1,2). & \text{if } 0.1 < r^i \leq 0.2 \\ c_1^i \in_N [900,1000], c_3^i \in_N [c_1^1,1000], c_k^i \in_N [0,100] \text{ for } k=(2,4). & \text{if } 0.2 < r^i \leq 0.3 \\ c_2^i, c_3^i \in_N [900,1000] \text{ and } c_1^i, c_4^i \in_N [0,100]. & \text{if } 0.3 < r^i \leq 0.4 \\ c_k^i \in_N [0,1000]. & \text{if } r^i > 0.4 \end{cases}$$

$$w_1^i = c_1^i + c_2^i + c_3^i$$

$$w_2^i = c_2^i + c_3^i + c_4^i$$

$$w_3^i = c_1^i + c_3^i + c_4^i$$

$$w_4^i = c_1^i + c_2^i + c_4^i$$

Set A contains only independent objectives. In set B all objectives are harmonious. Set C contains three pairs of conflicting objectives, (Z_2, Z_1), (Z_3, Z_2) and (Z_3, Z_4), while the weights are uncorrelated. Set D has conflicting objectives, like set C, but the weights are correlated to the objective values. The fifth set X was tailored to provide a misleading scenario.

5.4.2 Multiobjective Nurse Scheduling Problem (MONSP)

An extension of the nurse scheduling problem (NSP) presented by Maenhout and Vanhoucke [154] is also considered. It is important to notice that the datasets studied are artificially generated and do not reflect real world scenarios. Additionally, they were not conceived to provide any interesting multiobjective feature as data is uniformly distributed. The goal in evaluating these datasets is to assess whether the proposed anal-

ysis technique can successfully indicate their lack of multiobjective characteristics or not. The original problem consists of creating a work plan for a set of nurses, that must be assigned shifts across a time period (week or month) to cover hospital requirements. The problem is composed of a number of hard and soft constraints to be considered when generating each individual schedule. The hard constraints define strict work regulations and the soft constraints define working policies and desirable aspects.

- *Hard Constraints:* The NSP has two hard constraints, both of which aim to avoid certain shift patterns. The first forbids the assignment of an afternoon shift before a morning shift. The second forbids the assignment of a night shift before a morning or an afternoon shift.

- *Soft Constraints:* There are four soft constraints that represent nurses' preferences. The first defines a minimum and maximum number of working days within the scheduling period. The second defines a minimum and maximum number of consecutive working assignments. The third defines a minimum and maximum number of assignments of each shift type in the scheduling period. The fourth soft constraint defines a minimum and maximum number of consecutive shift assignments of the same type.

- *Preferences:* The preferences represent the main objectives for the MONSP. Each preference is a numerical value relating to a nurse and a shift. It could represent the nurse preference regarding that shift, the hospital preference to assign that nurse to that shift, the patients' preferences, or a combination of these.

For this work the number of objectives is artificially extended using preferences and soft constraint violations.

The MONSP can be defined as follows. Given a set N of nurses, where $N = \{n_1, n_2, \ldots, n_{max}\}$, a set D of days where $D = \{d_1, d_2, \ldots, d_{max}\}$ and a set S of shifts where $S = \{s_1, s_2, \ldots, s_{max}\}$, find an assignment of shifts for each day $d_i \in D$, such that each nurse $n_j \in N$ has been assigned a specific shift $s_k \in S$ and hard constraints are met. Shifts that refer to a given working period (early, day or night shift) or a rest period (free shift). Each tuple (*nurse,day,shift*) possesses a set of m values representing different preferences regarding that nurse being assigned to that shift on that day. The different preferences may represent personal preferences, the hospital preferences, patient preferences, costs, etc. The (multiple) objectives to optimise consist of the sum of each type of preference value, and the sum of a set of soft constraints.

NSPLib is a library with problem instances for the NSP [153]. Two types of files are present: base files, which describe the preferences; case files which specify the constraints.

The data from that library was used in this work, with the addition of extra base files to represent the new preference values. Four objectives are considered, so each instance is composed of the existing case file and base file, plus three new objective files. Table 5.1 presents the details for the different sets, where A to I are different base files (different preference values), Z_1 to Z_3 are the new preference objective values and Z_4 is the count of soft constraint violations.

- **Independent Set (IS):** There is no correlation between base file and objectives files (all objective files are distinct). The aim is that the fitness landscape will be different for different instances.

- **Objective-Dependent Set (ODS):** This set uses the same objectives files for all instances, but the base files change. This set as-

Table 5.1: Generational patterns for the MNSP using the NSPLib files. Letters represent base files.

Independent Set						Objectives-Dependent Set						Requirements-Dependent Set					
Case	**Base**	Z_1	Z_2	Z_3	Z_4†	**Case**	**Base**	Z_1	Z_2	Z_3	Z_4†	**Case**	**Base**	Z_1	Z_2	Z_3	Z_4†
1	A	A	B	C	V	1	A	A	B	C	V	1	A	A	B	C	V
1	D	D	E	F	V	1	B	A	B	C	V	1	A	D	E	F	V
1	G	G	H	I	V	1	C	A	B	C	V	1	A	G	H	I	V
2	A	A	B	C	V	2	A	A	B	C	V	2	A	A	B	C	V
2	D	D	E	F	V	2	B	A	B	C	V	2	A	D	E	F	V
2	G	G	H	I	V	2	C	A	B	C	V	2	A	G	H	I	V
3	A	A	B	C	V	3	A	A	B	C	V	3	A	A	B	C	V
3	D	D	E	F	V	3	B	A	B	C	V	3	A	D	E	F	V
3	G	G	H	I	V	3	C	A	B	C	V	3	A	G	H	I	V
4	A	A	B	C	V	4	A	A	B	C	V	4	A	A	B	C	V
4	D	D	E	F	V	4	B	A	B	C	V	4	A	D	E	F	V
4	G	G	H	I	V	4	C	A	B	C	V	4	A	G	H	I	V
5	A	A	B	C	V	5	A	A	B	C	V	5	A	A	B	C	V
5	D	D	E	F	V	5	B	A	B	C	V	5	A	D	E	F	V
5	G	G	H	I	V	5	C	A	B	C	V	5	A	G	H	I	V
6	A	A	B	C	V	6	A	A	B	C	V	6	A	A	B	C	V
6	D	D	E	F	V	6	B	A	B	C	V	6	A	D	E	F	V
6	G	G	H	I	V	6	C	A	B	C	V	6	A	G	H	I	V
7	A	A	B	C	V	7	A	A	B	C	V	7	A	A	B	C	V
7	D	D	E	F	V	7	B	A	B	C	V	7	A	D	E	F	V
7	G	G	H	I	V	7	C	A	B	C	V	7	A	G	H	I	V
8	A	A	B	C	V	8	A	A	B	C	V	8	A	A	B	C	V
8	D	D	E	F	V	8	B	A	B	C	V	8	A	D	E	F	V
8	G	G	H	I	V	8	C	A	B	C	V	8	A	G	H	I	V

† Z_4 is the number of soft constraints violation.

sesses whether keeping the objectives information across all instances will generate a standardised fitness landscape.

- **Requirements-Dependent Set (RDS):** This set employs a common base file for all instances but defines different objectives files for each instance. The goal of this scenario is to assess if keeping a problem base and changing the objectives information have an impact on the fitness landscape of the instances in this set.

Weekly problems with 25, 50 and 100 nurses were generated using random base and objective files following the generational pattern presented. Hence each set considered in this work is constituted of 75 instances totalling 225 problem instances.

5.4.3 Multiobjective Vehicle Routing Problem with Time Windows (MOVRPTW)

A Multiobjective Vehicle Routing Problem with Time Windows (MOVRPTW) is defined on a graph $G = (V, E)$ where V is the set of vertices representing the depot (vertex 0) and the customers (vertices $1 \dots n$) where each customer has a demand p_i ($i = 1, \dots, n$). The depot has h identical vehicles available, with capacity Q that must satisfy all demands from all customers. The edge set E denotes all possible connections between all vertices. Each edge (from vertex i to vertex j) has a cost that represents the distance or time, denoted by c_{ij}. Each customer must be served during a time window $[a_{ij}, b_{ij}]$. If a vehicle arrives early, it must wait until the beginning of the time window. Once the vehicle arrives at the customer, it stays there until the delivery is completed, for a duration known as the service time s.

Castro-Gutierrez et al. [46] proposed a benchmark set for the MOVRPTW. They consider five objectives: number of vehicles (Z_1), total travel distance (Z_2), makespan (Z_3), total waiting time (in case of early arrival) (Z_4), and total delay time (Z_5). They designed their instances based on different characteristics of the problem and each instance is a combination of these features. The features that constitute an instance are:

- **Number of customers:** 50, 150 and 250 customers.
- **Time window:** five different profiles ($tw0, tw1, tw2, tw3, tw4$) of time windows across eight hours were used. These are defined below in terms of minutes from the start of the working day 0 = 8:00am, 480 = 4:00pm, etc.). Customers are randomly assigned to one time-window:
  - $tw0$: [0,480].

  - $tw1$: [0,160],[160,320],[320,480].
  - $tw2$: [0,130],[175,305],[350,480].
  - $tw3$: [0,100],[190,290],[350,480].
  - $tw4$: all time-windows from $tw0, tw1, tw2$ and $tw3$.

- **Demand types:** three types of demand (10, 20, 30) uniformly distributed.

- **Vehicle capacity:** the capacity of the vehicles are calculated according to a δ parameter such that $Q = \underline{D} + \delta/100(\overline{D} - \underline{D})$. The dataset considers three δ values ($\delta 0 = 60$, $\delta 1 = 20$, $\delta 2 = 5$).

- **Service time:** three values of service time 10,20,30 uniformly distributed.

In total, there are 45 instances, and these can be downloaded from [47].

5.5 Application of the Four-Step Technique

To obtain the approximation sets, the MOEA/D [227] and NSGA-II [61] algorithms were chosen. Initial experiments showed that both algorithms struggled to produce good approximation sets on medium and large instances due to the difficulty of tackling many-objective problems, hence to improve the obtained Pareto fronts the following mechanism was applied to all datasets: for each instance, a single-objective genetic algorithm was executed for each objective alone, then both NSGA-II and MOEA/D algorithms were executed on each pair and triple of objectives. All of the non-dominated solutions obtained were then combined into a single archive which was used to initialise further executions of

the algorithms: Three runs of MOEA/D and three runs of NSGA-II were then executed, considering all objectives, where half of the individuals for the initial population were randomly drawn from the archive and the other half were randomly generated. The final approximation set was formed from all of the non-dominated solutions which were found in the process. Both NSGA-II and MOEA/D used a population of 200 individuals, binary tournament selection and half uniform crossover [72] for 1.5×10^6 function evaluations each run.

5.5.1 Multiobjective Multidimensional Knapsack Problem

After obtaining the approximation fronts, non-dominated sets were obtained, with approximately 900 combined solutions for each instance in set A, 3 for set B, 550 for for each instance in set C, 1500 for for each instance in set D and 2500 combined solutions for each instance in set X. Only a few non-dominated solutions were found for set B because the objectives there are strongly harmonious. Therefore, when maximising one of the objectives, the other objectives are also maximised, resulting in insufficient data for some of the analysis steps. However, the results are presented for completeness and illustrate that the number of solutions obtained can have a major impact on the analysis, and that it is important to have a comprehensive set, with enough well-spread solutions.

Step 1 - Global Pairwise Relationships Analysis. The results of this step are in Figure 5.5. Note that coefficient values for set B are not provided for the reason given above. The figure presents the individual pairwise correlation value for each combination of objectives. As expected for fully independent objectives, set A has values close to 0. The values for sets C and D are also predictably close to either 1 or -1 indi-

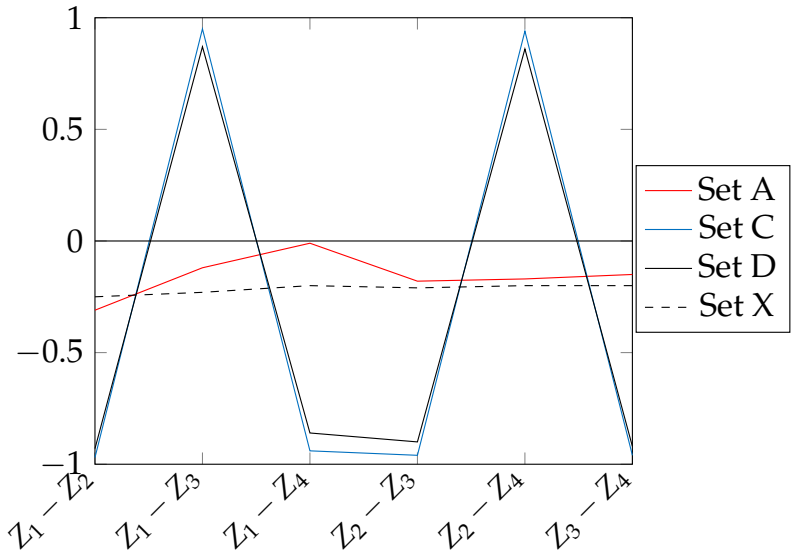
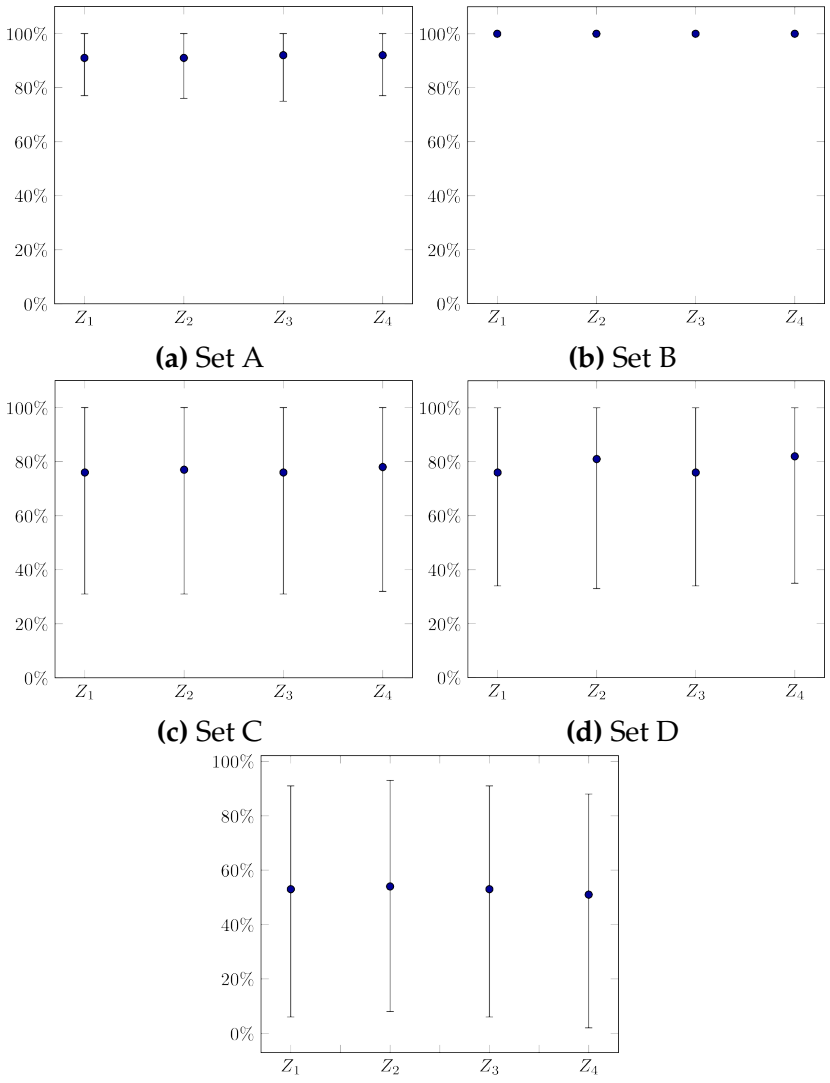


Figure 5.5: Pairwise correlations for each set of instances.

cating global conflicting or harmonious relationships. Set X has values similar to A – they do not reveal a strong global relationship between objectives as they are closer to 0 than to 1 or -1. Also, in set X it is not possible to decompose the decision variables according to the objectives, as every item has all weights and values above zero.

Step 2 - Objective Range Analysis. The results for this step are in Figure 5.6. Considering set B, it can be seen that the set presents small ranges of less than 0.3% on average. In this dataset, all objectives are harmonious and the solutions found are all located in a small region of the objective space. These few solutions dominated all other solutions explored. Both sets C and D present similar results to each other with large ranges for each objective (over 60%). This is expected since these are instances with conflicting objectives and present global trade-offs. The large ranges mean that while there are solutions with good values for a given objective, at least one other objective has a poor value.

While the global pairwise relationship analysis (step 1) hinted that sets A and X were similar, the difference between them now becomes clear with the results from step 2. In set A, each objective range is around



(a) Set A **(b)** Set B

(c) Set C **(d)** Set D

(e) Set X

Figure 5.6: Results for the objective ranges analysis. The central point shows the average value for the set, the upward lines represent the maximum value found and the downward line represents the minimum value.

24.0% of the maximum value – the smallest ranges excluding the harmonious instances – whilst in set X the ranges go up to 84.9%, the largest range found. Thus, it is apparent that the ranges for set X are actually closer to those for set D, a conflicting scenario with global trade-offs.

Step 3 - Trade-Off Regions Analysis. The results for this step are in Figures 5.7–5.11. First Figure 5.7 shows the number of instances with solutions in r_0 when the threshold is increased (normalised values). All

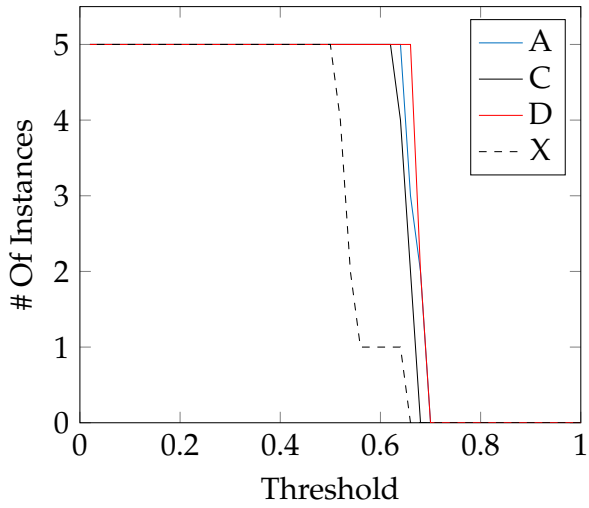


Figure 5.7: Threshold analysis showing the number of scenarios with solutions found in region r_0 when the threshold decreases.

instances present a similar threshold behaviour and with a threshold of roughly 0.7 there are no solutions with good values for all objectives. i.e. solutions with over 70% quality in all objectives could not be found. Also, the steepness of the lines in the chart is very acute, indicating that all instances from a same set have very similar fitness landscapes.

	$Z_2^{\checkmark}$		$Z_2^{\times}$		
$Z_4^{\checkmark}$	0.0%	0.3%	0.9%	0.1%	$Z_3^{\checkmark}$
	0.5%	5.3%	17.1%	4.6%	$Z_3^{\times}$
$Z_4^{\times}$	8.7%	7.4%	13.8%	11.6%	
	4.4%	8.5%	10.5%	6.5%	$Z_3^{\checkmark}$
	$Z_1^{\checkmark}$	$Z_1^{\times}$		$Z_1^{\checkmark}$	

(a) Overall distribution of solutions.

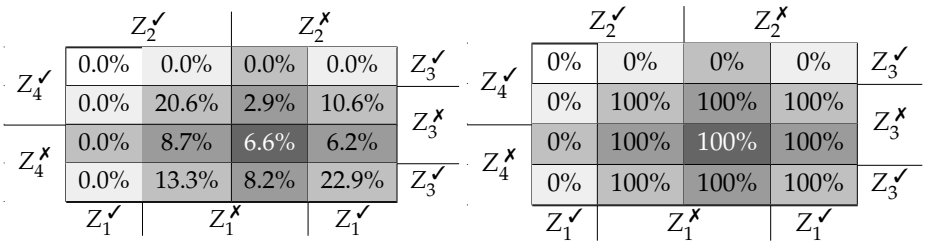
	$Z_2^{\checkmark}$		$Z_2^{\times}$		
$Z_4^{\checkmark}$	0%	20%	80%	20%	$Z_3^{\checkmark}$
	40%	100%	100%	100%	$Z_3^{\times}$
$Z_4^{\times}$	100%	100%	100%	100%	
	80%	100%	100%	100%	$Z_3^{\checkmark}$
	$Z_1^{\checkmark}$	$Z_1^{\times}$		$Z_1^{\checkmark}$	

(b) Frequency of instances.

Figure 5.8: Results for the trade-off regions analysis for Set A.

Region maps were calculated for each instance. The number of solutions in each region was computed and the map shows the average percentage of the solutions which lie in each region for each set. The range threshold was set to the minimum value such that there are no solutions in r_0.

In addition to the distribution of solutions map, the frequency of instances region map is also introduced here, displaying the percentage of

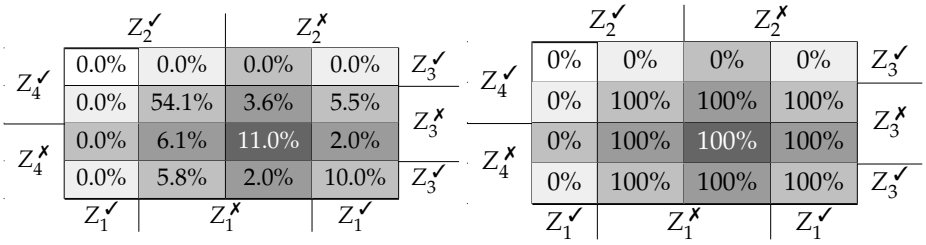


(a) Overall distribution of solutions.

(b) Frequency of instances.

Figure 5.9: Results for the trade-off regions analysis for Set C.

instances that contain at least one solution in a region. A value of 100% in a region means that in all instances at least one solution can be found in that region. In set *A* (Figure 5.8) the front is well distributed. Solutions can be found in all regions, scattered throughout the objective space, as a result of the independent objectives.



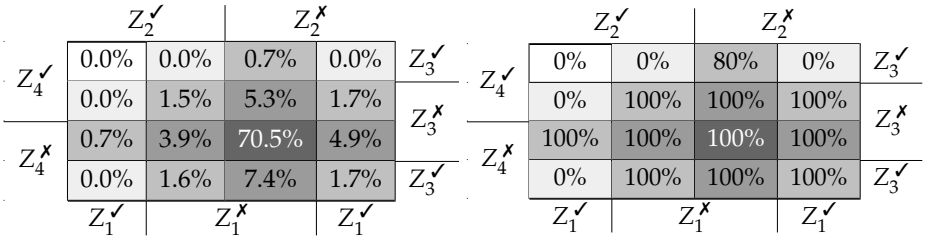
(a) Overall distribution of solutions.

(b) Frequency of instances.

Figure 5.10: Results for the trade-off regions analysis for Set D.

The global relationships are clear for *C* (Figure 5.9) and *D* (Figure 5.10). There are no solutions with good values in all objectives and most instances present no solution with good values in three of the objectives. The majority of the solutions are situated where Z_1 and Z_3 alone have good values or where Z_2 and Z_4 alone have good values, as these are the harmonious pairs. Additionally, it can be observed that almost no solutions are present in conflicting areas. For instance, where Z_1 and Z_2 present good values simultaneously. Moreover, solutions in conflicting areas should be close to the chosen threshold.

The set *X* (Figure 5.11) does not contain solutions in r_0 and there are



(a) Overall distribution of solutions.

(b) Frequency of instances.

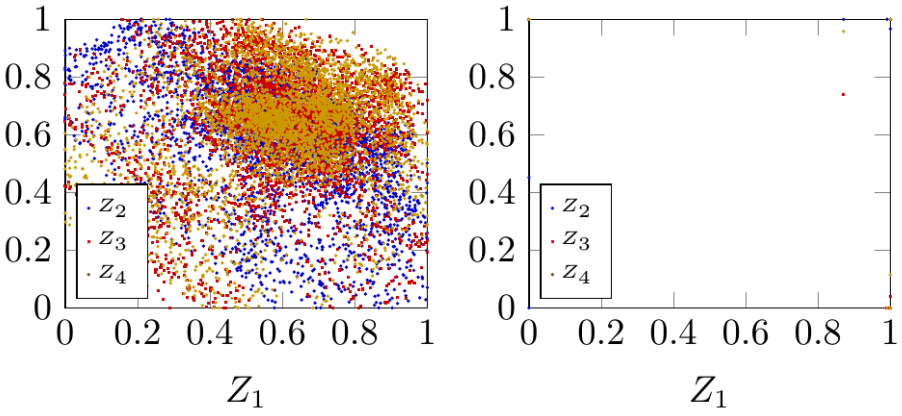
Figure 5.11: Results for the trade-off regions analysis for Set X.

no solutions in regions r_1, r_2, r_4 and r_8, meaning that no high values can be simultaneously found for three or more objectives. Good values can only be found simultaneously for up to two objectives. The map for set X resembles the ones for sets C and D in the sense that it is clear that there are several regions without solutions. Thus, there are trade-offs to present to the decision-maker. This means that the decision-maker has to choose between up to two good objective values to the detriment of the remaining objectives, since all of the regions containing solutions have at most two simultaneously good values.

Step 4 - Multiobjective Scatter Plot Analysis. This analysis was performed for each instance and Figure 5.12 presents the results for all instances of each set combined. It can be seen that although instances in dataset A are completely random, they show similar landscapes with a high concentration of solutions towards the $(1, 1)$ corner. Moreover, no local relationships can be identified, which is expected as the data is completely random.

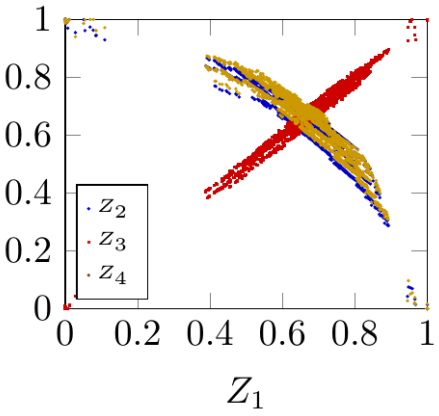
The trade-off regions are clear for sets C and D. There is also a noticeable gap in the objective space when Z_1 is in the range from 0 to 0.5 and when the remaining objectives are in the range from 0 to 0.4, approximately. Moreover, the landscape of the objective space appears to be similar for all instances of each of the sets C and D.

Since the data was uniformly generated, these gaps are unlikely to

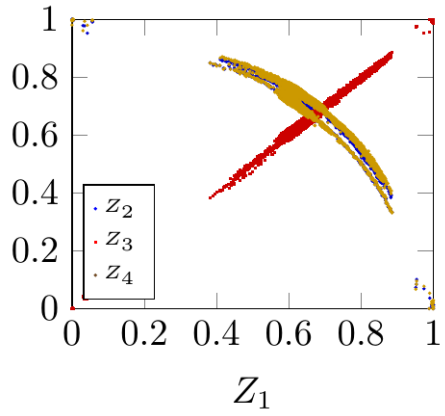


(a) Set A

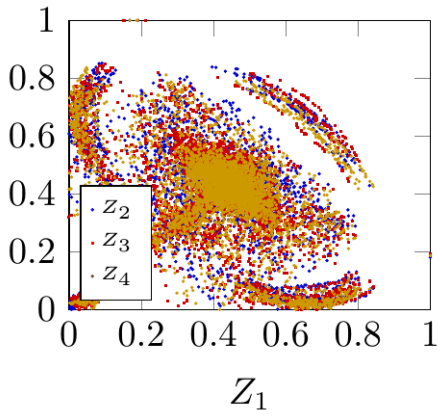
(b) Set B



(c) Set C



(d) Set D



(e) Set X

Figure 5.12: Results for the scatter plot analysis.

arise from the data itself, and could represent limitations in the solution algorithms, indicating that they did not explore the entire front. It is well known that the performance of some MOEAs is limited when the number of objectives is more than three [87].

Set X presents a unique scenario and patterns and gaps can be identified in the objective space. There is a lack of solutions with values within $[0.85, 1]$. This is again likely to be due to limitations of the solution algorithms. However, it can be seen that the size of the gap is small, confirming that instances with strongly conflicting objectives present a bigger challenge for these algorithms. Several local relationships can also be identified. When Z_1 ranges from 0.5 to 0.8, the three remaining objectives simultaneously conflict and harmonise. Knowing that only two simultaneous objectives present high values (from the region map analysis), it can be concluded that whenever Z_1 increases, only one of the other objectives simultaneously increases too.

Discussion

The proposed analysis and visualisation technique can aid in understanding the multiobjective nature of the problem instances considered. Looking at only the correlation coefficients, it could be concluded that: sets A and X do not present interesting multiobjective traits, that set B is inconclusive and that sets C and D present conflicting and harmonious objectives. However, by applying this analysis and visualisation technique, a more comprehensive understanding of these instance sets can be gained.

As fully random instances, dataset A does not present relevant global or local pairwise relationships according to the global pairwise analysis (step 1) and the multiobjective scatter plot analysis (step 4). Additionally, the objective range analysis (step 2) shows that even though there is a large set of non-dominated solutions, these are concentrated in a reasonably small area of the search space. For this dataset, the information from the trade-off region maps can be used to interact with the decision-

maker to identify which regions are of more interest and then use single-objective optimisation algorithms to find solutions in that region. Since there are solutions in all of the regions of the map, any objective vector could provide an adequate solution.

Set *B* presents a completely harmonious case and by analysing the ranges and bearing in mind that the algorithms found just a handful of solutions, it can be concluded that a single-objective algorithm aiming to maximise any of the objectives could provide a good solution.

Sets *C* and *D* present similar scenarios, hence the correlation between weights and coefficients does not impact on the nature of the problem. The entire solution set represents a huge trade-off. The algorithms were not successful in expanding along the front and they mainly explored the region surrounding the intersection of the trade-off. Nonetheless, by perceiving that all instances in these sets have similar landscapes and by knowing the approximate boundaries of each objective (by applying single-objective algorithms to each objective alone), the landscape of solutions could be estimated for other instances in those sets. The search could then be directed to the regions of interest after presenting the expected trade-offs to the decision-maker. However, if it is imperative to use an *a posteriori* approach, the global pairwise analysis and the scatter plots provide sufficient information to make the grouping of harmonious objectives feasible.

Finally, set *X* presents a quite different picture. By only evaluating the global pairwise analysis (step 1) it could be concluded that there is no strong pairwise relationship between objectives. However, the objectives range analysis (step 2) shows that in fact there are non-dominated solutions that vary greatly in quality. This is an indication of the existence of trade-offs (as can be seen by comparing this set with sets *C*

and D). The trade-off region analysis (step 3) showed the existence of overall trade-offs as it is not possible to have solutions with good values in more than two objectives simultaneously. Finally, the multiobjective scatter plot analysis (step4) identified local relationships between objectives and gaps in the objective space, pointing to the existence of local conflicts. Therefore, instances in dataset X exhibit a distinctive multiobjective nature perhaps with interesting options for a decision-maker. A sound possibility to tackle this problem would be to use the region map to identify the regions of interest and then locate those regions in the scatter plot. In case a selected region contains a local conflict, the algorithm proposed by [130] can be used to reach the trade-off front and then expand through it.

5.5.2 Multiobjective Nurse Scheduling Problem

To assess the MONSP, the instances were initially grouped into the three original sets proposed in Section 5.4.2 (IS, ODS and RDS). However, meaningful dissimilarities were not obvious between the sets, because they presented similar fitness landscapes, meaning the instance generation procedure did not create sufficiently different scenarios. However, after evaluating the data, a different grouping of instances was identified – by the case files:

- **Set A:** Instances built using the case files 1, 2, 3, 4 and 6.
- **Set B:** Instances built using the case files 5, 7 and 8.

The results of the analysis for these new sets A and B are presented here, along with the results for the original sets (in the initial steps) to validate the claim regarding their similarities.

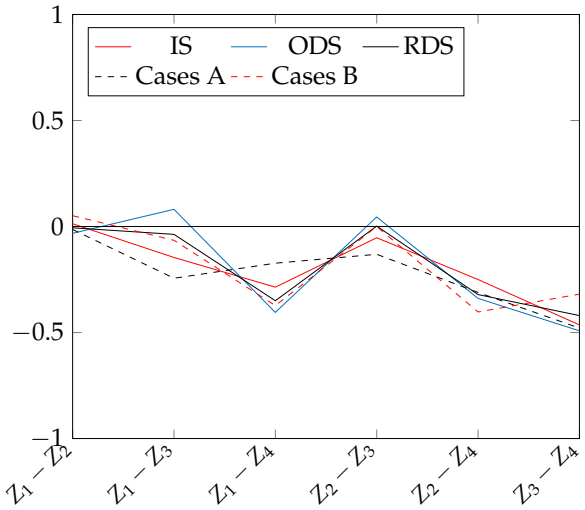
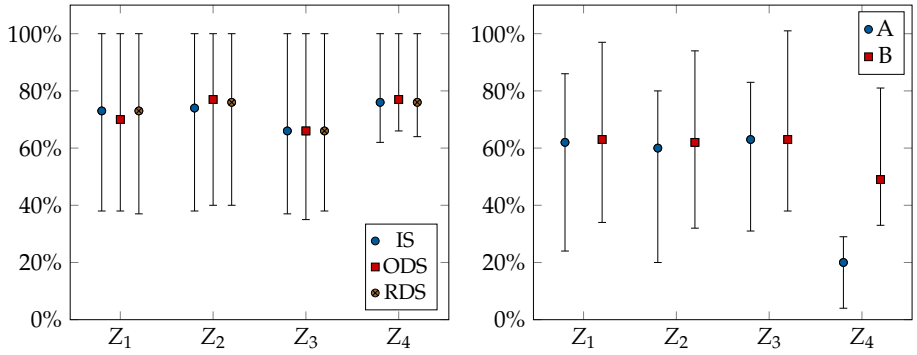


Figure 5.13: Pairwise correlations for each set of instances.

Step 1 - Global Pairwise Relationships Analysis. Figure 5.13 presents the results for the pairwise relationships analysis. Note that all solid lines representing the original sets have similar values on all pair of objectives. Now, it is visible that set B approximately follows the curve of the original sets, however, set A slightly diverges on $(Z_1 - Z_3)$, $(Z_1 - Z_4)$ and $(Z_2 - Z_3)$. Nonetheless, it is noticeable that the differences are minimal and that no set presents strong global pairwise relationships.



(a) Generational sets. **(b)** Distinct cases.

Figure 5.14: Range analysis results for both the generational sets and the distinct cases. The central point shows the average value for the set, the upward lines represent the maximum value found and the downward line represents the minimum value.

Step 2 - Objective Range Analysis. On the objective range analysis

the differences between the original sets and the new sets A and B becomes evident. Figure 5.14a compares the ranges between the original sets. The similarity across all sets in the ranges and average values for all objectives is apparent. However, Figure 5.14b shows that although the average values and ranges are equivalent between sets A and B on Z_1, Z_2 and Z_3, the maximum and minimum values found for set A are roughly 10% lower than for set B. Additionally, for objective Z_4 both ranges and average values are noticeably smaller on A. It can be concluded that grouping the instances based on the original sets did not present any meaningful differences. However, considering the case sets, meaningful differences are apparent. Therefore, from this point on, the analysis considers these sets.

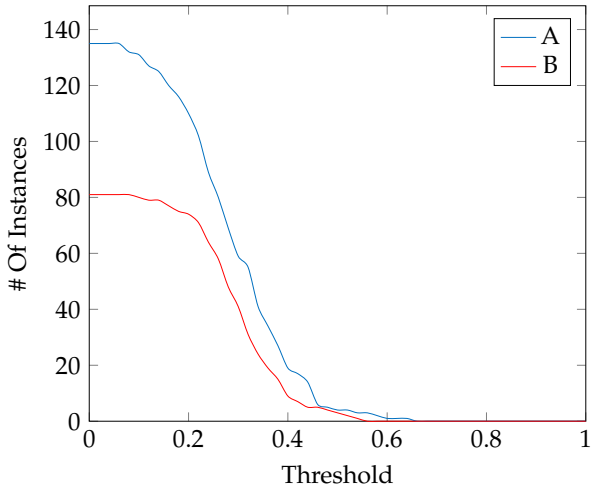
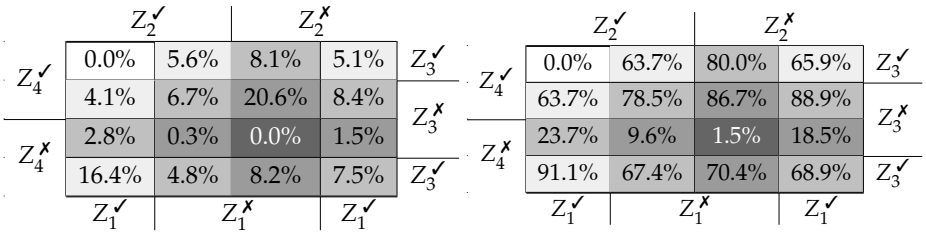


Figure 5.15: Threshold analysis showing the number of scenarios with solutions found in region r_0 when the threshold decreases.

Step 3 - Trade-Off Region Analysis. The threshold analysis is presented in Figure 5.15, where both curves can be seen to be similar and to converge to roughly the same point, showing that in both cases no solution in r_0 can be found with more than 60% of the maximum values. This means that to obtain an objective with more than 60% quality, at least one other objective will present an inferior value.

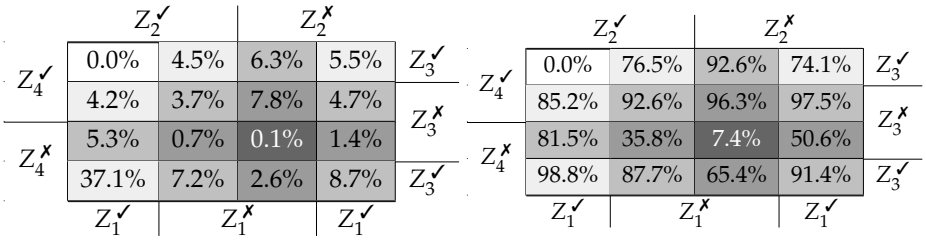


(a) Overall Distribution of Solutions.

(b) Frequency of instances.

Figure 5.16: Region maps for Cases A.

The region maps for sets A and B are shown in Figures 5.16 and 5.17. Sets A and B are similar regarding the regions where solutions can be found, but some differences can be observed. The first difference is that the frequency of solutions is higher in most regions for set B, meaning that the fitness landscapes of instances in B are more alike than the instances in A. In both cases, solutions can be found in all regions of the map, as for the MOMKP independent set (Figure 5.8). The difference, however, is that in the independent MOMKP set, there were 100% frequencies in most regions while here the values greatly vary. This means that the instances share less in common between themselves than in the independent MOMKP case.



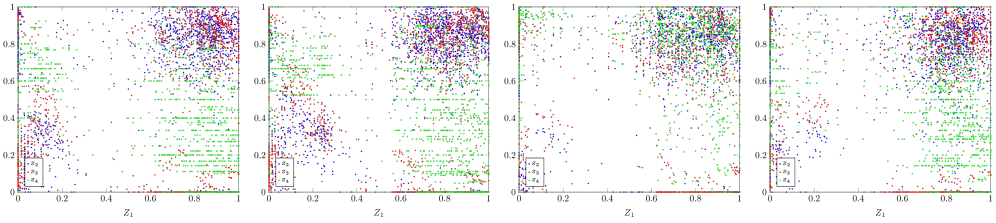
(a) Overall Distribution of Solutions.

(b) Frequency of instances.

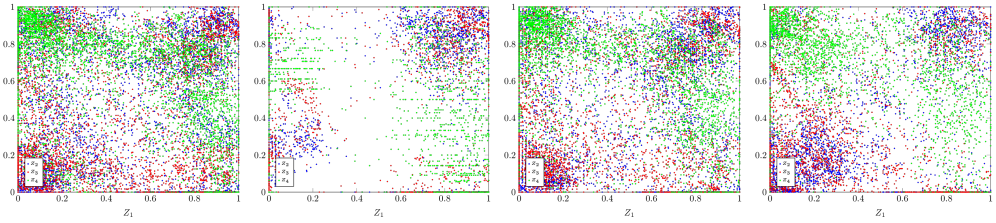
Figure 5.17: Region maps for Cases B.

Step 4 - Multiobjective Scatter Plot Analysis. Finally, Figure 5.18 presents the scatter plot graphs of all individual cases (5.18a–5.18h), for set A (5.18i) and for set B (5.18j). The Pareto front looks similar in each case, with only minor differences, which are highlighted for sets A and

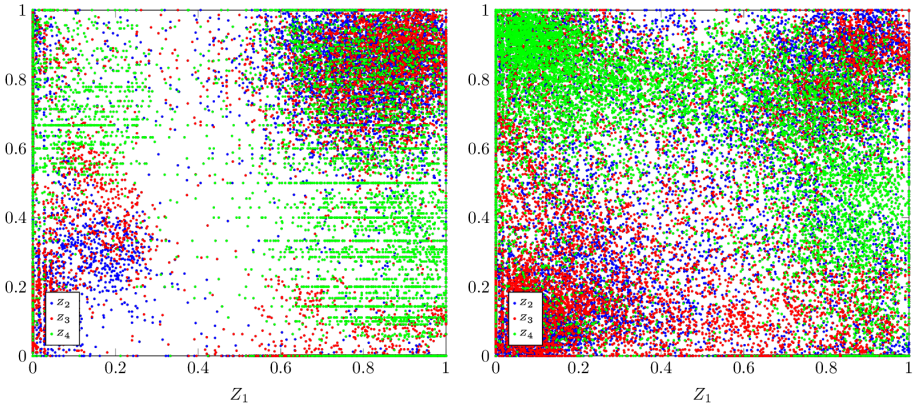
B. A higher concentration of solutions with low Z_3 values when $Z_1 \lessapprox 0.2$ can be observed for set B than for set A, along with a higher concentration of large Z_4 values for the same interval.



(a) Case file 1 **(b)** Case file 2 **(c)** Case file 3 **(d)** Case file 4



(e) Case file 5 **(f)** Case file 6 **(g)** Case file 7 **(h)** Case file 8



(i) Cases files A **(j)** Cases files B

Figure 5.18: Results for the scatter plot analysis.

There are, however, no clear trade-off regions and the overall appearance of both sets is closer to the independent MOMKP set than to a scenario where global and local relationships and trade-offs are evident.

Discussion

We attempted to use the NSPLib, a well known single-objective NSP problem database, to fabricate scenarios for the MONSP. We followed a straightforward process to create different sets of instances, maintaining

either the objectives information or the base problem definition across all instances. Then we used our analysis technique to assess whether these new instances present significant multiobjective traits or not.

The initial two steps of the analysis revealed us that the original sets do not present meaningful differences between themselves, hence by grouping the instances using objective files or base file generate no dissimilarities on fitness landscapes across different instances. The reason for that behaviour is that the data is based on a uniform distribution, hence there are no patterns that could outcome different landscapes. When grouping the instances by the case files (which describes the soft constraints), we managed to note slight dissimilarities between two sets, A and B, namely on Z_4 ranges, which on set A presented much smaller values.

Additionally, the fairly high ranges presented on step two proved that we cannot have near-optimal values in all objectives simultaneously, however the threshold analysis and the region maps shows us that the distribution of solutions is well spread throughout the Pareto front, meaning that solutions can be found in all regions of the solutions space and hence the decision maker can decide about the desired quality of each individual objective. Finally the scatter plot analysis showed us graphs resembling the independent MOMKP set with little differences between sets A and B.

We conclude that due to the uniform distribution of the data of the NSPLib, the original sets did not pose any relevant multiobjective difference between themselves. We managed, however, to find some distinct features due to different constraint set ups, a scenario completely different from the MOMKP instances, where all the differences were obtained by the data itself while the constraints remained unchanged. What the

instances of set B (cases 5, 7 and 8) have in common is that the minimum number of consecutive working shifts is set to two (set A is zero), which outcomes considerably more combinations of solutions with violations on that constraint. With a higher range of the $Z4$, we have a higher number of solutions in the front, hence the scatter plots look denser.

Regarding the multiobjective nature of theses instances, the existing trade-off relates only to the quality of each objective and the fitness landscape is similar among all instances. This means that although the problem is multiobjective, it may not pose any additional difficulties to search algorithms as there are no clear local relationships and non-dominated solutions can easily be found in all regions.

Finally, the uniformity of the data may not reflect real world problems and algorithms tailored for these scenarios may lack the ability to perform well on more complex fitness landscapes.

5.5.3 Multiobjective Vehicle Routing Problem with Time Windows

The MOVRPTW instances proposed by Castro-Gutierrez et al. [46] are now considered using the proposed analysis technique, to assess whether changing time windows and capacities impact on the fitness landscape of the MOVRPTW.

Step 1 - Global Pairwise Relationships Analysis. The global pairwise relationship analysis (Figure 5.19) corroborates the authors' claim that these instances provide interesting multiobjective challenges, as some pairs of objectives have either high harmonious or conflicting relationships. It is, however, noticeable that changing the time windows and δ does not greatly impact on the nature of the pairwise global relation-

ships. The only exception is with $tw0$ which slightly deviates from the other configurations by presenting stronger (conflicting and harmonious) relationships.

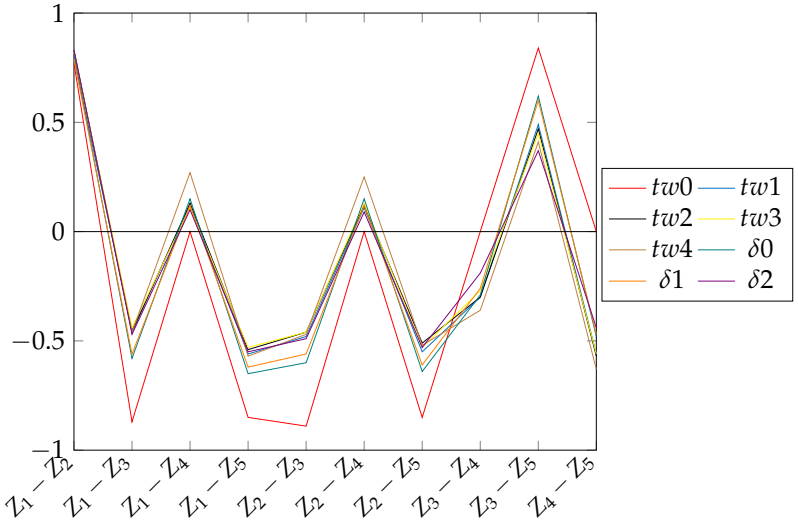
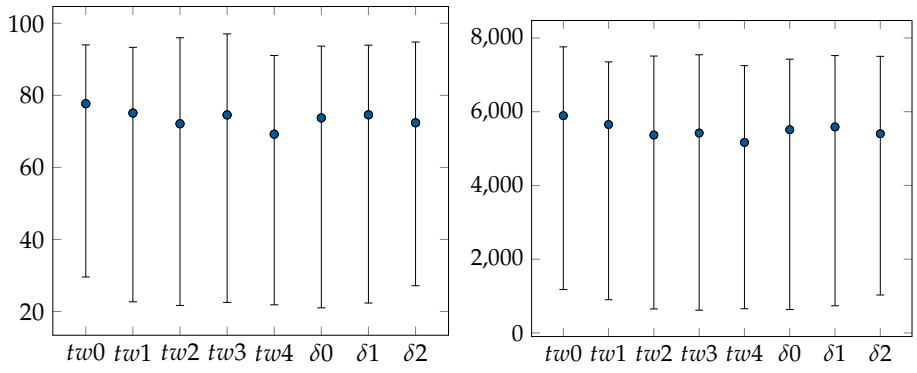


Figure 5.19: Pairwise correlations for each set of instances.

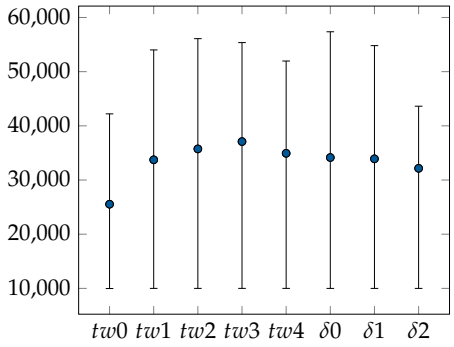
Step 2 - Objective Range Analysis. On the objective range analysis (Figure 5.20), because the values of individual objectives are similar, the actual values are presented here rather than the relative margin, with a chart for each objective. Excluding Z_4 and Z_5 for $tw0$, all objectives present high ranges, again supporting the authors' claim of interesting multiobjective scenarios. The average values found for all setups are similar, but some ranges, namely Z_4 and Z_5 present high variance. On Z_4 and Z_5, it is clear that $tw0$ is smaller than the other scenarios. Z_5 (5.20e) is also the objective with the larger discrepancy between sets.

Step 3 - Trade-Off Regions Analysis. The threshold analysis, presented in Figure 5.21, reveals that there are no instances with solutions in r_0 when the threshold is roughly 60% or higher, meaning that if we require a solution with over 60% quality in at least one criterion, then one or more different criteria will present inferior quality. The steepness of the chart is also accentuated if compared to the MONSP, mean-

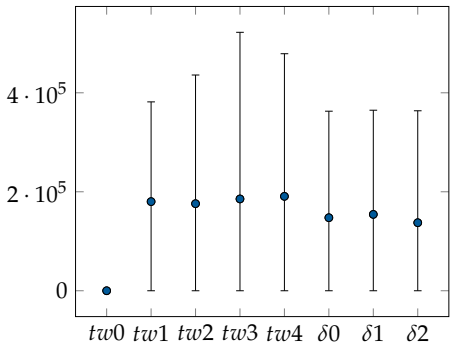


(a) Z_1 – Number of Vehicles

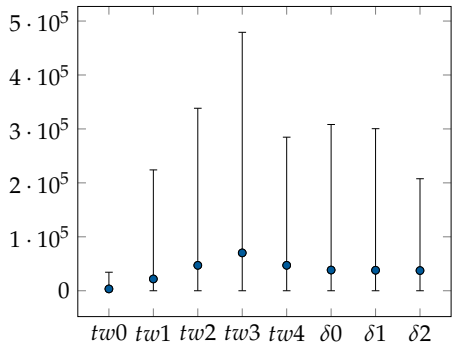
(b) Z_2 – Total Travel Distances



(c) Z_3 – Makespan



(d) Z_4 – Total Waiting Time



(e) Z_5 – Total Delay Time

Figure 5.20: Range analysis results for each objective considering instances with different time windows and delta set-ups. The central point shows the average value for the set, the upward lines represent the maximum value found and the downward line represents the minimum value.

ing that the instances in the MOVRPTW are more alike than the MONSP instances.

Figure 5.22 presents the region map and the frequency map using the threshold found. Many regions without solutions are identified, representing trade-offs for the decision-maker. Some global relationships can

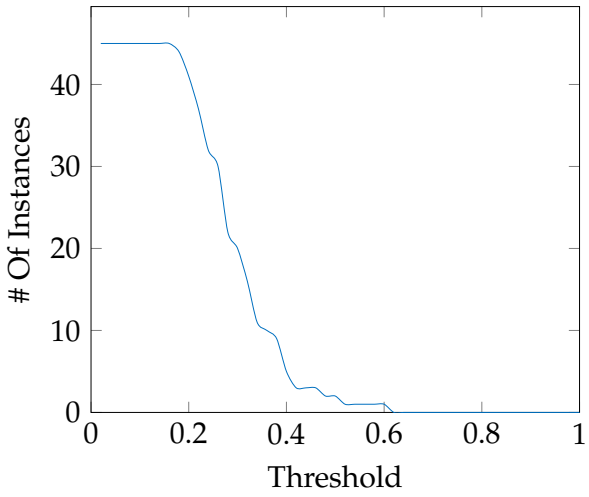
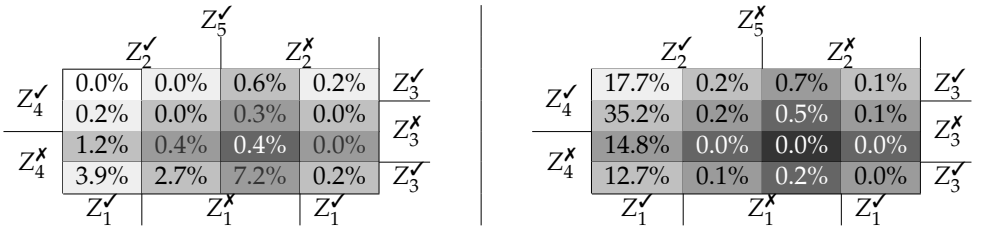
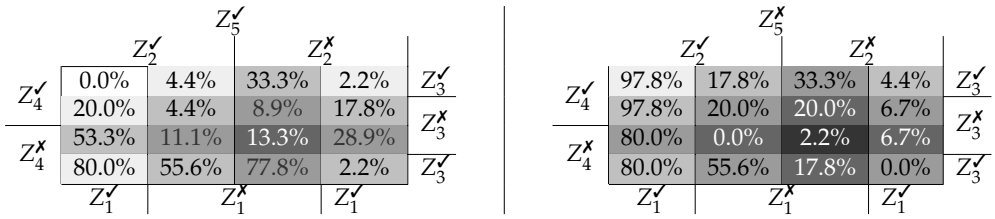


Figure 5.21: Threshold analysis showing the number of scenarios with solutions found in region r_0 when the threshold decreases.

also be observed, for instance, because Z_1 and Z_2 are harmonious, few solutions can be found when $Z_1 = \checkmark$ and $Z_2 = \times$ and vice-versa.



(a) Overall Distribution of Solutions.

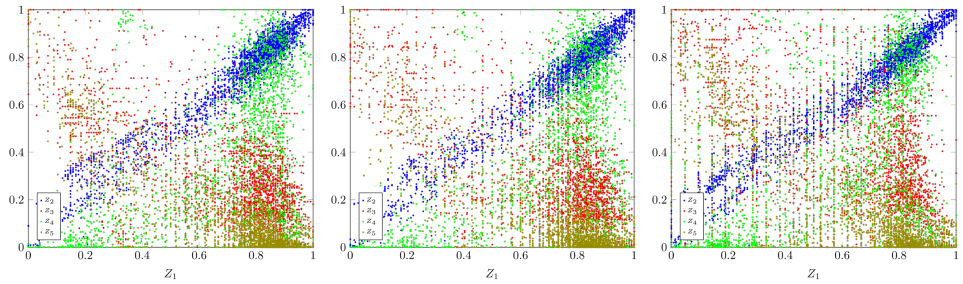


(b) Frequency of instances.

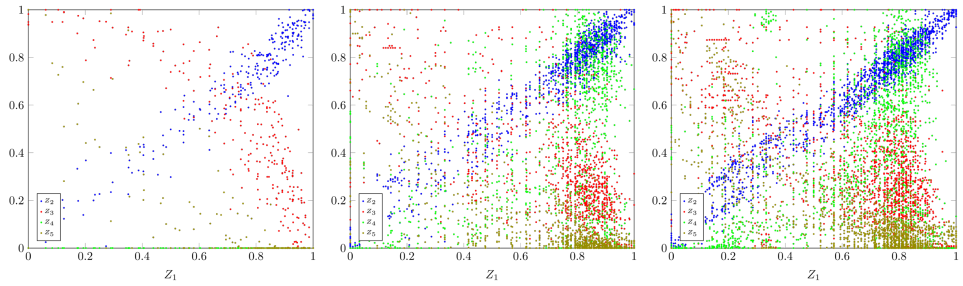
Figure 5.22: Overall region maps.

Interestingly, because Z_3 and Z_5 are fairly globally harmonious, a relationship similar to Z_1 and Z_2 would be expected, however over 17% solutions are in regions where $Z_3 = \checkmark$ and $Z_5 = \times$ or $Z_3 = \times$ and $Z_5 = \checkmark$.

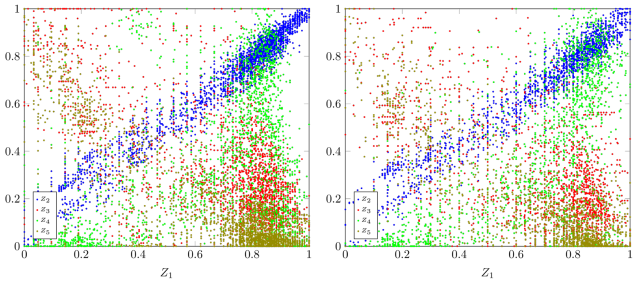
Step 4 - Multiobjective Scatter Plot Analysis. Figure C.30 presents the scatter plots of all sets and the combined overall scatter plot. The first relevant information shown is that the number of solutions for $tw0$ is no-



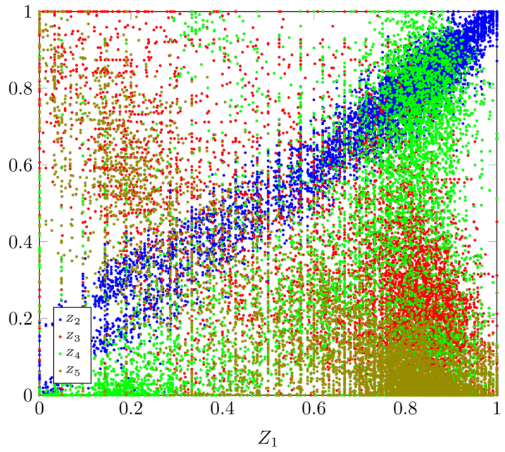
(a) Delta $\delta 0$ **(b)** Delta $\delta 1$ **(c)** Delta $\delta 2$



(d) Time window $tw0$ **(e)** Time window $tw1$ **(f)** Time window $tw2$



(g) Time window $tw3$ **(h)** Time window $tw4$



(i) Overall scatter-plot.

Figure 5.23: Results for the scatter plot analysis.

ticeably smaller than for the other time windows. This happens because each customer in $tw0$ must be served in a single time window consisting of the entire day. Hence, there is no waiting time (because there is

never an early arrival) and there will rarely be delays. Therefore, taking out two objectives drastically reduces the number of non-dominated solutions.

Additionally, the fitness landscapes of all configurations are alike, meaning that varying the delta and time windows does not impose scenarios with distinct multiobjective natures. Finally, the global pairwise harmonious relationship between Z_1 and Z_2 is evident in all graphs.

Discussion

The analysis performed on the benchmark instances of the MOVRPTW [46] corroborates their claim of a challenging multiobjective dataset. The global pairwise relationship analysis clearly showed fairly strong relationships while the range analysis presented high variance in all objectives, meaning that it is not possible to find good solutions in all areas. The region maps presented many areas where no solutions were found, hence the decision-maker must pick the regions of interest. Finally, the scatter plots showed similarities between the setups and where the global relationships stand.

The parametrisation proposed by the authors was observed to have little to no impact on the fitness landscape, meaning that as a benchmark for the problem it fails to provide diverse scenarios. An algorithm that can successfully explore $\delta 0$ will most likely find no difficulties to explore $\delta 1$ or $\delta 2$ due to the similarities of the objective space. Just like with the original sets of the MONSP, the parameters used to generate the scenarios (delta and time windows) were not sufficient to provide instances with distinct fitness landscapes.

When comparing these scenarios with the previous problems, the fitness landscape resembles the MONSP and the independent set of the

MOMKP, with no clear local relationships and a high density of solutions in a region: in this case when $Z_1 \gtrapprox 0.7$. However, the global relationships can be clearly observed, especially the harmonious $Z_1 - Z_2$ and $Z_3 - Z_5$ and the conflicting $Z_1 - Z_3$ and $Z_1 - Z_5$. These scenarios should represent challenging MOVRPTW instances because of the global relationships, high ranges and regions with no solutions.

It is useful to note that other visualisation techniques might provide additional information. In particular, heatmaps [132, 219] might prove to be a useful tool and the inclusion of heatmaps into the proposed analysis technique is a subject that will be investigated in future work. These were not considered here since the tools in hand were already able to provide sufficient information and the added complexity may be detrimental to the clarity, for example the colour scheme of the heatmaps could cause confusion with the colour scheme used in the scatter-plots.

Nonetheless, it is important to emphasise that the proposed technique has not been compared to other visualisation tools and no user studies have been conduced. Future work includes performing further studies and comparisons. Therefore, caution is advised when employing the proposed analysis technique.

5.6 Conclusion

When working with real-world problems, many instances will often exhibit similar structures and characteristics. Even where structures differ between instances, it is common to have only a few different structures. Similarly, for many benchmark problems, the structure will often be similar within a set of instances which were created in the same way, unless measures were actively taken to avoid this. For this reason, under-

standing the structure of some of the instances from a particular problem can give insight into the structure of other instances. Similarly, solution methods which work well for some instances of a problem can often work well for others, where the structure is similar. On the other hand, where the structure is different, an alternative solution method may be preferable, or at least an adaptation to handle that structural change. For example, some problems may be better suited for alternative techniques such as decomposition or weighted sum approaches. However assessing the nature of a MOP is itself, to make this determination, is a non-trivial task.

The technique proposed in the previous chapter was applied to sets of different multiobjective problems:

1. Firstly, five sets of instances of a multiobjective multidimensional knapsack problem were considered and it was shown that the proposed technique can provide accurate information about different problem specifications. A misleading scenario (X) was considered, where correlation values pointed towards independence, but the proposed analysis technique identified composite and local relationships.

2. Secondly, the well known NSPLib [153] was used to generate multiobjective scenarios for the nurse scheduling problem. This analysis technique showed that the generational patterns were not sufficient to create diversity in the fitness landscapes and to generate interesting multiobjective traits. However, the analysis identified the difference between fitness landscapes when the minimum number of consecutive days constraint was set to a more restrictive value.

3. Finally the technique was applied to a well known dataset for a

multiobjective vehicle routing problem with time windows [46]. This analysis technique corroborated the authors' claim of interesting multiobjective scenarios due to the high pairwise correlation values. Nonetheless, the analysis technique also showed that the different parameters used to generate the instances (time windows and δ setups) were not sufficient to provide instances with different fitness landscapes.

In addition to a demonstration of the capabilities of the proposed analysis technique, this chapter gave insights into the generation of MOP benchmark scenarios. It was seen that constraint setup variance, commonly used in the literature, has limited impact on the fitness landscapes, hence relying solely on that may not be sufficient to generate varied scenarios. On the other hand, using data dependency to generate the instances was able to achieve higher diversity and obtain truly unique fitness landscapes, as was shown with the MOMKP.

Chapter 6

Multiobjective Analysis of the Home Healthcare Problem

This chapter defines and analyses the HHC problem presented in Chapter 3 as a multiobjective problem. First, we present the multiobjective definition of the problem. Later, we perform the four-step analysis introduced in Chapter 5.

6.1 The Multiobjective Home Healthcare Optimisation Problem (MOHHC)

To transform the HHC problem to a multiobjective problem we follow a similar process as with the MONSP and the MOVRPTW, shown in Chapter 5, where different objectives and preferences were split into separate objectives. Therefore, the equations 3.1-3.4 in the MIP model presented in Chapter 3 are replaced by the following equation 6.1.

Thus, the objectives of the MOHHC are as follows:

Z_1 denotes the sum of the distances travelled by the workers, including the commuting from their homes to the first assignment, be-

$$\text{Minimize} \quad \vec{f}(\mathbb{X}) = (Z_1, Z_2, Z_3, Z_4, Z_5), \tag{6.1}$$

$$Z_1 = \sum_{k \in K} \sum_{i \in V^S} \sum_{j \in V^N} d_{i,j} x_{i,j}^k, \tag{6.2}$$

$$Z_2 = \sum_{k \in K} \sum_{i \in V^S} \sum_{j \in V^N} p_j^k x_{i,j}^k, \tag{6.3}$$

$$Z_3 = -\sum_{k \in K} \sum_{i \in V^S} \sum_{j \in V^N} \rho'_{k,j} x_{i,j}^k, \tag{6.4}$$

$$Z_4 = -\sum_{k \in K} \sum_{i \in V^S} \sum_{j \in V^N} \rho''_{k,j} x_{i,j}^k, \tag{6.5}$$

$$Z_5 = -\sum_{k \in K} \sum_{i \in V^S} \sum_{j \in V^N} \rho'''_{k,j} x_{i,j}^k \tag{6.6}$$

tween each assignment, and from the final assignment back to their homes;

Z_2 denotes the sum of incurred assignment monetary costs;

Z_3 denotes the sum of skills preferences;

Z_4 denotes the sum of preferences by assigning preferred workers; and

Z_5 denotes the sum of preferences by assigning workers to visits in their preferred area.

The objectives Z_3, Z_4, and Z_5 (equations 6.4, 6.5 and 6.6 respectively) are negative because the MOHHC is defined as a minimisation optimisation problem but the preferences need to be maximised. Also, in this study, we ignore the time and area availability violations (third tier of the HHC objective function) because early experiments showed that restricting the objective space substantially reduced the quality of the approximation (of the Pareto Front) sets obtained..

6.2 Experimental Configuration

We performed the experiments on the datasets described in Chapter 3. The evolutionary algorithms MOEA/D and NSGA-II were employed with uniform crossover, two-individual tournament selection and a population of 500 individuals for datasets A, B and C; 250 individuals for datasets D and E; and 100 individuals for dataset F. In order to facilitate the exploration of the feasible objective space, the indirect representation of individuals proposed by Algethami et al. [4] was applied.

6.2.1 Indirect Individual Encoding

The indirect individual scheme works as follows: for each possible individual assignment (a visit requiring multiple workers is considered as multiple possible individual assignments), a list of eligible workers is created. An eligible worker is any worker fully qualified to perform a given visit. The chromosome of an individual is an integer array such that each gene corresponds to an individual assignment. The content of a gene is an integer value ranging from one to the number of eligible candidates for its corresponding assignment. Also, to eliminate time-conflicts (the assignment of a worker to two simultaneous visits), if the assignment of the i-th gene would incur a time-conflict with any gene from 1 to $i-1$, then the next non-conflicting worker in the eligible workers list is selected instead. If no workers are available, then the assignment is left empty. Therefore, this scheme is called an indirect encoding because the content of a gene does not correspond directly to a worker, but to the index of a list that contains the worker.

6.2.2 Obtaining the Approximation Set

Because modern multiobjective algorithms struggle to find good approximation sets in combinatorial problems with many objectives (more than three) [87], and because the objective of this study was not to develop new multiobjective algorithmic components, we resort to a tailored procedure to obtain improved results. Giagkiozis and Fleming [88] state that most multiobjective algorithms can be classified as either Pareto-based or decomposition-based. Therefore, this study employs one of the most widely employed algorithm in each cathegory: NSGA-II as the Pareto-based algorithm and MOEA/D as the decomposition-based. Thus, for each problem instance the approximation set was obtained as follows:

1. run both the NSGA-II and MOEA/D for one million objective evaluations on each possible bi-objective vector (Z_1, Z_2), (Z_1, Z_3), ... (Z_4, Z_5);

2. run both the NSGA-II and MOEA/D for one million objective evaluations on each possible three-objective vector (Z_1, Z_2, Z_3), (Z_1, Z_2, Z_4), ... (Z_3, Z_4, Z_5);

3. run both the NSGA-II and MOEA/D for one million objective evaluations on each possible four-objective vector (Z_1, Z_2, Z_3, Z_4), (Z_1, Z_2, Z_3, Z_5), ... (Z_2, Z_3, Z_4, Z_5);

4. create an archive composed of the non-dominated solutions found in the previous three steps;

5. generate an initial population for the main algorithms where half of the elements are randomly generated and the other half are randomly drawn from the archive built in the previous step;

6. run both the NSGA-II and MOEA/D four times each, for two million objective evaluations, using the initial population generated in the previous step and the five-objective vector; and

7. compile an approximation set with all non-dominated solutions found in all steps.

6.3 Four-Step Analysis

We obtained an approximation set for each HHC instance presented in Chapter 3. Datasets A, B, D and E presented a uniform number of solutions among their instances. with dataset A averaging roughly 1200 solutions in each set, 1800 solutions for dataset B, 2500 solutions for dataset D and 3600 solutions for dataset E. Dataset C showed a different behaviour, where some instances presented small approximation sets while others presented large approximation sets with up to 1800 solutions. Dataset F also showed a different behaviour with some instances presenting over 1800 solutions on the smaller instances and only four solutions on the largest instance.

Regarding dataset C, it was explained in Chapter 3 that some instances of this set are much larger than others in terms of a number of visits. Hence, on the bigger instances, the algorithms managed to find a large number of non-dominated solutions, while on the smallest scenarios the number of non-dominated solutions is small.

The variance on the number of solutions in dataset F happens for a different reason. Because of the large size of the instances, even after eight hours of processing, the quality of the approximation set obtained was not good enough. It is not clear if the algorithms were not capable of providing good results for a problem of this size, or if the time limit

of eight hours per instance was insufficient. Therefore, for the biggest scenarios only a few non-dominated solutions were found.

6.3.1 Step 1 - Global Pairwise Relationships Analysis

The global pairwise relationship analysis (Figure 6.1) performed on the approximation sets shows that no dataset present strong global relationships with dataset D presenting the overall strongest harmonious relationships on Z_1–Z_5 and Z_3–Z_5 and datasets C and D presenting the strongest conflicting relationships on Z_2–Z_3, Z_2–Z_4 and Z_2–Z_5. Also, note that dataset C does not have distances information, hence all correlations with Z_1 are zero.

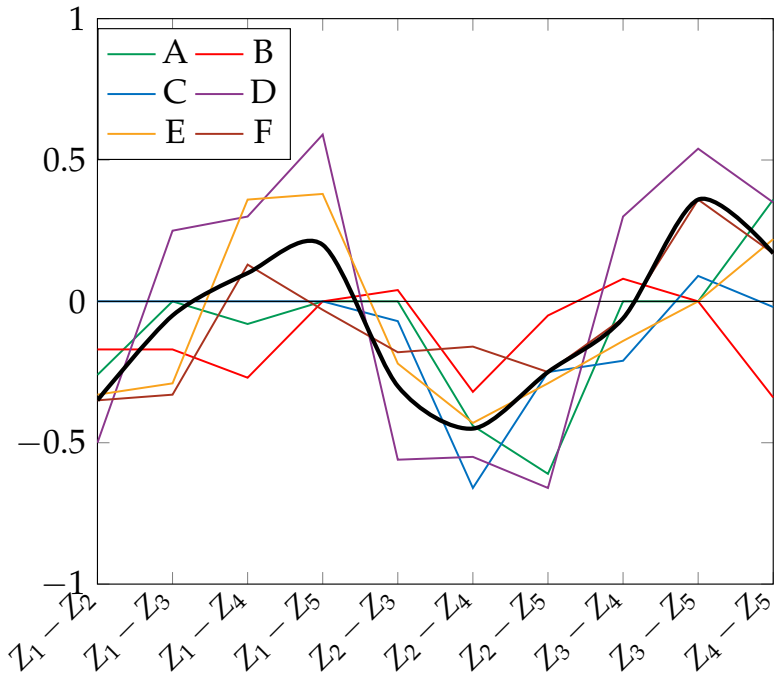


Figure 6.1: Pairwise correlations for each set of instances. The tick black line highligths a trend.

Additionally, the objective Z_2 has an overall conflicting relationship with all other objectives in all datasets, meaning that reducing costs implicates a deterioration in at least one other objective. Moreover, the thick black line in Figure 6.1 highlights a common trend between the

majority of the datasets indicating that even though the data was gathered from different home healthcare companies, the underlying problem has similar traits.

6.3.2 Step 2 - Objective Range Analysis

Figure 6.2 presents the results of the objective range analysis on the approximation sets obtained. Because the datasets have different sizes, instead of showing the raw values we show the percentual margin of values where 100% represents the highest value for an objective in the corresponding instance. Note that Z_1 is zero for dataset C because that dataset does not contain distances information and Z_3 is zero for dataset A because that dataset does not possess skill preferences.

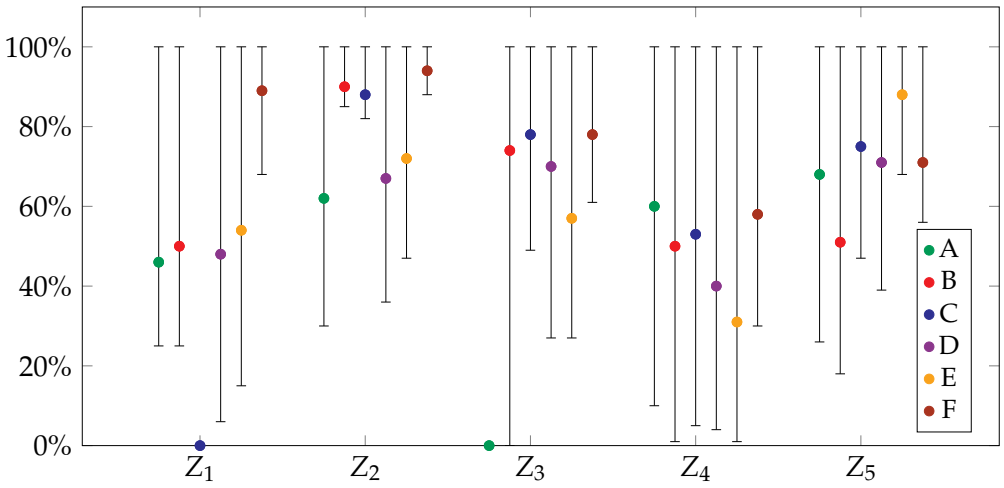


Figure 6.2: Range analysis results for both the generational sets and the distinct cases. The central point shows the average value for the set, the upward lines represent the maximum value found and the downward line represents the minimum value.

The ranges vary according to objective and dataset. Noticeably, the objective Z_2 (the most conflicting objective) does not present high ranges on datasets B, C and F. Hence, for those datasets, a small increase of incurred costs (Z_2) may allow substantial gains on the remaining objec-

tives. On datasets A, D and E, the range of Z_2 is larger, indicating that the monetary costs of improving the remaining objectives are proportionally larger in comparison to the other datasets. Also, all the preferences (objectives Z_3, Z_4 and Z_5) present high ranges, particularly the preferred worker objective (Z_4).

Dataset F presents the overall smallest ranges (except on Z_5). However, because of the large size of that dataset, even a small percentual decrease of an objective may be significant to the decision maker, as it might translate to thousands of pounds sterling, hundreds of miles or attending the preferences of hundreds of people.

More importantly: no two sets present similar ranges on all objectives, meaning that although the datasets are defined as the same problem, they have their own characteristics, making each dataset unique. This happens because of the specific needs that each company have and how they input the data into the ERP system.

6.3.3 Step 3 - Trade-Off Regions Analysis

The threshold analysis, shown in Figure 6.3, reveals that the approximation sets obtained for datasets A, B, and C have no solutions in r_0 when the threshold reaches roughly 40%. Datasets D and E present different results with thresholds of roughly 50% and 60%. The dataset F present an interesting scenario where the number of instances slowly decreases as the threshold increases. This happens because on the largest instances the algorithms failed to explore the objective space and reach solutions closer to the optimal Pareto front, hence the quality of the results obtained is not high enough for a precise analysis.

Figures 6.4–6.9 present the region maps with the overall distribution of solutions and the frequency of instances for all datasets.

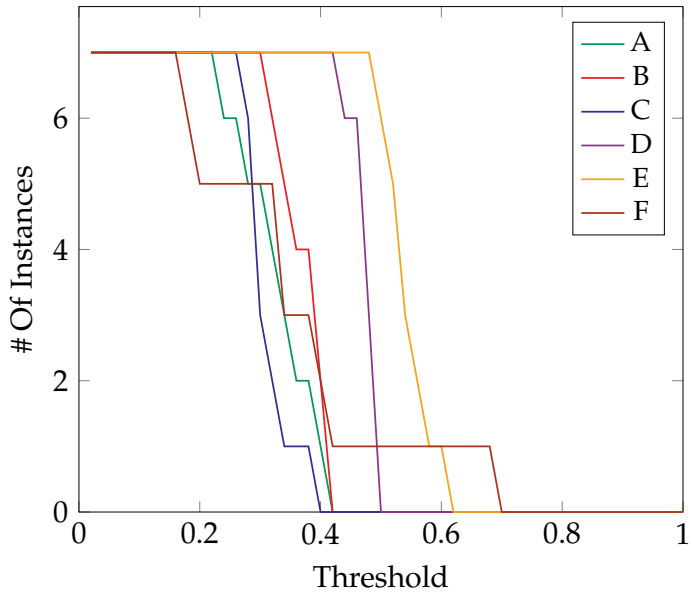
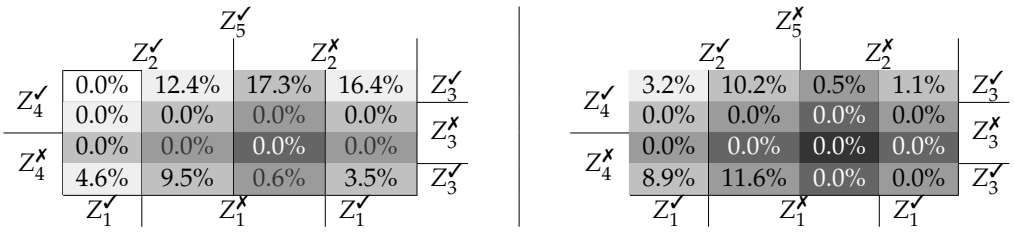
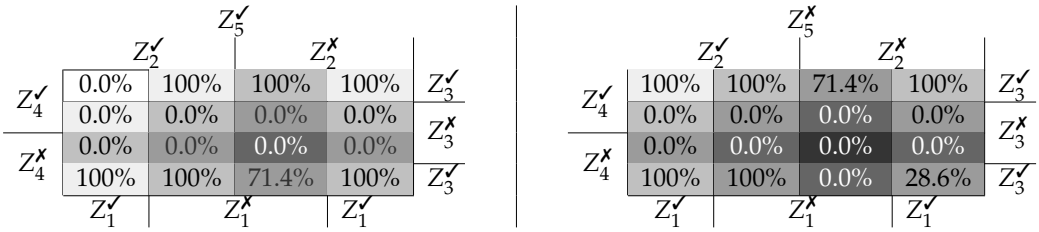


Figure 6.3: Threshold analysis showing the number of scenarios with solutions found in region r_0 when the threshold decreases.



(a) Overall Distribution of Solutions.

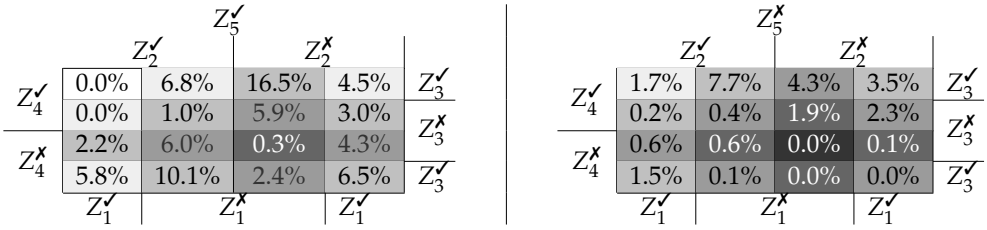


(b) Frequency of instances.

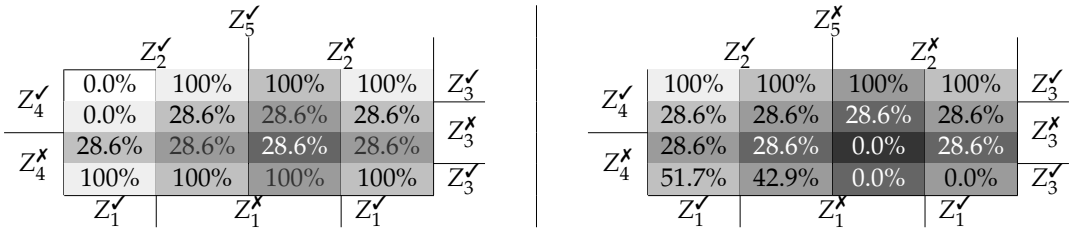
Figure 6.4: Region maps for instance set A.

Figure 6.4 presents the region maps for dataset A. The uniformity among the instances is clear in the frequency of instances map, as it shows that the majority of values are either 0% or 100%, meaning that most instances have solutions in the same regions. Regarding the objectives, there are no solutions when $Z_3 = \times$ because that objective is not present in dataset A. Hence, solutions can be found in all remaining regions. The dataset A resembles the independent MMKP set shown in

Chapter 5 which also presented solutions in the majority of the regions.



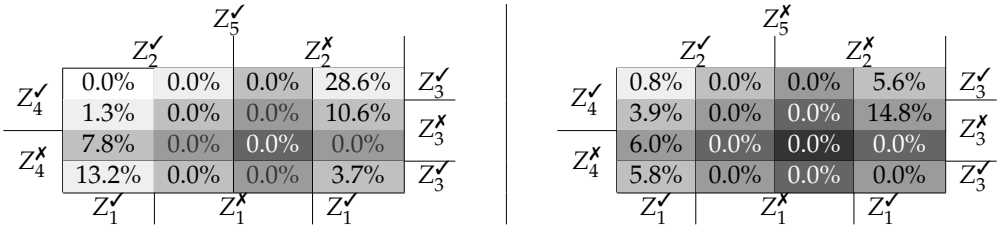
(a) Overall Distribution of Solutions.



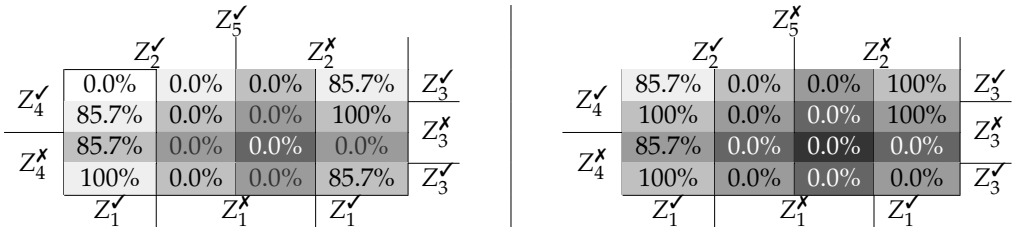
(b) Frequency of instances.

Figure 6.5: Region maps for instance set B.

In Figure 6.5 we have the region maps for dataset B. As with dataset A, the solutions are well spread throughout the objective space. However, this dataset is not as uniform as the previous because not all instances possess solutions in the same regions. For example, only two instances present solutions when $Z_3 = \times$.



(a) Overall Distribution of Solutions.



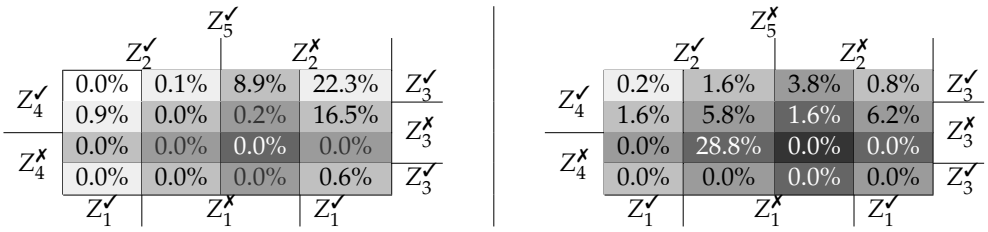
(b) Frequency of instances.

Figure 6.6: Region maps for instance set C.

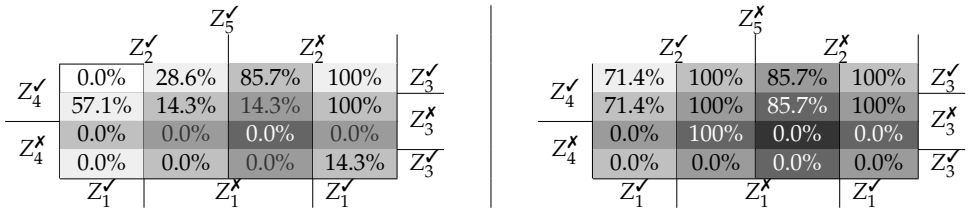
Figure 6.6 displays the region maps for dataset C. Because that dataset does not have distances information, there is no solutions when $Z_1 = \times$.

Considering the regions where $Z_1 = \checkmark$, it is clear that solutions can be found in most regions. Also, this dataset is uniform as the majority of the instances possess solutions in the same regions. Therefore dataset C is similar to datasets A and B regarding not only the threshold found but also the uniformity and the distribution of solutions of the independent MOMKP set.

Note that even though it is possible to find solutions in most regions of datasets A, B and C, the threshold selected is low, meaning that a ✓ objective-value may still be improved by up to 60% on individual objectives.



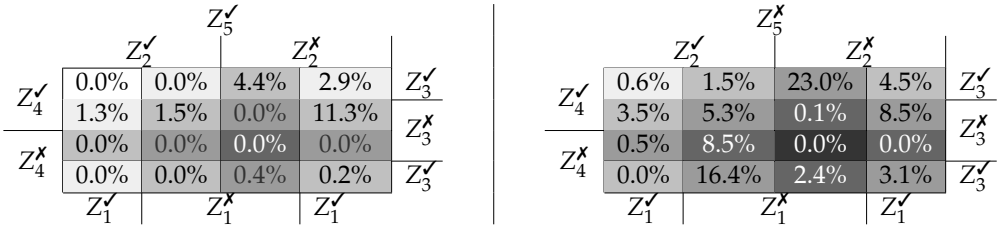
(a) Overall Distribution of Solutions.



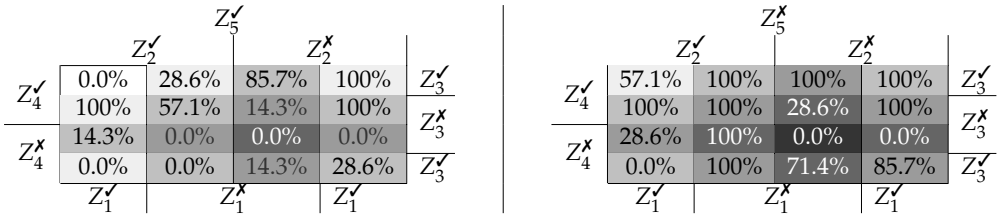
(b) Frequency of instances.

Figure 6.7: Region maps for instance set D.

The region maps of dataset D, provided in Figure 6.7, show a different scenario. Some regions have no solutions, revealing trade-offs. For instance, a decision-maker has to choose between Z_2, Z_3 and Z_5 as there are no solutions in regions where these three objectives are ✓ on the majority of the instances. Additionally, only a few regions present a substantial number of solutions ($> 1\%$) when $Z_5 = \checkmark$, hinting that a small increase of the threshold potentially will reduce the decision maker's options.



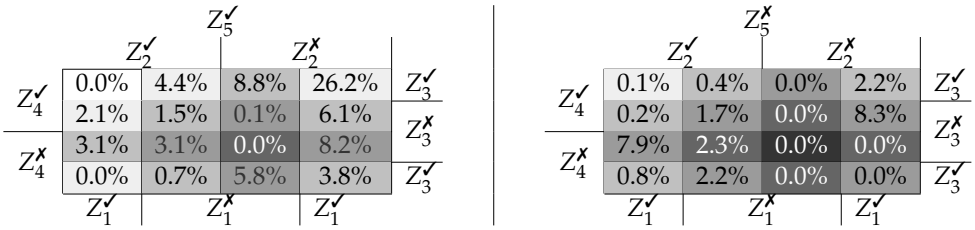
(a) Overall Distribution of Solutions.



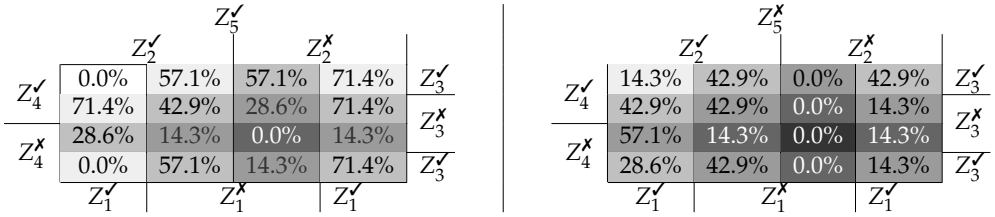
(b) Frequency of instances.

Figure 6.8: Region maps for instance set E.

Figure 6.8 displays the region maps for dataset E. It can be observed that dataset E presents a similar scenario to dataset D, as some regions show zero solutions across all instances. For example, like in that other dataset, the decision-maker has to choose between Z_2, Z_3 and Z_5. Also, this dataset presented the highest threshold (except for dataset F), meaning that solutions in r_0 can be found with an overall objective quality of 60%.



(a) Overall Distribution of Solutions.



(b) Frequency of instances.

Figure 6.9: Region maps for instance set F.

The results for dataset F are displayed on Figure 6.9. The frequency of

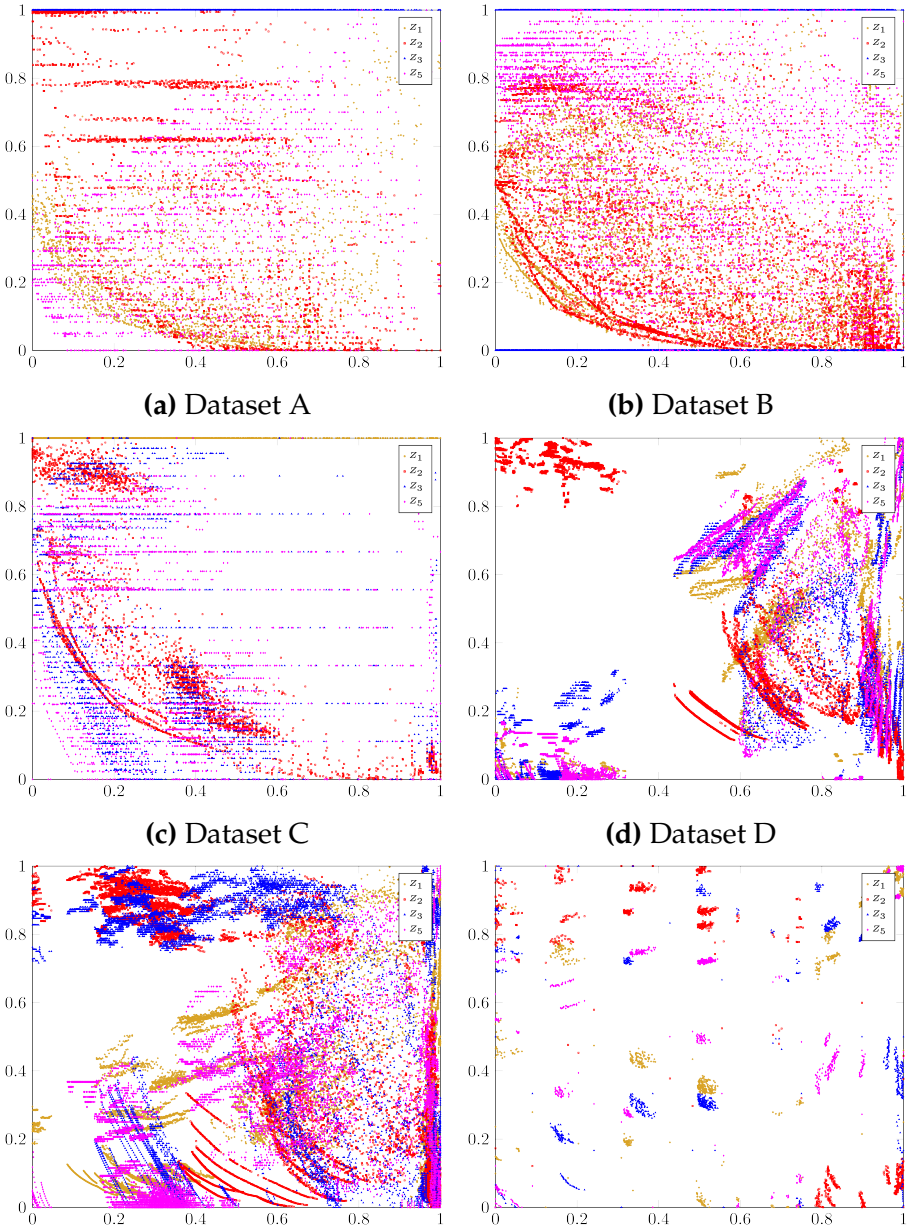
instances' region maps shows the non-uniformity of the approximation sets obtained, where the maximum value found is 71.4% in four regions, meaning that there is little consistency throughout the sets. We know that this happened because the algorithms could not find an approximation set closer to the optimal Pareto front.

6.3.4 Step 4 - Multiobjective Scatter Plot Analysis

The scatter plot analysis, presented in Figure 6.10, shows the combined scatter plots for all approximation sets of all datasets. We choose Z_4 as the x axis because it presented a good visualisation for all datasets, however, Appendix C presents scatter plots with different choices of objectives for the x axis along with individual scatter plots for all instances.

In the previous steps of the analysis, we identified that the approximation sets of datasets A, B and C are similar to each other and to the MOMKP independent set. Figures 6.10a, 6.10b and 6.10c further validate that claim. Clearly, these datasets present a strong resemblance with the MOMKP independent set as solutions can be found throughout the objective space, except near the point $(0, 0)$ (note that the MOHHC is defined as a minimisation problem and the MOMKP is defined as a maximisation problem, hence in that problem solutions cannot be found near the point $(1, 1)$).

The approximation sets of datasets D, E and F (Figures 6.10d, 6.10e and 6.10f) present a different scenario where many local relationships can be observed and a few gaps in the objective space, where no objective can be found. It could be argued that this happened because the algorithms could not explore the objective space, thus exploring instead sub-optimal regions that originated those unique scatter-plots. In fact, if both algorithms present the same systematic bias across all instances of



(a) Dataset A **(b)** Dataset B

(c) Dataset C **(d)** Dataset D

(e) Dataset E **(f)** Dataset F

Figure 6.10: Scatter plots of each combined dataset, showing the relationship between objective Z_4 and each of the other four objectives. In each case, the x axis shows the value of objective Z_4 and the y axis shows the values of each of the other objectives in different colours.

the same dataset, the fitness landscapes might be artefacts of the algorithms and not representations of the Pareto front. However, it is less likely that the observed fitness landscapes are the result of systematic bias because two distinct solving algorithms are employed with a differ-

ent number of objectives being considered. Also, the results from step 3 show us the consistency of the fitness landscape of the instances in these datasets. Additionally, datasets D and E presented similar fitness landscapes among all instances of the datasets (Appendix C). Therefore, if the several local relationships were caused by poor algorithmic performance, there probably would be different fitness landscapes across the instances, just like it is observable on dataset F.

6.4 Conclusion

The multiobjective analysis presented in this chapter showed that the HHC clearly has distinctive multiobjective traits. Datasets A, B and C presented characteristics similar to the MOMKP independent set as shown in steps 3 and 4. However, unlike that independent set, the ranges shown in step 2 are large, meaning that although solutions can be found in all regions, the threshold for that is low and if decision-makers require higher quality objective-values in one or more objectives, they must choose between four, three or two objectives, depending on the quality desired.

Datasets D and E introduced a different scenario. Dataset D possesses the overall highest global pairwise relationship values. Also, both datasets D and E presented some of the largest ranges in step 2. Step 3 identified trade-offs between several objectives and also showed the consistency between the instances in those scenarios as many regions presented either 100% or 0% values, meaning that either all instances had solutions in a given region or none of them had. Finally, many local relationships and gaps in the objective space were highlighted in step 4. Although the scatter plots of these datasets be considerably different

from the scatter plot of set X of the MOMKP, these HHC datasets present several similarities with that special set. The ranges are high, the fitness landscape is unique and trade-offs involving more than two objectives can be found.

Regarding dataset F, it became clear during the initial experiments that the computational effort to process multiobjective algorithms in a problem of that size is prohibitive. The reason for the poor performance is that the computational cost to evaluate the objective function is too high, as the evaluation of a single solution can take up to four seconds on F7 because of the large size of the data structures, the complexity of the calculations due to the large number of constraints, the necessity of fully evaluating each solution. It runs out of the scope of this research to scrutinise the literature in search of solving methods that can succesfully tackle problems of the size of instance F7, however the literature presents sound ideas to tackle it, such as the ParEGO framework [129], which is a multiobjective algorithm specifically designed for scenarios where the solution evaluation is expensive. In only the first two instances we managed to obtain some results and in those two instances, the similarities are evident (Appendix C). However it might be the case that this was due to the algorithms not being able to properly explore the objective space, hence the results are inconclusive.

One interesting finding is the consistency of fitness landscapes between instances of the same dataset. This happens because part of the staff is the same across all instances and part of the visits are recurring (elderly people might need a bath every day, or to take a medicine at a specified time every day, etc.), hence an identical visit might happen in several of the instances. Also, each dataset arises from a different home healthcare provider, and their staff possibly use the ERP system in a con-

sistent way.

Chapter 7

A Variable Neighbourhood Search for the Home Healthcare Problem

Due to the large size of some of the HHC datasets considered in this thesis, the use of exact methods to solve the problem is not applicable [137], hence, we take a heuristic approach. The literature shows that Variable Neighbourhood Search (VNS) algorithms have been successfully applied to both scheduling and routing problems. Among the works found, we highlight that Burke et al. [37] and Della-Croce and Salassa [64] employed VNS algorithms to tackle the nurse scheduling problem. More recently, Constantino et al. [55] proposed a VNS with two domain-specific neighbourhoods to solve the preference balanced nurse scheduling problem. Roshanaei et al. [193] proposed a VNS to tackle the job shop scheduling problem and Davidovic et al. [59] presented a VNS for the multiprocessor scheduling problem with communication delay. Bräysy [31] successfully tackled a vehicle routing problem with time windows using a VNS and Polacek et al. [178] used the algorithm to tackle the

multi-depot version of the vehicle routing problem with time windows, a problem that shares many features with HHC problems.

This chapter presents a VNS for the HHC problem that employs three domain-specific local search neighbourhoods to provide solutions. The first sorts the workers by priority, estimating the best workers for each task and restricting the search space to the highest priority workers. The second attempts to eliminate time and area violations from a current solution. A third local-search uses a cost estimation instead of a direct evaluation of the objective-values, hence drastically increasing the speed of the search procedure, and because of that, this local search is capable of producing improved results. In order to maximise the performance of the proposed local search, we propose a variant of the inner Variable Neighbourhood Descent to accommodate heuristics with different computational performances. Additionally, we propose two constructive greedy heuristics to generate fast initial solutions. We compare the results of the proposed heuristics against two other algorithms found in the literature, a genetic algorithm and a hybrid heuristic. We show that the proposed VNS outperforms the other assessed algorithms and that it is, to the best of our knowledge, the best-known algorithm to tackle these instances of the HHC problem.

7.1 Variable Neighbourhood Search

Variable neighbourhood search is a technique widely applied in the literature partly because of its flexibility: one can design distinct domain specific local search procedures and attach them to the algorithm. Therefore, because the multiobjective analysis on the HHC datasets showed us that each dataset has its own characteristics and challenges, the VNS

provides an adequate framework to incorporate tailored local search operators.

A Variable Neighbourhood Search consisting of two specialised local searches is proposed along with two constructive heuristics to solve the instances of the HHC studied in this thesis.

7.1.1 Shaking Neighbourhoods

Following we list the shaking neighbourhoods designed for these HHC problem scenarios. Note that some of these neighbourhoods might generate a solution with time conflicts.

Random Flip. Randomly picks a visit and changes the assigned worker to a random different worker that is skilled and has a valid contract to perform that visit.

Area Availability Flip. Randomly picks a visit where the area availability constraint is violated and attempts to fix the violation by picking any other worker that is skilled, has a valid contract to perform that visit and is available to work in that area.

Time Availability Flip. Randomly picks a visit where the time availability constraint is violated and attempts to fix the violation by picking any other worker that is skilled, has a valid contract to perform that visit and is available to work at the visit time.

Preferred Worker Flip. Randomly picks a visit where the assigned worker is not the most preferred (according to the patients' preferences) and replaces the worker by the most preferred worker for the visit (as long as that worker is skilled and has a valid contract for the visit).

Preferred Skills Flip. As *Preferred Worker Flip*, but uses the preferred skill value.

Preferred Area Flip. As *Preferred Worker Flip*, but uses the preferred

area value.

Priority-Based Flip. Uses a priority list (defined in Section 7.1.3) that estimates valid assignments' penalty costs. Selects a random visit for which the currently assigned worker is not the top one on the priority list and assigns the top priority worker instead.

7.1.2 Randomised Hill Climbing (RHC)

Hill climbing is a well-known local search technique that consists of an extensive exploration of the solution space. We include this local search as a benchmark for our specialised techniques. Algorithm 8 presents the pseudo-code: for each visit following a random order (line 1), the RHC iterates through all *feasible workers* – skilled and possessing a valid contract – (lines 2 and 3), assigning each one of them at a time and evaluating the solution (line 4), keeping the best assignment (line 5).

Algorithm 8 Randomised Hill Climbing

Require: Initial solution $\mathbb{X}$
1: **for each assignment** (v, k) **following a random order do**
2: **for each worker** $k' \in K$ **do**
3: **if** (v, k') is feasible **then**
4: **if** $f(\mathbb{X} \setminus (v, k) \cup (v, k')) < f(\mathbb{X})$ **then**
5: $\mathbb{X} \leftarrow \mathbb{X} \setminus (v, k) \cup (v, k')$
6: **end if**
7: **end if**
8: **end for**
9: **end for**
10: **return** $\mathbb{X}$

7.1.3 Priority-Based Search

The literature presents many works where local search neighbourhoods are designed to tackle specific constraints of the problem [31, 55, 178]. However, the multiobjective analysis of the HHC datasets studied in

this thesis showed that the datasets present distinct fitness landscapes and objective relationships. Thus, the constraints are not equally challenging in all datasets. By having local search neighbourhoods designed to tackle specific constraints, we risk having mechanisms that are only useful on a subset of the datasets. The problem with that approach is that we end up with different versions of the algorithm that present distinct performances on different datasets, making it difficult for the academic community to choose the right configuration to tackle related problems, hence, decreasing the applicability of the technique because they would also need to know the intrinsic multiobjective characteristics of the tackled problem (or experiment with all neighbourhoods). Therefore, we present two local search neighbourhoods that exploit the objective function instead, and only one that is based on constraints.

As shown later, the RHC presents good results on smaller datasets. However, it gets prohibitively slow on larger datasets because of the high number of objective evaluations. The problem is exacerbated as delta evaluation is not employed due to the large number of constraints in the problem and the complexity of maintaining the source code, thus, when an objective evaluation is required the entire solution is recalculated. The Priority-Based Search (PBS) attempts to improve that issue and considers that the objective function presented in Chapter 3 stipulates that the routing aspect of the problem (i.e. the minimisation of distances) is of less importance than assigning all tasks, satisfying the preferences and avoiding constraint violations. Also, given any assignment, is it possible to estimate the impact of that assignment in the objective-function: suffice to calculate time and area availability violations, preferences attendance, staff cost and apply their respective weights (λ).

In this heuristic, we assume that the likelihood of an assignment to

be present in a good solution is inversely proportional to the costs in the matrix. Hence, workers that present the lowest costs (minimisation problem) have a higher probability of appearing in better solutions, given that time conflicts are properly avoided.

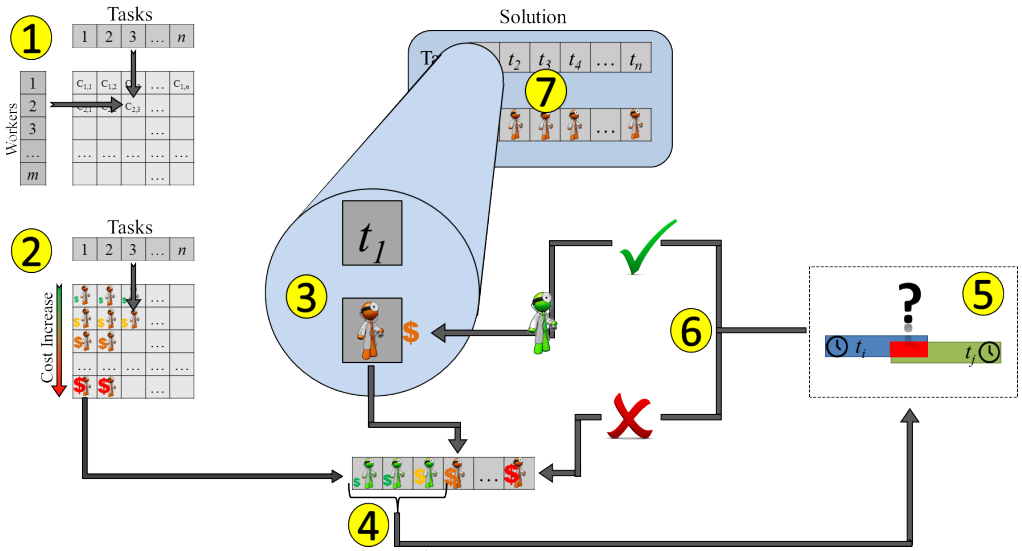


Figure 7.1: Diagram for the Priority-Based Search (PBS).

Figure 7.1 present the concepts of this local search. The steps as described as follows:

1. A $|T| \times |K|$ cost matrix C is defined, containing the estimated costs of assigning each worker k to each visit v. As explained above, an estimation of $f(\mathbb{X})$ is used for each assignment. More specifically, the cost c_{vk} for each assignment in C is the weighted sum of the *staff costs* (greedily selecting the cheapest contract); preferred *skills*, *workers* and *areas*; and worker availability for *area* and *time*. For each assignment where k does not have the required skills or a valid contract to perform the visit v we set $c_{vk} = \infty$.

2. A priority list P^v is built for each visit v, sorting the workers in ascending order of c_{vk}.

3. For each visit, following a random visiting order perform the next

steps 4 – 6.

4. Use P^v to determine if there is a worker with lower costs for the current visit v.

5. If one or more such workers exist, pick the first in the list and check if assigning that worker will generate a time conflict.

6. If the assignment can be done (no time conflict), then assign the worker and evaluate the solution. If this reduces the cost, the assignment is accepted, otherwise, the assignment is reverted. If the assignment generates a time conflict (hence reverted), repeat from step 5, selecting the next valid worker in the list.

7. Repeat from step 3 until all visits have been processed.

Because the priority list strongly restricts the search space, the proposed algorithm can iterate through all assignments much faster than the RHC, but there is a caveat: it is possible that the assignment of a worker with a higher cost than the lowest cost worker may allow the assignment of lower costs workers to other visits (because of time conflicts) such that the overall solution is improved. Thus, the local search may get stuck in a region of the search space and incapable of moving out. To approach this issue we adopt a 'best-of-α' approach where a new integer parameter α is introduced such that instead of assessing only the lowest-cost worker available, the search method now evaluates the first α lowest-cost workers. The value of α is adjusted according to the problem size in a way that larger problems have a smaller α (to improve the overall performance). Additionally, the VNS can dynamically increase the value of α for every iteration of the algorithm that does not present improvements. Algorithm 9 presents the pseudo-code of the Priority-

Based Search heuristic. Note that the objective-function weighing ensures that no time conflicts are present in the final solution.

Algorithm 9 Priority-Based Search

Require: Initial solution $\mathbb{X}$. Estimated-cost matrix C. $\alpha \geq 1$

```
for each assignment (v, k) following a random order do
    for y ← 1 to α do
        k' ← y-th feasible worker with the lowest-cost in C_v
        if f(X \ (v,k) ∪ (v,k')) < f(X) then
            X ← X \ (v,k) ∪ (v,k')
        end if
    end for
end for
return X
```

7.1.4 Availability Violations Seeker (AVS)

The next local search methodology, the Availability Violations Seeker. The concept of this search is presented in Figure 7.2 and Algorithm 10 and involves the following steps:

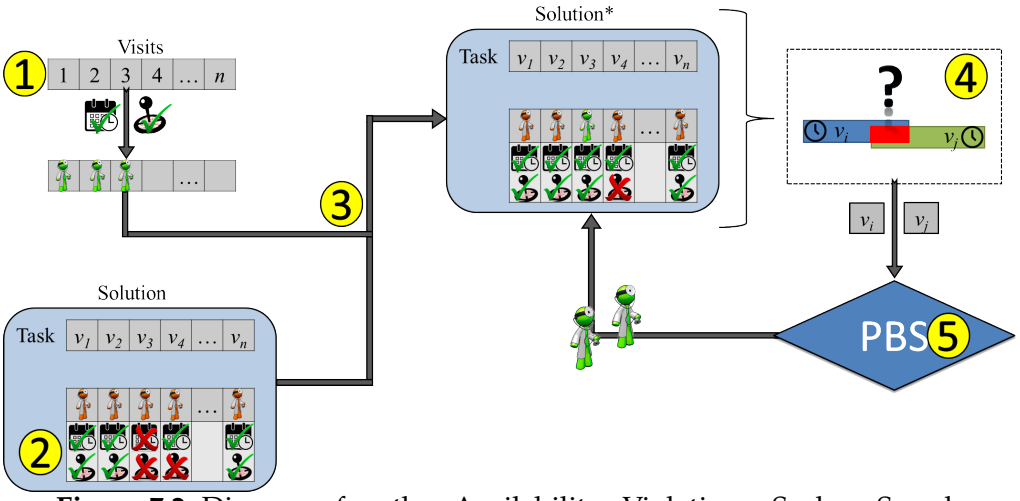


Figure 7.2: Diagram for the Availability Violations Seeker Search (AVS).

1. For each visit, identify the candidate worker with the minimum number of time and area availability violations for that visit. If

Algorithm 10 Availability Violations Seeker

Require: Initial solution $\mathbb{X}$. Estimated-cost matrix C. A list of candidate workers L. $\alpha \geq 1$. VIOL(v, k) function that returns the number of area and time availability violations in the given assignment. PBS(v, k) function which works exactly as the Priority-Based Search, but applied to a single given assignment (lines 2-7 of Algorithm 9).

1: **for each assignment** (v, k) **following a random order do**
2: **if** VIOL(v, k) > VIOL(v, L_v) **then**
3: $\mathbb{X}' \leftarrow \mathbb{X} \setminus (v, k) \cup (v, L_k)$
4: **for all** $(v, k') \in \mathbb{X}'$ conflicting with (v, L_k) **do**
5: $\mathbb{X}' \leftarrow$ PBS(v, k')
6: **end for**
7: **if** $f(\mathbb{X}') < f(\mathbb{X})$ **then**
8: $\mathbb{X} \leftarrow \mathbb{X}'$
9: **end if**
10: **end if**
11: **end for**
12: **return** $\mathbb{X}$

multiple workers meet that criterion, choose the one with the lowest value in the cost matrix C used by the PBS.

2. For each assignment (v, k), following a random visiting order, perform the next steps 3 – 7 (line 1 of the algorithm).

3. If the current assignment (v, k) presents a higher count of violations in time and area than the candidate worker identified for visit v in step 1 (line 2), then the worker is replaced (line 3).

4. Identify whether time conflicts were created in step 3 (line 4).

5. Where time conflicts occur, each conflicting visit is unassigned then perform steps 4 to 6 of the PBS heuristic to find a new worker for the visit (line 5). This will eliminate all time conflicts.

6. Evaluate the solution (line 7). If the cost improved (compared with those prior to step 3), accept the changes (line 8), otherwise, revert the changes.

7. Repeat from step 2 while there are still visits for which candidates were identified in step 1.

This local search exploits the fact that time and area availability violations are contained on a higher tier in the objective function (λ_3) and as shown later, violations of this kind are likely to be present in solutions. Hence, this heuristic attempt to reduce the number of these violations in order to improve the overall fitness.

7.1.5 Flat-Costs Search

Given a problem instance with $|T|$ visits and $|K|$ workers, the RHC performs $O(|T| \times |K|)$ solution evaluations in the worst case. The PBS attempts to use estimated costs to reduce the evaluations to a list of α best candidates, hence, it performs $O(\alpha \times |T|) = O(|T|)$ solution evaluations in the worst case. The new Flat-Costs Search extends that idea by bringing the number of solution evaluations to $O(1)$ in the worst case. To achieve that, it performs similar to the RHC (Algorithm 8), but instead of evaluating entire solutions (line 4), it uses the estimated costs matrix of the PBS to evaluate local assignments. The caveat is that the cost matrix does not inform the algorithm about potential time conflicts while the full evaluation penalises them. Thus, each potential assignment is checked for conflicts against visits currently assigned. The concept of this heuristic is presented in Figure 7.3 and Algorithm 11.

1. A $|T| \times |K|$ conflict matrix X is defined and describes where any two visits can be performed by the same worker (considering time, duration and travel time between locations). If any two visits (v, v') can be performed by a same worker without time conflicts, then $x_{vv'} = \checkmark$. Otherwise, in case time clashes occur, $x_{vv'} = \text{✗}$.

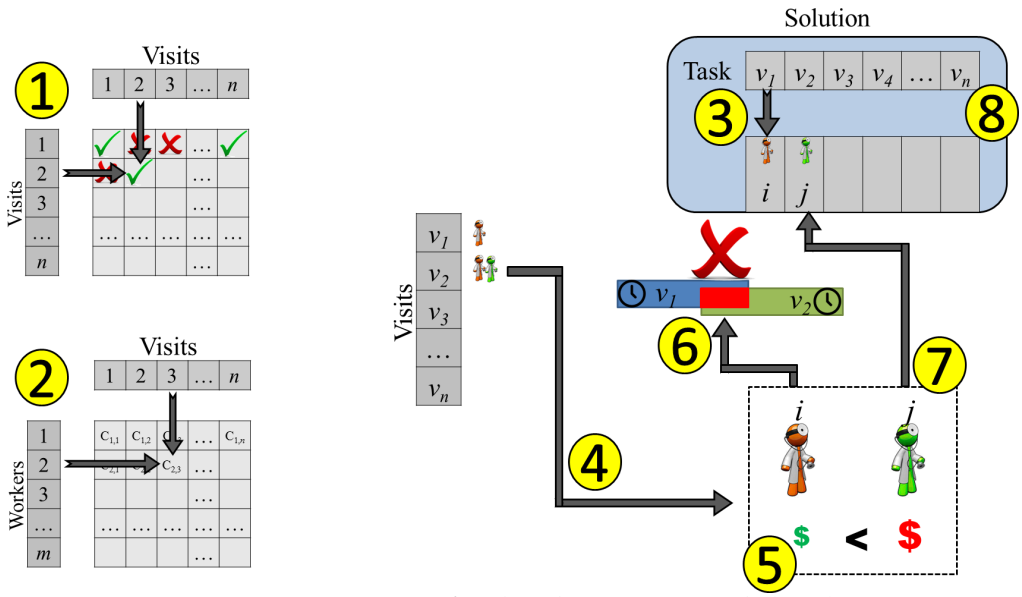


Figure 7.3: Diagram for the Flat-Costs Search (FCS).

Algorithm 11 Flat-Costs Search

Require: Initial solution $\mathbb{X}$. Conflict matrix X and cost matrix C. CONF($\mathbb{X}$) function that uses matrix X to evaluate if the given partial solution contains a time conflict.

1: **for each assignment** (v, k) **following a random order do**
2: **for each worker** $k' \in K$ **in ascending order in** C_v **do**
3: **if** CONF$(\mathbb{X} \setminus (v,k) \cup (v,k')) =$ ✓ **then**
4: $\mathbb{X} \leftarrow \mathbb{X} \setminus (v,k) \cup (v,k')$
5: **go to** line 7
6: **end if**
7: **end for**
8: **end for**
9: **return** $\mathbb{X}$

2. An estimated cost matrix C similar to the matrix used by the PBS is generated.

3. For each assignment (v, k), following a random visiting order, perform the next steps 4 – 7 (line 1 of the algorithm).

4. Sort all feasible workers by the values found in the column of C corresponding to the selected visit (line 2).

5. Select the worker with the smallest cost (line 2).

6. Using the matrix X, check if that worker can be assigned (checking his previous visits) (line 3). In the case his assignment would generate a time conflict, go back to step 5 and pick the next worker with the smallest cost.

7. When the worker with the smallest cost and no time conflicts is found, the solution is updated (line 4).

There is a potential issue with this local search: its advantages rely on its speed to allow the exploration of a wider region of the search space, hence, if the FCS is paired with slower heuristics on a VND procedure, the speed of the other heuristics will become a bottleneck and most of the time spent by the VNS will be running those other heuristics. Thus, we propose a variation of the VND procedure invoked by the VNS, here called FCS-VND. The idea is to run the slower heuristics once every ρ iterations of the VNS. On the remaining iterations, only the FCS is run. Hence, for $\rho = 1,000$, for the first 999 iterations of the VNS we perform a VND with only the FCS local search, on iteration 1,000 we perform the VND with the remaining local searches, then from iteration 1,001 to 1,999 we perform only the FCS again, and so on. Therefore, we use the ratio parameter ρ to specify how much time the algorithm should spend on the FCS in detriment of the other local searches. Algorithm 12 presents this variant of the VND.

The FCS-VND procedure (Algorithm 12) performs like a basic VND, thus it keeps processing while the solution can be improved (line 1). For every ρ times that the FCS-VND procedure was called (line 2), the regular VND is run with the PBS and AVS local searches (lines 3 – 5). However, on the remaining times, the FCS local search is called instead (line 7). If the solution is an improvement from the previous one (line 9), the

Algorithm 12 FCS-VND

Require: Initial solution $\mathbb{X}$. Ratio parameter ρ. Number of times x that the FCS-VND procedure has been invoked by the VNS procedure. Set of local search structures N_l, for $l = 1, \ldots, l_{max}$.

1: **while** there is improvement **do**
2: **if** $(x$ **mod** $\rho) = 0$ **then**
3: **for** $l \leftarrow 1$ **to** l_{max} **do**
4: $\mathbb{X}' \leftarrow$ best neighbour of $\mathbb{X}$ using the neighbourhood N_l
5: **end for**
6: **else**
7: $\mathbb{X}' \leftarrow \text{FCS}(\mathbb{X})$
8: **end if**
9: **if** $f(\mathbb{X}') < f(\mathbb{X})$ **then**
10: $\mathbb{X} \leftarrow \mathbb{X}'$
11: $l \leftarrow 0$
12: **else**
13: $l \leftarrow l + 1$
14: **end if**
15: **end while**
16: **return** $\mathbb{X}$

new solution is kept (line 10) and the processing continues until no more improvements are found (line 1).

7.1.6 Constructive Heuristics

Castillo-Salazar et al. [44] raised the importance of a constructive heuristic to problems combining scheduling and routing, such as the HHC, as it may be difficult to find an initial feasible solution. Therefore, we propose two constructive heuristics to obtain a good initial solution.

The first method is the straightforward greedy heuristic presented in Chapter 3 (Algorithm 5. It is important to notice that due to the random visitation order, running the algorithm multiple times may result in different solutions. Also, the objective function weighting ensures that no time conflicts are present in the final solution.

Analogously to the FCS heuristic, the second constructive heuristic

uses the estimated cost matrix to generate a new solution. Basically, this is a version of the FCS starting from an empty solution. Hence, it builds the solution as it proceeds. Also, because of the random visit order of assignments, running the algorithm multiple times may outcome different solutions.

7.2 Experimental Results

All experiments were performed on Intel quad-core i7 machines with 16GB DDR2 RAM memory. The presented values result from the average of eight runs. For scenarios A, B and C the runtime limit for the metaheuristics was 15 minutes, for scenarios D and E was one hour and for scenario F, eight hours (simulating an overnight run because of the size of the instances).

To evaluate the proposed algorithms, five distinct experiments were perfomed. First, we evaluate the impact of a good starting point on the proposed VNS. Next, we assess how the proposed local search components affect the overall results. Lastly, we evaluate the Flat-Costs Search on the largest scenarios and the overall performance of the algorithm against known optimal solutions (for the smallest scenarios) and against other metaheuristics and techniques found in the literature.

The parameters for the experiments are: ρ – the ratio between using the quicker FCS and the slower AVS and PBS local searches; the stop criterion previously described – run time; and α (PBS only) – set to 100 for datasets A, B and C, 20 for datasets D and E and 5 for dataset F.

Also, the experiments refer to several variants of the proposed algorithm, as we compare individual components and configurations. The algorithm variants are as follows:

- **Full-VNS** is the full proposed algorithm including the FCS, PBS and AVS as the local searches and all seven shaking neighbourhoods. The frequency ρ is adjusted according to the problem size such that for datasets A, B, C, D and E it is set to 1,000 and to dataset F it is set to 10,000. We chose these values because the larger presented better results for dataset F and the smaller presented better results for the remaining datasets. Additionally, the following configurations are considered:
  - **1K-VNS** is the Full-VNS with the frequency $\rho = 1,000$, hence for every 1,000 iterations of the FCS the AVS and PBS are executed once.
  - **10K-VNS** is the Full-VNS with the frequency $\rho = 10,000$, hence for every 10,000 iterations of the FCS the AVS and PBS are executed once.
- **AVPB-VNS** is the version of the VNS algorithm presented in [176], including the AVS and PBS local searches (not the FCS) and all seven shaking neighbourhoods.
- **FCS-VNS** is the Full-VNS without AVS and PBS (using only FCS as the local search).
- **PBS-VNS** is the Full-VNS without AVS and FCS (using only PBS as the local search).
- **AVS-VNS** is the Full-VNS without PBS and FCS (using only AVS as the local search).
- **HC-VNS** uses only six shaking neighbourhoods (excluding the Priority-based Flip) and uses only RHC as the local search (not PBS, AVS nor FCS).

We use charts to illustrate the results, however Tables 7.1 and 7.2 presented towards the end of this section contain the detailed results of all experiments.

7.2.1 Starting Point Evaluation

We performed experiments to assess if a constructive heuristic is required, or beneficial, for producing improved solutions. [44] showed a computational study on other instances of the WSRP and concluded that it was very difficult to find feasible solutions, however, these instances differ from those in not having time windows or visit dependencies.

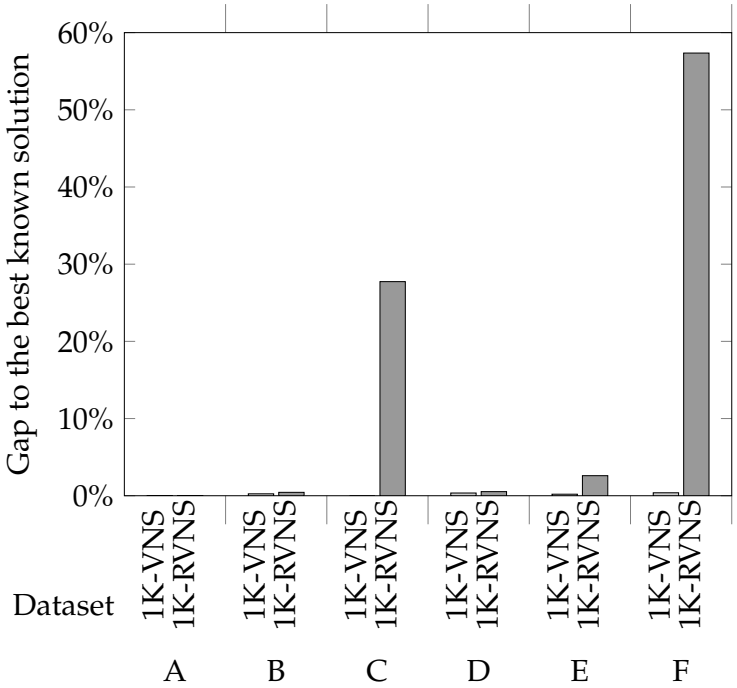


Figure 7.4: Average gap to the best known solution for each set. Featuring the proposed VNS starting from the constructive heuristics (1K-VNS) and from a random solution (1K-RVNS).

The VNS algorithm was executed using all three local searches and all shaking neighbourhoods, also the frequency $\rho = 1,000$ in this experiment. Figure 7.4 presents comparative results for starting from a random solution (*1K-RVNS*) and starting from the best solution obtained by run-

ning both constructive heuristics (*1K-VNS*).

Each value represents the average gap between the average solution computed and the single best solution found among all runs of all algorithms. For some instances (sets A and B and instances C2, C4, C5 and C7), the best is also the optimal solution, obtained by a mathematical solver. No effect was seen for set A, a small benefit for sets B, D and E, and a significant benefit for sets C and F. In sets C and F, the VNS had difficulties to find a feasible solution without the good initial solution, resulting in a large gap. These sets possess the largest pool of workers (Table 3.4), and, after inspecting the stochastically generated solutions, it is clear that they violate the required skill constraint in the majority of the assignments.

Thus, considering the small computational costs of the constructive heuristics (especially the FCH), it is beneficial to obtain an initial feasible solution before applying the improvement metaheuristic.

7.2.2 Evaluation of Individual Components

Next, we evaluate the performance of the PBS, AVS and FCH local searches and assess if combined they improve the overall performance.

Results are shown in Figure 7.5, where the y axis presents the gap between the average solution found by each algorithm for each of the problem instances in each dataset, compared to the best solution found by all runs of all algorithms for that instance. All three algorithm variants perform well for the smaller sets (A and B). However, from C onwards, PBS-VNS and AVS-VNS alternate, showing that on some scenarios one local search has an edge over the other. Notably, the AVS provided better solutions than the PBS for scenarios where the number of time and area availability violations is high, whereas PBS provided better solutions for

the remaining scenarios. The Full-VNS produced better results on all instances, especially on the largest sets.

Additionally, the FCS-VNS alone provided results almost as good as the Full-VNS. It is clear the objective-costs estimation is accurate enough to allow the heuristic to find good solutions and that the extra iterations allow the algorithm to explore a wider region of the search space, hence achieving improved results. Nonetheless, benefits could be obtained by combining the three local searches.

7.2.3 Overall Performance

Next, we compare the overall performance of the algorithm. To that purpose, we compare our improved VNS (Full-VNS) against a VNS using only the hill climbing local search (*HC-VNS*) and the VNS using only the AVS and PBS local searches (AVPB-VNS). Figure B.13 presents the results.

The HC-VNS manages to present good solutions – the gap is always

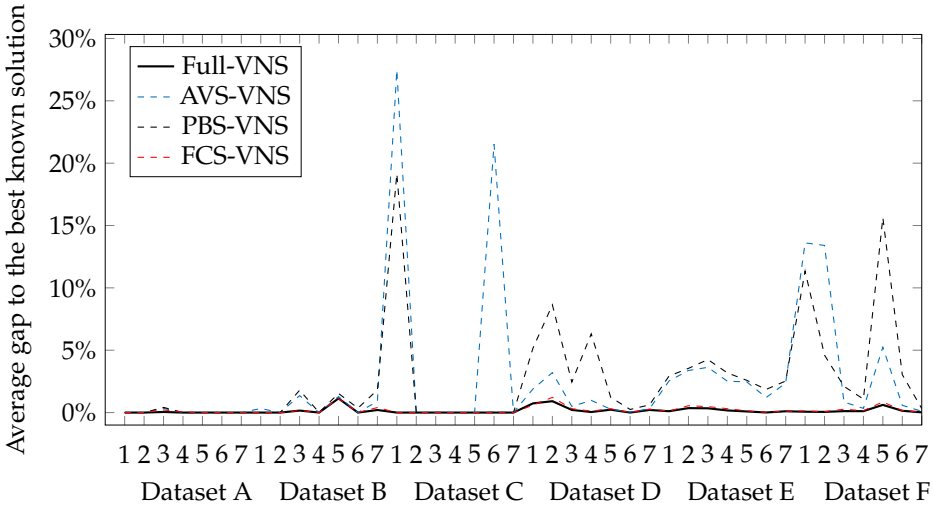


Figure 7.5: Relative gap comparison between the proposed VNS (Full-VNS), a VNS with only the AVS local search (AVS-VNS), a VNS with only the PBS local search (PBS-VNS), and a VNS with only the FCS local search (FCS-VNS)

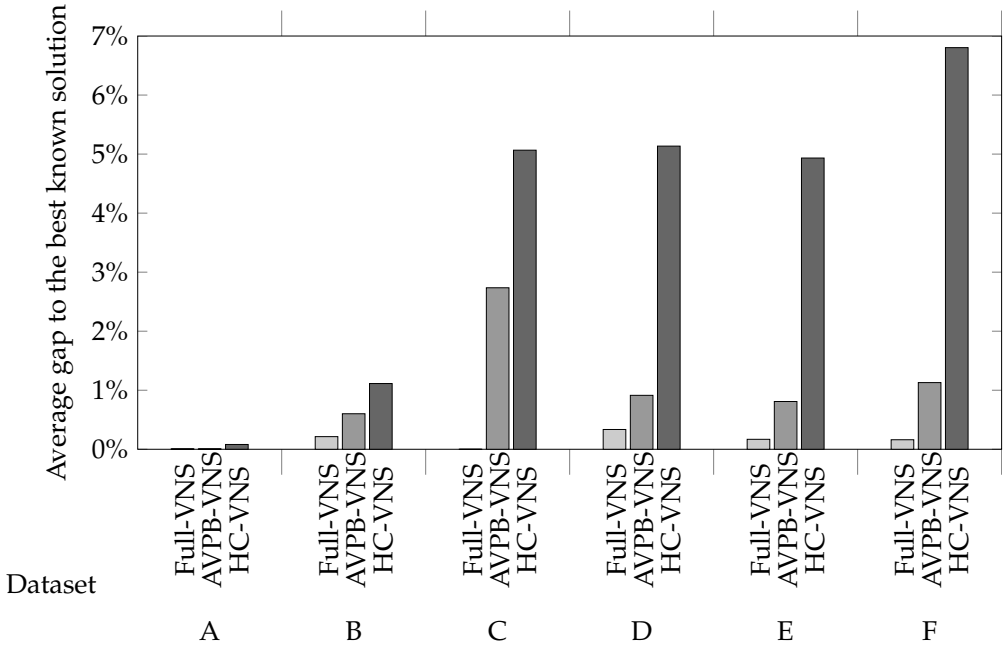


Figure 7.6: Comparison of the proposed VNS (Full-VNS) against the VNS with a simple hill climbing neighbourhood (HC-VNS) and a VNS with only the AVS and PBS local searches (AVPB-VNS) in relation to the average gap to the best known solution for each set.

below 7%. However, as the size of the problem grows, the performance of that local search decreases. That happens because the hill climbing is slow for this problem. A single iteration of the HC-VNS algorithm for the largest F instance took over eight hours while the AVPB-VNS could iterate in less than 20 minutes and the Full-VNS in a fraction of a second. Nonetheless, the proposed Full-VNS presented better solutions in all scenarios and it can be used to provide faster solutions where the HC-VNS cannot.

Moreover, for the scenarios where the optimal solution is known (sets A and B and scenarios C2, C4, C5 and C7), the proposed Full-VNS was able to reach the optimal in all cases and in all runs while other investigated algorithms were not (Table 7.2).

7.2.4 Flat Cost Search

Following the overall performance comparison, we now perform a thorough analysis on the novel FCS and explain how we set the ρ parameter. As previously stated, the FCS performs, in the worst case, $O(1)$ solution evaluations, while the PBS and HC perform, in the worst case, $O(|T|)$ and $O(|T| \times |K|)$ solution evaluations respectively. On dataset F, a single evaluation can take roughly one second.

Figure 7.7 presents the count of the number of iterations performed by versions of the VNS with only the HC local search (HC-VNS), only the PBS local search (PBS-VNS) and only the FCH local search (FCH-VNS). Note that the graph uses a logarithmic scale to better display the values. On the smallest instance, the PBS is ten times faster than the HC and this number grows to 50 times faster on the largest instance. The FCS, however, is over 1,000 faster than the PBS on the smallest F instance and over 1.600 faster on the largest. This improved speed allows the FCS to explore a wider region of the search space and, although arguably there is a loss of precision because of the estimation of objective-values, the results show that the superior speed more than compensates that.

For datasets A through E, we selected $\rho = 1,000$ because in these scenarios the PBS and AVS takes a few seconds for a single iteration in the worst case. Hence, with that ρ value we have a good balance between the local searches. However, for dataset F, because the AVS and PBS together can take around 20 minutes, $\rho = 1,000$ means that most of the time the algorithm will be processing the AVS and PBSm hence it will not be taking advantage of the speed of the FCS. Therefore, we set $\rho = 10,000$ and performed more experiments.

Figure 7.8 presents the results, where the FCS-VNS is the VNS using only the FCS, the 1K-VNS is the Full-VNS with $\rho = 1,000$ and 10K-VNS

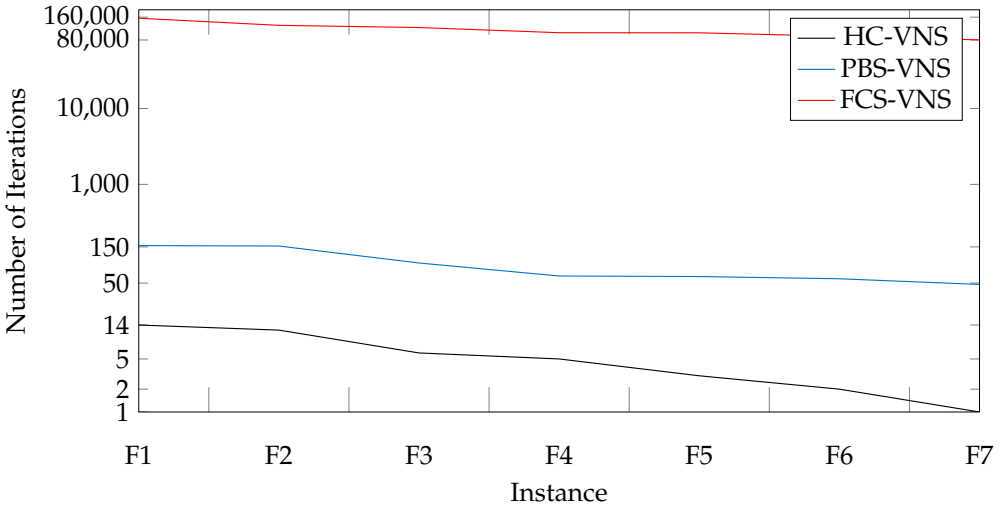


Figure 7.7: Number of iterations for the HC-VNS, the PBS-VNS and the FCS-VNS on eight hours runs for the F dataset.

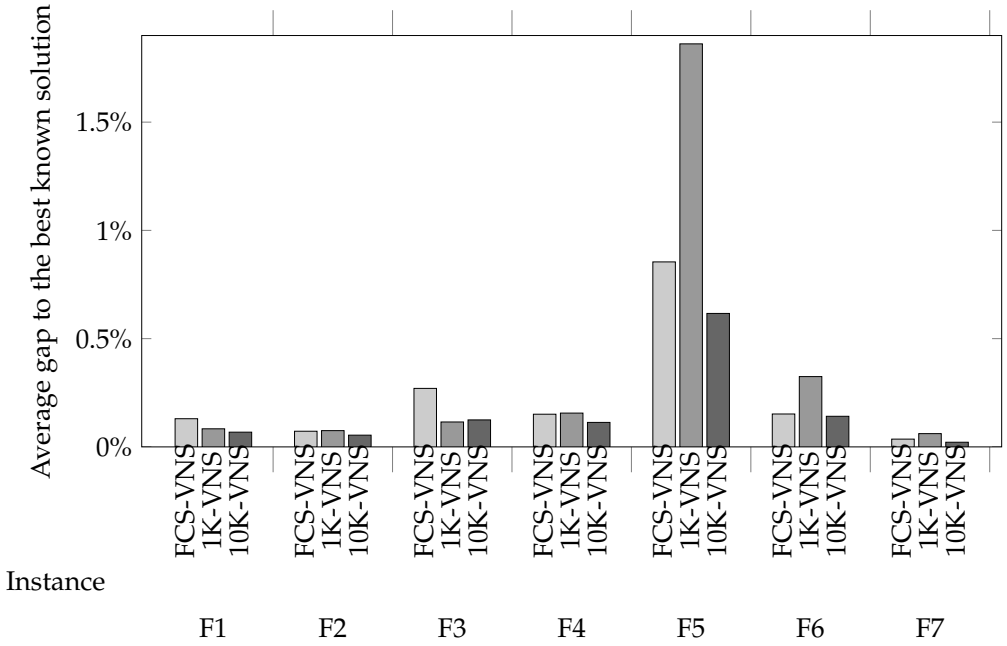


Figure 7.8: Comparison of the VNS using only the FCS local search (FCS-VNS) and the VNS using all local searches and a ρ value of 1.000 (1K-VNS) and 10.000 (10K-VNS).

is the Full-VNS with $\rho = 10,000$. In the chart, we show the results only for the F instances because those instances were the only ones where $\rho = 1,000$ underperformed. We see that on instances F1 and F3, the 1K-VNS provide better results than the FCS alone. On the remaining

instances, the FCS alone can provide better results than the combined local searches with $\rho = 1,000$. That happens because the VNS spends a large majority of its run time processing the AVS and PBS local searches. Therefore, when we increase the ratio to 10,000, we increase the time spent on the FCS, hence the algorithm (10K-VNS) provides the best results on those instances and the combination of local searches always represents improvements.

7.2.5 Comparative Results

Finally, we compare the results of the proposed VNS algorithm against two other techniques available in the literature:

- The Genetic Algorithm (GA) proposed by [4] using an indirect representation of its individuals to restrict the search space to the feasible region; cost-based tailored operators to increase the performance; 500 individuals populations for sets A, B and C; 250 individuals population for sets D and E; and 100 individuals population for set F. The runtime is the same as the VNS: 15 minutes for sets A, B and C; 60 minutes for sets D and E; and eight hours for set F.

- The decomposition (Decomp.) technique proposed by [139] which initially decomposes the problem by geographical locations to solve each sub-problem to optimality using a mathematical solver. After solving the subproblems, a conflict-repair routine tackles conflicting visits in different sub-problems. Later, a heuristic algorithm greedily amasses the final solution using the resulting sub-problem solutions.

Figure 7.9 presents the results. Each data point represents the av-

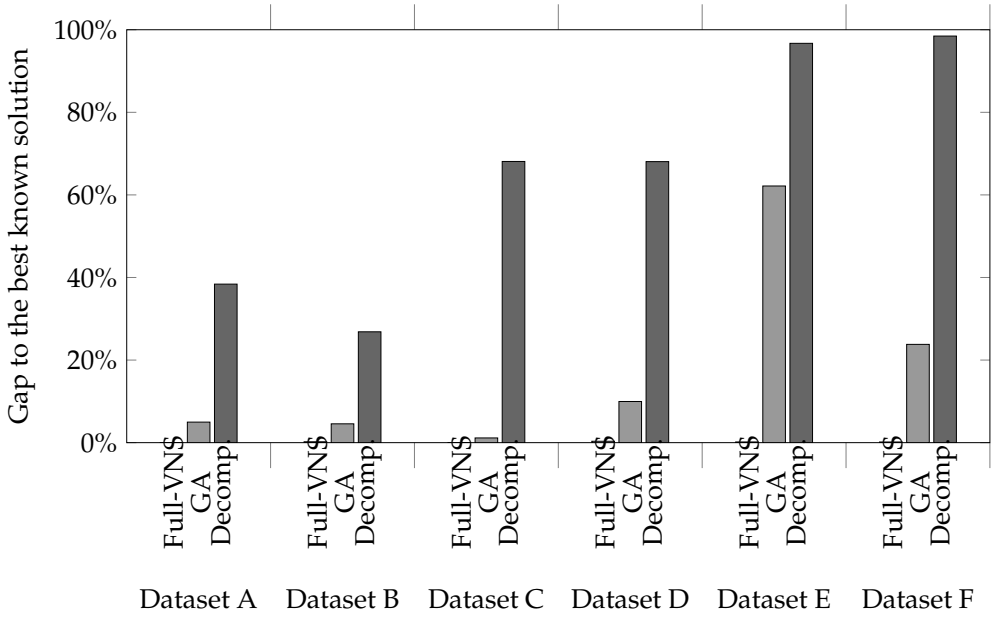


Figure 7.9: Comparative results between the proposed Full-VNS, the genetic algorithm (GA) and the decomposition technique using exact methods (Decomp.).

erage gap to the best-known solution of all solutions obtained from an algorithm in the respective dataset. Clearly, the proposed VNS presents the best results. The difference from the GA is not large on sets A, B and C, however on the larger sets (D onwards) the difference is noticeable. The decomposition technique showed the worst results in all instances. Noticeably, the GA performed better on the largest set F than on set E. That happened because E is the set with the highest number of feasible workers per visit, meaning a vast number of solutions with close objective-values, and while the VNS could successfully escape local optima in these circumstances, the GA presented difficulties.

7.2.6 Constructive Heuristics

Figure 7.10 shows the average gaps obtained by the constructive algorithms alone (without the VNS). We observe that both techniques have similar performances with the Greedy on par with the FCH on sets A

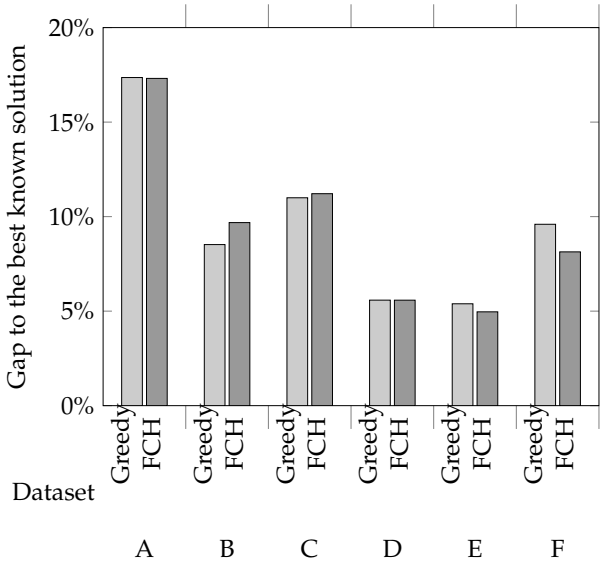


Figure 7.10: Performance comparison of the FCH heuristic against the Greedy heuristic.

and D, slightly superior on sets B and C and slightly inferior on sets E and F. However, the difference is minimal.

Because the constructive heuristics are designed to provide an initial solution to the VNS, it is important that they perform fast. Figure 7.11 presents a comparison between the runtimes for the Greedy and the FCH heuristics. For sets A, B and C both heuristics perform fast, providing solutions in less than a second. The greedy algorithm is much slower for sets D and E (up to 107 seconds compared to less than one second for FCH). The performance gap is huge for the largest scenario, with the greedy heuristic taking up to 26 minutes to produce one solution but the FCH still taking less than one second.

Therefore on the smallest scenarios (A, B), where the time performance of both heuristics is the same, the Greedy can produce slightly better solutions. On the medium-sized scenario C, the performance and time consumption of both algorithms is equivalent. On the large scenarios D and E, the FCH was able to provide better results on E only, but presented a considerably superior time performance. Finally, on the

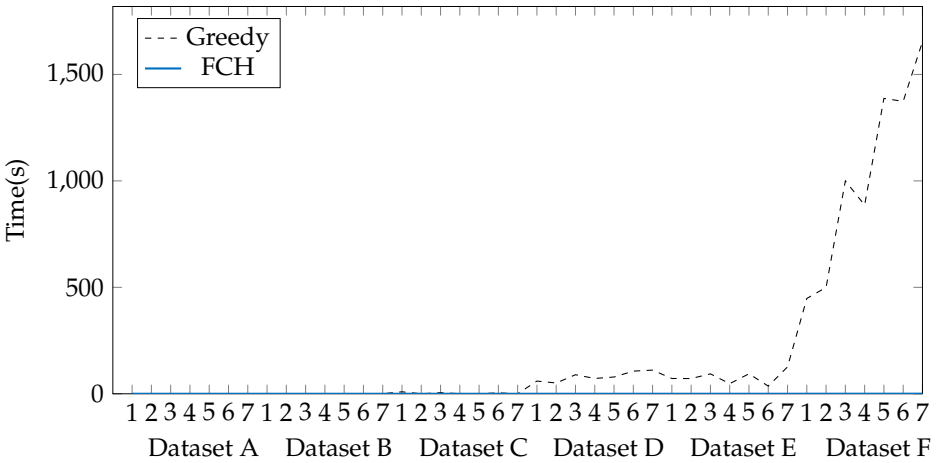


Figure 7.11: Time comparison between the FCH heuristic and the Greedy heuristic.

extra-large set F, the FCH presented slightly superior solutions but far superior time performance.

Table 7.1: Detailed results of the experiments for datasets A, B and C.

		Constructive		Partial VNS					Full VNS				Other Methods		Best
		Greedy	FCH	HC-VNS	AVS-VNS	PBS-VNS	AVPB-VNS	AVPB-RVNS	FCS-VNS	1K-VNS	10K-VNS	1K-RVNS	GA	Decomp.	Known†
A	**1**	3.81	3.70	**3.49**	**3.49**	**3.49**	**3.49**	**3.49**	**3.49**	**3.49**	**3.49**	**3.49**	**3.49**	5.65	*3.49**
	2	3.35	3.52	**2.49**	**2.49**	**2.49**	**2.49**	**2.49**	**2.49**	**2.49**	**2.49**	**2.49**	2.97	4.53	*2.49**
	3	4.91	5.00	3.01	3.01	3.01	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	3.59	10.65	*3.00**
	4	1.57	1.56	**1.42**	**1.42**	**1.42**	**1.42**	**1.42**	**1.42**	**1.42**	**1.42**	**1.42**	**1.42**	3.09	*1.42**
	5	3.10	2.96	**2.42**	**2.42**	**2.42**	**2.42**	**2.42**	**2.42**	**2.42**	**2.42**	**2.42**	**2.42**	3.54	*2.42**
	6	3.63	3.92	**3.55**	**3.55**	**3.55**	**3.55**	**3.55**	**3.55**	**3.55**	**3.55**	**3.55**	3.62	3.74	*3.55**
	7	4.34	4.10	**3.71**	**3.71**	**3.71**	**3.71**	**3.71**	**3.71**	**3.71**	**3.71**	**3.71**	**3.71**	4.81	*3.71**
B	**1**	1.86	1.90	**1.70**	1.71	**1.70**	**1.70**	**1.70**	**1.70**	**1.70**	**1.70**	**1.70**	1.76	1.79	*1.70**
	2	1.81	1.83	**1.75**	**1.75**	**1.75**	**1.75**	**1.75**	**1.75**	**1.75**	**1.75**	**1.75**	**1.75**	1.89	*1.75**
	3	2.01	2.01	1.77	1.74	1.75	1.75	1.76	**1.72**	**1.72**	**1.72**	1.73	1.93	2.06	*1.72**
	4	2.13	2.13	2.09	**2.07**	**2.07**	**2.07**	**2.07**	**2.07**	**2.07**	**2.07**	**2.07**	2.08	2.21	*2.07**
	5	2.03	2.08	1.86	1.85	1.85	1.85	1.86	**1.84**	1.85	**1.84**	1.85	1.96	4.74	*1.82**
	6	1.83	1.86	1.64	**1.62**	1.63	**1.62**	**1.62**	**1.62**	**1.62**	**1.62**	**1.62**	1.72	2.52	*1.62**
	7	1.97	2.01	1.81	1.81	1.82	1.81	1.82	1.80	1.80	**1.79**	1.81	1.87	4.06	*1.79**
C	**1**	148.65	149.09	152.46	157.47	141.26	141.25	**114.21**	**114.21**	**114.21**	**114.21**	**114.21**	119.25	904.98	*114.21*
	2	**3.05**	**3.05**	**3.05**	**3.05**	**3.05**	**3.05**	**3.05**	**3.05**	**3.05**	**3.05**	**3.05**	**3.05**	3.61	*3.05**
	3	141.79	135.51	103.93	103.53	103.54	103.53	2879.02	**103.52**	**103.52**	**103.52**	2879.08	106.15	1186.27	*103.52*
	4	11.48	11.68	**11.15**	**11.15**	**11.15**	**11.15**	**11.15**	**11.15**	**11.15**	**11.15**	**11.15**	**11.15**	81.25	*11.15**
	5	12.81	12.89	**12.34**	**12.34**	**12.34**	**12.34**	**12.34**	**12.34**	**12.34**	**12.34**	**12.34**	**12.34**	68.94	*12.34**
	6	176.19	181.58	156.00	178.94	140.47	**140.44**	6302.68	**140.44**	**140.44**	**140.44**	6302.55	142.26	3102.26	*140.44*
	7	**4.30**	**4.30**	**4.30**	**4.30**	**4.30**	**4.30**	**4.30**	**4.30**	**4.30**	**4.30**	**4.30**	**4.30**	5.29	*4.30**

† Single best solutions found among all runs of all algorithms.

* Optimal solution guaranteed by a mathematical solver.

Table 7.2: Detailed results of the experiments for datasets D, E and F.

		Constructive		Partial VNS						Full VNS			Other Methods		Best
		Greedy	FCH	HC-VNS	AVS-VNS	PBS-VNS	AVPB-VNS	AVPB-RVNS	FCS-VNS	1K-VNS	10K-VNS	1K-RVNS	GA	Decomp.	Known†
D	1	178.17	177.98	180.37	170.54	176.42	170.40	170.02	168.32	168.63	168.56	**168.20**	188.53	496.45	*167.32*
	2	179.37	178.63	181.69	165.16	174.99	164.41	164.35	161.84	161.47	161.34	**161.23**	193.60	372.94	*159.87*
	3	185.18	185.18	183.54	178.33	181.90	178.19	179.55	178.01	**177.83**	178.25	178.87	194.15	3213.32	*177.45*
	4	178.44	178.75	182.96	167.01	176.51	166.80	166.85	165.57	**165.45**	**165.45**	165.58	196.72	418.89	*165.37*
	5	164.69	164.51	163.22	161.13	162.70	161.12	161.92	161.20	**161.08**	161.20	161.63	172.91	243.89	*160.70*
	6	179.50	179.93	178.40	177.45	177.82	177.44	177.99	**177.38**	**177.38**	**177.38**	178.01	187.70	1411.27	*177.38*
	7	180.17	180.48	180.38	177.93	178.49	177.86	178.42	177.88	**177.76**	177.94	178.31	184.81	753.28	*177.39*
E	1	1.221	1.224	1.241	1.195	1.200	1.169	1.185	**1.166**	1.167	**1.166**	1.167	3.456	33.012	*1.165*
	2	1.252	1.254	1.238	1.211	1.214	1.177	1.193	1.177	**1.175**	**1.175**	**1.175**	3.179	26.050	*1.171*
	3	1.273	1.266	1.266	1.236	1.244	1.196	1.234	1.197	**1.195**	1.196	1.198	4.861	28.990	*1.191*
	4	1.348	1.347	1.355	1.305	1.314	1.276	1.294	1.276	1.276	**1.274**	1.277	2.987	28.518	*1.272*
	5	2.551	2.479	2.305	2.280	2.283	2.275	2.327	2.227	**2.226**	2.227	2.662	4.735	270.088	*2.224*
	6	1.357	1.356	1.332	1.303	1.312	1.293	1.307	1.288	**1.287**	1.288	1.288	3.541	24.628	*1.287*
	7	1.760	1.756	1.767	1.737	1.740	1.719	1.734	**1.697**	**1.697**	1.698	1.698	3.859	427.780	*1.696*
F	1	2426.1	2542.3	2318.7	2326.4	2267.1	2017.7	575832.3	2012.5	2011.5	**2011.2**	575827.8	2261.1	64305.1	*2009.8*
	2	2334.4	2178.8	2257.3	2391.1	2170.6	2077.2	1001432.3	2072.2	2072.2	**2071.8**	1001427.5	2306.8	73291.2	*2070.7*
	3	670.1	620.2	614.2	603.9	611.7	603.7	305976.0	600.4	**599.5**	**599.5**	305974.0	838.3	115235.2	*598.8*
	4	1345.6	1345.0	1335.6	1322.8	1332.8	1323.1	1325.3	1320.4	1320.4	**1319.9**	1320.4	1551.2	102994.2	*1318.4*
	5	216.4	215.8	221.9	191.4	214.9	191.6	189.1	182.9	184.8	**182.5**	184.4	469.0	101438.2	*181.3*
	6	693.8	692.7	640.5	621.0	636.7	620.9	622.3	618.3	619.4	**618.2**	620.9	864.9	76007.1	*617.4*
	7	3565.9	3565.1	3496.6	3488.0	3495.8	3488.1	2555127.8	3485.7	3486.6	**3485.2**	2555126.4	3947.8	176540.6	*3484.4*

† Single best solutions found among all runs of all algorithms.

7.3 Conclusion

The size of the instances in sets D, E and especially F, provoke unacceptably high runtimes for a simple greedy algorithm, potentially hindering the performance of search methodologies that rely on the systematic exploration of neighbourhoods (such as the HC-VNS). Also, that happens because evaluating a solution can be costly on larger scenarios due to a large number of constraints and objectives. Hence, the design of our Priority-Based Search and the Flat-Costs Search proved to be much more efficient, as they allow a much faster exploration of the neighbourhoods of the current solution, hence achieving a faster convergence to an improved solution.

Compared to other techniques, the constructive heuristics present competitive results. Overall, the average gap of these heuristics to the best-known solution is approximately 9.5%. The nature of the problem favours the assignment of the lowest cost worker (considering the objectives and soft constraints) to each task when there are no time conflicts. The preferred worker constraint relates to the continuity of care, i.e. the same worker providing services to the same patients. Thus, a worker that is preferred for a visit is usually the worker who most often performs visits for that patient. In addition to being skilled and having a valid contract, that worker is often also available in the area of the visit (and possibly on time as well), and is likely to live nearby (thus, the distance to the patient is short). Hence, the assignment of that worker is likely to generate a low cost in the objective-function.

Considering that the FCH provided results averaging only 9% worse than the proposed VNS, and provided solutions in less than a second (even for the largest scenario, where we run the VNS for eight hours),

this constructive heuristic may be sufficient to provide good enough solutions very quickly, illustrating the efficacy of the developed approach for combining the objectives considered here.

It was not expected that the AVS alone could present results comparable to the Randomised Hill Climbing and the Priority-Based Search because the AVS is designed to fix area and time violations, hence, this methodology is incapable of improving the results when the number of violations is zero. The improvements came from a chain reaction triggered by the shaking step, which caused the violations to appear and the AVS heuristic to fix them (and improve the solution).

The novel FCS proved that the cost estimation as a mean to gain computational performance is very effective as we could not only obtain much faster iterations, but also improve the quality of the solutions obtained by using the speed to explore a wider region of the search space. However, even though the FCS alone could provide excellent results, the combination of AVS, PBS and FCS provided even better results, given that we allow the FCS to perform considerably more iterations with the proposed VND variant.

Regarding the characteristics of the studied WSRP instances, the fact that the FCS and PBS outperformed the RHC in all scenarios means that, even on small scenarios where computational time is not an issue, the restriction of workers by the priority list based on estimated costs do not eliminate the best candidates for assignments. Also, the estimation of costs can produce very accurate results. We can reinforce this claim with the AVS heuristic, which fixes the time and area availabilities using the first worker available in the priority list and can also reach the optimal and provide some of the best results. The fact that the top elements of the priority list can provide improved results also supplement

the aforementioned statement regarding this problem being well-suited for greedy algorithms.

Moreover, when comparing the results of the PBS and the AVS heuristics, it is noticeable that the performance varies. The latter technique performs 3.5% and 2.6% better for sets D and F, respectively. This shows that the occurrence of area and time violations are common in these datasets and that a focus on fixing these violations is beneficial. On the other hand, for set C the priority-based approach performs 4% better.

Finally, when comparing all algorithms, it is clear that improvements in the solution quality can be obtained by employing domain knowledge into the solver design. HC-VNS includes some domain-specific shaking methods, giving reasonably good solutions. FCS-VNS, AVS-VNS and PBS-VNS use more domain-specific knowledge, giving even better results. The full proposed VNS makes the most use of domain-knowledge and provides the best results overall.

Chapter 8

Using VNS on Estimated Pareto Fronts to Solve Multiobjective Problems

The literature presents a few techniques that use domain knowledge to estimate Pareto fronts. Giagkiozis and Fleming [88] propose a technique to estimate the Pareto front of continuous multiobjective problems and then use the estimated front to obtain values for the decision variables of interesting solutions. Their technique is particularly useful because it can be applied to subregions of the objective space. Another estimation technique includes the Bayesian multiobjective optimisation, which consists of using a Bayesian model to learn computationally expensive objective functions and use the estimation model to quickly explore the search space [77].

In Chapter 6 we presented a study about the multiobjective nature of the HHC problem. We showed that modern multiobjective algorithms struggle to perform on the largest HHC datasets because of the large size of the instances and number of objectives (five). We also showed that in-

stances of the same dataset have similar fitness landscapes. This chapter presents a methodology to exploit these multiobjective characteristics of the HHC problem.

The proposed methodology works as follows: for a given dataset consisting of multiple instances, a single *pilot* instance is solved using multiobjective algorithms as presented in Chapter 6 or using the best available method. A decision-maker uses the approximation set of the pilot instance to choose *target* solutions for the remaining instances of the dataset. Then, goal programming is applied with the VNS proposed in Chapter 7 to obtain solutions that are close enough to (or better than) the target solutions.

Note that this methodology is not exclusive to the HHC problem and it can also be applied to any optimisation problem where instances of the same dataset present a similar fitness landscapes and the performance of multiobjective algorithms is weak.

8.1 Goal Programming (GP)

Goal programming is a field of multiobjective optimisation and an alternate approach to tackle multiobjective problems [49]. Basically, goal programming consists of establishing a specific numeric goal for each of the objectives of the problem, and then search for a solution that minimises the weighted sum of deviations of these objective functions from their respective goals.

There are three types of goals in goal programming [133]:

- Lower bound: defines a lower value for an objective such that solutions that fall below the lower value are penalised.
- Upper bound: defines an upper value for an objective such that

solutions that present higher values than the upper value are penalised. In the MOHHC problem considered in this thesis, the goals are of the upper bound type. Note that even though the preferences are meant to be maximised, the preference-related objectives (Z_3, Z_4 and Z_5) have been formulated as minimisation objectives.

- Strict bound: defines a specific target value such that solutions that present values above or below are penalised.

Once the target goals are set, goal programming techniques derive problem models (LP, MIP, etc.) to find solutions that reach (or are close enough to) the target goals. Several strategies, or goal programming variants, are presented in the literature. We briefly review the three most widely employed variants [117]:

- **Weighted GP** [192]: used when the decision maker is able to assign an *importance* weight to each goal. The objective function of the problem comprises a weighted sum of the deviation to the target values. The methodology proposed in this chapter does not assume that the decision maker can establish importance weights, hence this GP technique is not used in our work. However, if it is the case that the decision maker can define weights, weighted GP can be a good alternative to the techniques proposed.

- **Lexicographic GP**: when weighting the goals is difficult, but the decision maker is able to prioritise them, the lexicographic GP technique is commonly applied [209]. The deviations to the target goals are minimised according to defined priority levels such that deviations to a higher level goal are infinitely more important that deviations of a lower level goal. Because our proposed methodology

attempts to reach an exact solution, all deviations are equally important and the lexicographic GP technique is not applicable.

- **Chebyshev GP** [79]: consists of minimising the maximum weighted normalised deviation, hence promoting solutions well-balanced regarding the achievement of the target values. This technique is detailed in the next section.

A potential issue with goal programming is that it may produce solutions that are not Pareto efficient (e.g. one or more criteria can be improved without worsening any other criterion) [116]. This is especially true when the goals are 'pessimistic' and the objectives can be easily achieved. Several methods are proposed to address the issue. Most methods rely on extra information from the decision maker in order to promote the further improvement of certain objectives [210]. Other methods involve extending the search after the solution is found by the goal programming in order to find dominating solutions [100].

The literature usually presents goal programming with exact methods [116, 117]. However, many works exist where metaheuristics are employed to solve goal programming models. Baykasoglu [19] presents a simulated annealing to tackle several test problems of preemptive goal programming. Mishra et al. [161] employ a fast converging simulated annealing to solve a machine-tool selection and operation allocations problem with fuzzy variables. Ghoseiri and Ghannadpour [86] propose a genetic algorithm to tackle a goal programming model for the vehicle routing problem with time windows and Leung [146] presents genetic algorithm to tackle a goal programming model for a transportation planning problem with three objectives.

Moreover, in the context of healthcare, scheduling and routing prob-

lems, goal programming has been applied to a nurse scheduling problem in Saudi Arabia [10]; and to the scheduling of eleven nurses in a hospital ward [166]. A version of the vehicle routing problem using soft time-windows was tackled by Calvete et al. [41] with weighted goal programming. Therefore, goal programming has been widely applied to related problems and using related techniques, making it a sound option for the methodology presented in the following section.

8.2 Proposed Estimation Methodology

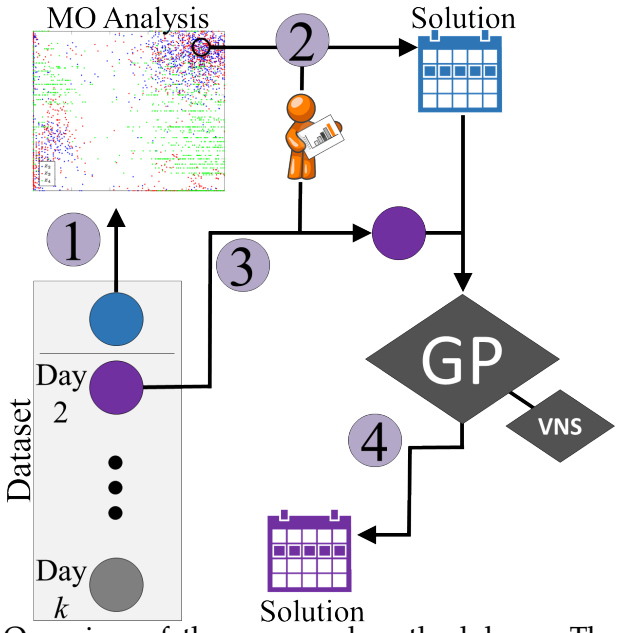


Figure 8.1: Overview of the proposed methodology. The numbered steps are explained below.

Figure 8.1 presents the overall concept of the proposed methodology. Following, we explain each of the four numbered steps indicated in the figure.

1. A *pilot instance* is selected within the dataset and solved using multiobjective algorithms to obtain the best possible non-dominated approximation set.

2. The decision-maker chooses an appropriate solution from the non-dominated set provided. This chosen solution is knows as the *target solution* and its objective vector is denoted by

$$\vec{Z}^t = (Z_1^t, Z_2^t, Z_3^t, Z_4^t, Z_5^t).$$

3. Each remaining instance in the dataset may now be solved by the VNS proposed in Chapter 7 using a modified objective function (goal programming variant) that attempts to reach the target solution in the current dataset.

4. The final solution obtained by the modified VNS is presented.

The modified objective function of step 3 has an important role in the methodology. Following, we present three approaches for determining the objective function. The first is the well known Chebyshev approach aforementioned. The second aims to derive a weight-vector from the target solution and the approximation set of the pilot instance. The third approach minimises the euclidian distances to the target objective-vector.

8.2.1 Chebyshev Goal Programming

Chebyshev goal programming [79] is one of the most widely employed goal programming techniques that does not necessarily rely on the decision maker to set priorities between objectives [117]. The technique aims to obtain a balanced solution by minimising the gap to the target of the objective that presents the highest gap. Hence, if the target objectives are similarly difficult to be reached, the Chebychev GP technique can obtain a balanced solution. However, if at least one target objective is more dif-

ficult to be achieved (i.e. the target goal is too optimistic), the quality of that objective can be a bottleneck for the remaining objectives because the search will solely focus on improving that objective. We define the Chebyshev objective function for the HHC problem as follows:

$$\text{Minimise} \quad \underline{\lambda} \tag{8.1}$$

Subject to

$$\frac{Z_1}{Z_1^t} \leq \underline{\lambda} \tag{8.2}$$

$$\frac{Z_2}{Z_2^t} \leq \underline{\lambda} \tag{8.3}$$

$$\frac{Z_3}{Z_3^t} \leq \underline{\lambda} \tag{8.4}$$

$$\frac{Z_4}{Z_4^t} \leq \underline{\lambda} \tag{8.5}$$

$$\frac{Z_5}{Z_5^t} \leq \underline{\lambda} \tag{8.6}$$

The Chebyshev objective function (Equation 8.1) replaces the objective function of the HHC problem presented in Equation 3.18 and the constraints presented in equations 8.2 – 8.6 are added to the model. The main objective is now to minimise $\underline{\lambda}$, thus finding a well balanced solution regarding reaching the target values. If all targets are reached, $\underline{\lambda}$ can assume fractional values and a solution that shows balanced improvements on all objectives may be obtained.

8.2.2 Derived Weight Vector

One problem with the Chebyshev approach is that it does not guarantee Pareto efficiency. However, the optimal solution of a weighted sum (where weights are not simultaneously null) objective function is always

Pareto efficient. To derive a weight vector from the target solution, we first convert the approximation set of the pilot instance into a system of linear inequalities. Considering that the approximation set is composed of $\overline{n}$ objective-vectors ($\vec{Z}^1, \vec{Z}^2, \ldots, \vec{Z}^{\overline{n}}$), the linear inequalities system can be defined as:

$$\begin{cases} \vec{w}\vec{Z}^t \leq \vec{w}\vec{Z}^1 \\ \vec{w}\vec{Z}^t \leq \vec{w}\vec{Z}^2 \\ \vdots \\ \vec{w}\vec{Z}^t \leq \vec{w}\vec{Z}^{\overline{n}} \end{cases} \tag{8.7}$$

where the aim is to find the values of $\vec{w} = (w_1, w_2, w_3, w_4, w_5)$.

There is no guarantee that the system of linear inequalities has a solution because the fitness landscape is non-convex, meaning no set of weights can achieve some given points in the front. Therefore, instead of finding a solution for the system, we aim to find a weight vector $\vec{w}$ that satisfies the largest number of inequalities. Hence, we define the problem of finding the best weight vector as the following MIP minimisation problem.

$$\text{Minimise} \qquad \sum_{i=1}^{\overline{n}} \overline{x}_i \tag{8.8}$$

subject to

$$\vec{w}\vec{Z^t} - \vec{w}\vec{Z^i} \leq M\overline{x}_i \qquad , i = 1, \ldots, 5 \tag{8.9}$$

$$w_i \in (0, 1], \overline{x}_i \ \text{ binary} \qquad , i = 1, \ldots, 5 \tag{8.10}$$

Note that M is a very large constant.

The objective function (Equation 8.8) aims to find a weight vector $\vec{w}$ that minimises the number of linear equations in the system (Equations 8.7) that does not fulfill the condition $\vec{w}\vec{Z^t} \leq \vec{w}\vec{Z^i}$ (Constraint 8.9). Note

that Constraint 8.10 guarantees that zero cannot be chosen as a weight-value as we don't want criteria to be removed.

Finally, the weight vector $\vec{w}$ obtained from the MIP model replaces the weights of the original objective function of the HHC, defined by Equation 3.18 (Chapter 3). The new objective function is shown below (Equation 8.11).

$$\begin{aligned}\text{Minimize} \quad & \sum_{k\in K}\sum_{i\in V^S}\sum_{j\in V^N}\left(w_1 d_{i,j} + w_2 p_j^k\right)x_{i,j}^k \\ & + \sum_{j\in T}\left(3r_j - \sum_{i\in V^S}\sum_{k\in K}\left(w_3\rho'_{k,j} + w_4\rho''_{k,j} + w_5\rho'''_{k,j}\right)x_{i,j}^k\right) + \lambda_4\sum_{j\in T} y_j \quad (8.11)\end{aligned}$$

Note that λ_4 still has absolute precedence over the other terms.

8.2.3 Euclidian Distances

We propose an alternative based on the euclidian distances to the target vector. In essence, this is a lexicographic method where all objectives are equally important. Hence, minimising the euclidian distances alone does not guarantee Pareto efficiency. In order to reduce this issue, the proposed method consists of minimising the distances to the target vector for the objectives that are worse than the target. If the current distance of the objectives that are worse than the target vector is small ($< \bar{\epsilon}$), then the aim is to maximise the distances of the objectives that are better than the target vector.

Henceforth, the objective function of the HHC problem (Equation 3.18) is replaced by the objective function presented in Equation 8.12.

In summary, when the euclidian distances of the objectives that are worse than the target vector are larger than the given parameter $\underline{\epsilon}$, the

$$\text{Minimize} \quad \begin{cases} \lambda_4 \sum_{j \in T} y_j + \underline{z} & \text{, if } \ \underline{z} > \overline{\epsilon} \\ \lambda_4 \sum_{j \in T} y_j - \underline{z}' & \text{, otherwise} \end{cases} \tag{8.12}$$

where

$$\underline{z} = \sqrt{\underline{z}_1 + \underline{z}_2 + \underline{z}_3 + \underline{z}_4 + \underline{z}_5} \tag{8.13}$$

$$\underline{z}' = \sqrt{\underline{z}'_1 + \underline{z}'_2 + \underline{z}'_3 + \underline{z}'_4 + \underline{z}'_5} \tag{8.14}$$

$$\underline{z}_i = \begin{cases} (Z_i - Z_i^t)^2 & \text{, if } \ Z_i > Z_i^t \\ 0 & \text{, otherwise} \end{cases} \tag{8.15}$$

$$\underline{z}'_i = \begin{cases} (Z_i - Z_i^t)^2 & \text{, if } \ Z_i \leq Z_i^t \\ 0 & \text{, otherwise} \end{cases} \tag{8.16}$$

objective function consists of minimising the number of unassigned visits (λ_4 term) and the euclidian distances themselves ($\underline{z}$). Otherwise, when $\underline{z} \leq \underline{\epsilon}$, the objective consists of minimising the number of unassigned visits and maximising the distances for the objectives that are better than the target solution. Thus, if the solution has not reached the target, the objective function attempts to close the gap to the target. If the solution is close or better than the target, the objective function attempts to further improve it.

8.3 Experimental Configuration

We applied the proposed methodology to the HHC datasets presented in Chapter 3. The first instance of each dataset (A-01, B-01, C-01, D-01, E-01 and F-01) was selected as the pilot instance of its respective dataset. We used the approximation sets obtained from the experiments presented in Chapter 6. Fifteen target vectors were randomly selected from each approximation set and the same target vectors were used for the Esti-

mated Weight Vector (WV) objective function, the Euclidian Distances (ED) objective function, and the Chebyshev (CV) objective function.

For each remaining instance in each dataset, the Full-VNS algorithm proposed in Chapter 7 was applied to the fifteen selected target vectors from the pilot instance of its respective dataset using all three objective functions. For each target vector of each instance, we ran the Full-VNS algorithm eight times for one minute each. The results shown comprise the aggregate data.

8.4 Results

First, we show the effectiveness of the estimated weight vector obtained from the MIP model presented in Equations 8.8 – 8.10. The effectiveness of a weight vector $\vec{w}$ is the percentage of solutions (in the pilot instance approximation set) in which $\vec{w}\vec{Z}^t \leq \vec{w}\vec{Z}^i$. Hence, if the effectiveness is 100%, it means that the MIP model found a solution for the inequalities system presented in Equation 8.7.

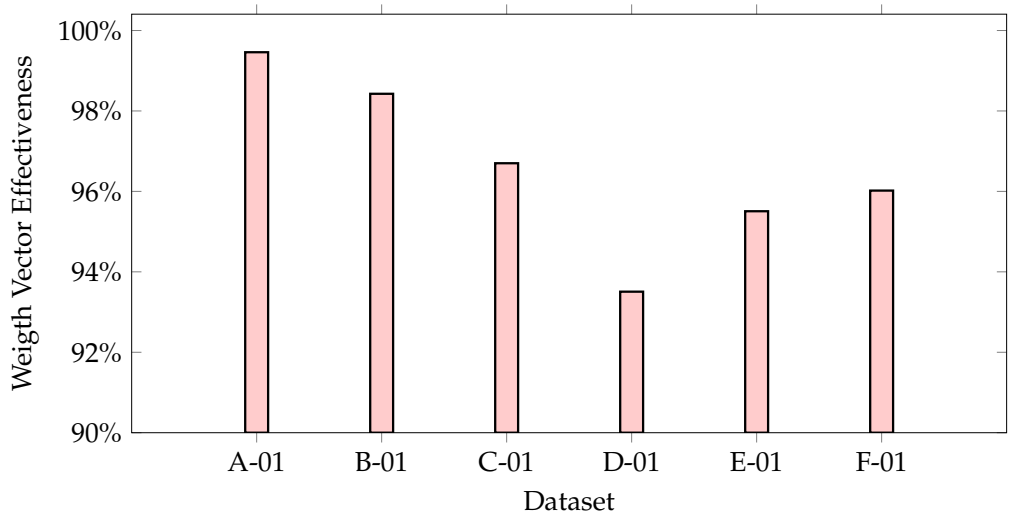


Figure 8.2: Average percentage of the solutions in the approximation set of each target instance such that $\vec{w}\vec{Z}^t \leq \vec{w}\vec{Z}^i$.

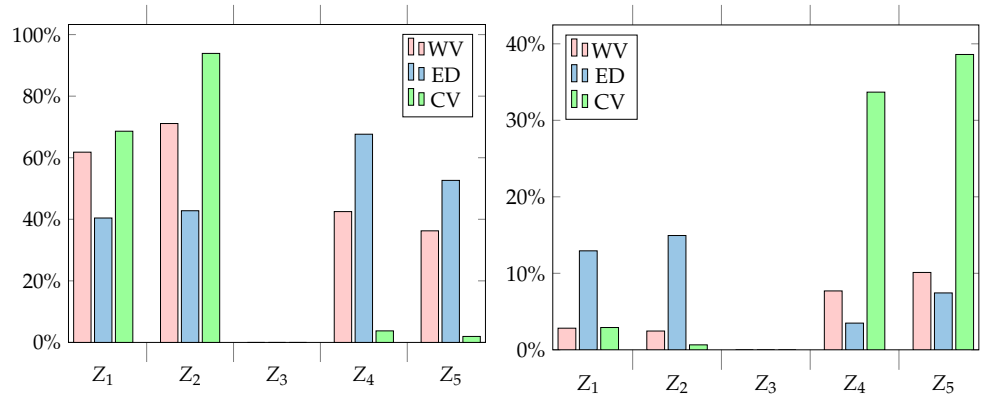
Figure 8.2 presents the results of the effectiveness analysis of the weight vectors obtained. In all pilot instances, the effectiveness surpassed 90% on average. Pilot instance D-01 presented the worst overall effectiveness, with an average of 93.5%, while the pilot instance A-01 presented the most effective weight vectors averaging 99.5%. In all cases, the MIP model provided good weight vectors and these were applied to the WV objective function presented in Equation 8.8.

Next, we show the results for each dataset in three graphs. The first, entitled *target achiement*, displays the percentage of solutions, in the given dataset, that achieved the target value in each objective. The second, entitled *gap to target* contains the average gap to the target solutions for the solutions that did not reach the target. Finally, the *overall comparison* graph displays the average quality of solutions where positive values indicate that, on average, the solutions found are better than the target solution and negative values indicate that the solutions are worse than the target solution.

Figure 8.3 presents the results of applying the Full-VNS algorithm with the WV, ED and CV objective functions to the remaining instances of dataset A. Results comprise the average values of eight runs for each target vector of each problem instance for each objective function.

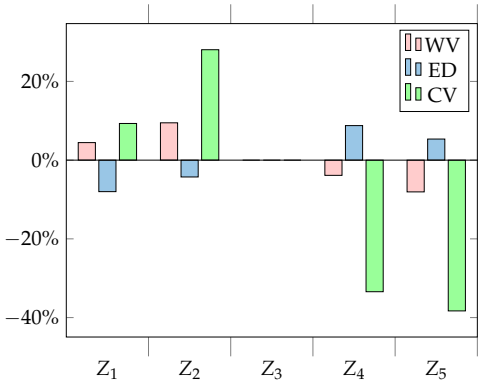
In Figure 8.3a, the percentage of achievement of individual target objective is shown. Note that there are no values for Z_3 because dataset A does not present preferred skills information. It is clear that the CV objective function presented better overall target achievement on Z_1 and Z_2, followed by WV, while the ED objective function is better on Z_4 and Z_5, whilst CV clearly underperforms.

Figure 8.3b shows that the average gap of the solutions that has not reached the target is always smaller than 16% for WV and ED, but high



(a) Target achievement.

(b) Gap to the target.

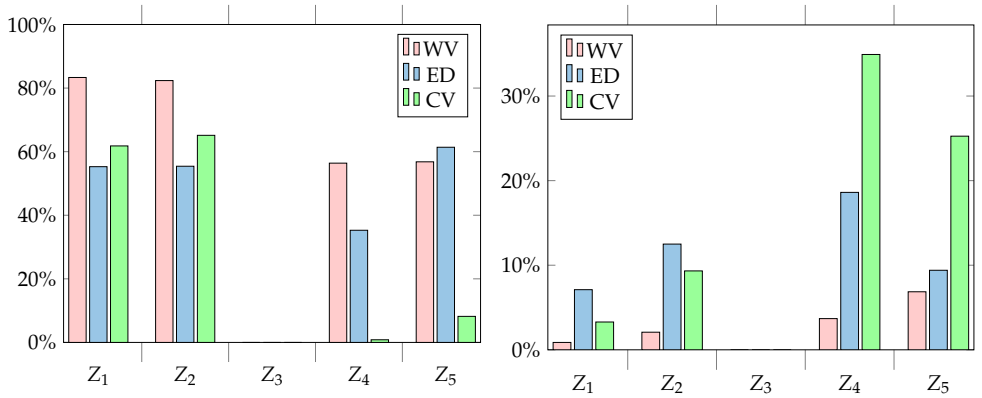


(c) Overall comparison.

Figure 8.3: Summary of results for Dataset A.

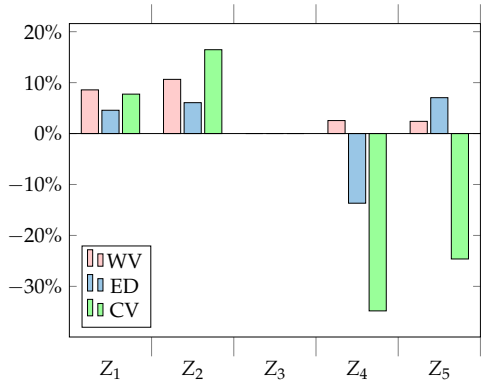
for CV. Also, in conformity with the previous graph, the CV objective function presented better values for the objectives Z_1 and Z_2 while the ED objective function obtained better values for Z_4 and Z_5. The CV objective function presented large gaps on the two latter objectives. Finally, in Figure 8.3c, it is clear that on average solutions are at most 10% off the target for WV and ED, but the CV approach clearly presents a stronger bias towards Z_1 and Z_2 causing the quality of Z_4 and Z_5 to deteriorate.

The results for dataset B are presented by Figure 8.4. In Figure 8.4a, it is clear that the WV objective function obtains better target achievement than the ED on Z_1, Z_2 and Z_3. On Z_4 the ED has a small advantage. Also, when compared to dataset A, the overall achievement of the targets is higher. The CV objective function shows competitive results on Z_1 and Z_2, but presents extremely low achievement rates on Z_4 and Z_5.



(a) Target achievement.

(b) Gap to the target.

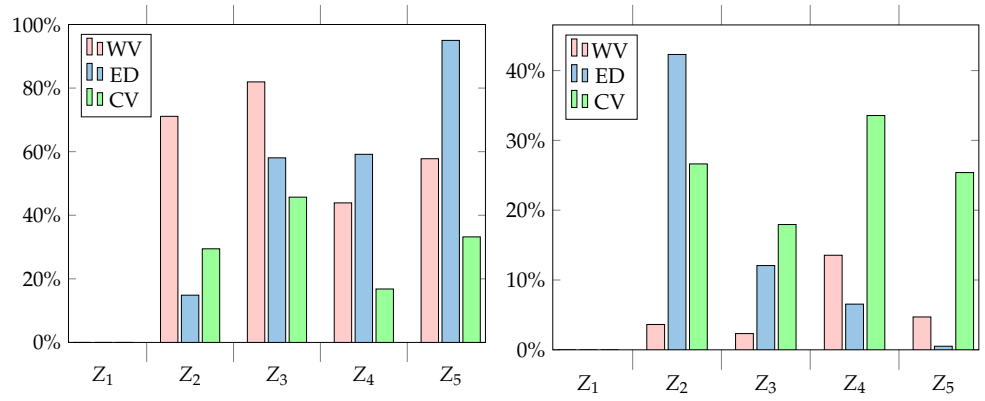


(c) Overall comparison.

Figure 8.4: Summary of results for Dataset B.

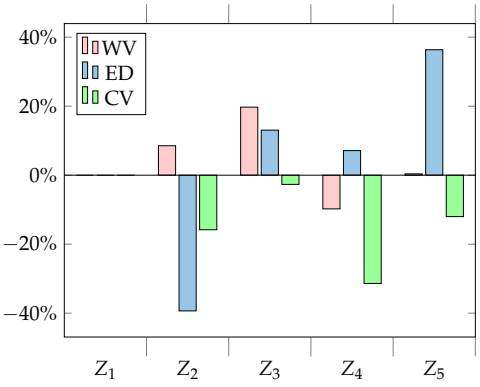
Figure 8.4b shows that on all objectives, the WV objective function obtained smaller average gaps for the solutions that have not met the target. Conversely, the CV objective function presents large gaps on Z_4 and Z_5. Figure 8.4c displays that on average, the objective function WV can provide improved objective-values on all objectives. However, the objective function ED presents improved results on Z_1, Z_2 and Z_5 only, failing to deliver good results on Z_4, while CV presents a similar behaviour as on dataset A: good performance on Z_1 and Z_2 but bad performance on Z_4 and Z_5. Thus, in this dataset, the WV objective function presents a better alternative to solve the problem.

Figure 8.5 presents the results for dataset C. Note that because that dataset does not possess distances information, Z_1 is always zero. In Figure 8.5a, it is clear both the objective functions WV and ED are bal-



(a) Target achievement.

(b) Gap to the target.



(c) Overall comparison.

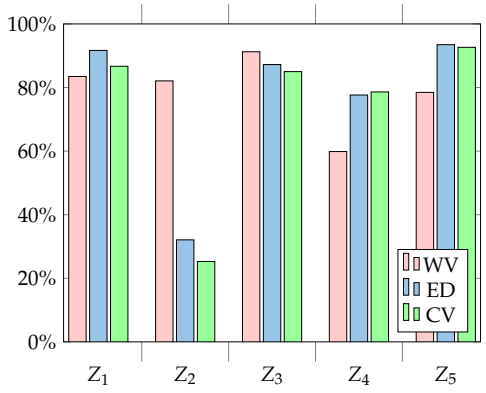
Figure 8.5: Summary of results for Dataset C.

anced, with WV presenting better results for Z_2 and Z_3 while the ED has the edge on Z_4 and Z_5. However, the ED objective function presents deteriorated results on Z_2 and exceptional results on Z_5. The CV objective function underperforms when compared to the other methods.

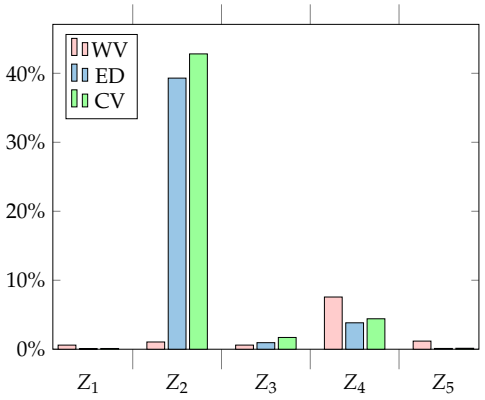
On Figure 8.5b, it is clear that the ED objective function not only struggles to reach the target for Z_2, but it also misses it by 40% on average. This information is also reflected on Figure 8.5c. Overall, the WV objective function presents a more consistent choice for this dataset as only on Z_4 the overall results are negative and the average overall performance is superior. Again, the CV objective function presents the worst results as it can be seen that for every objective there is a deficit in the overall quality obtained.

Also, it is important to notice that this dataset presents a high vari-

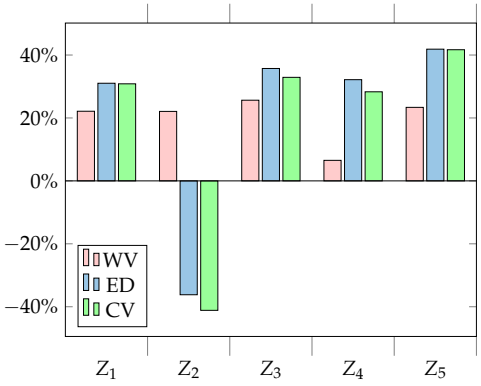
ance in the sizes of the instances. Therefore, because C-01 is a large instance, the target solution simply cannot be achieved on the smaller instances of that dataset (C-02, C-04, C-05 and C-07).



(a) Target achievement.



(b) Gap to the target.



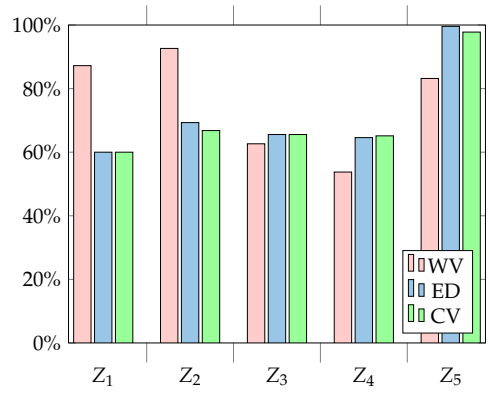
(c) Overall comparison.

Figure 8.6: Summary of results for Dataset D.

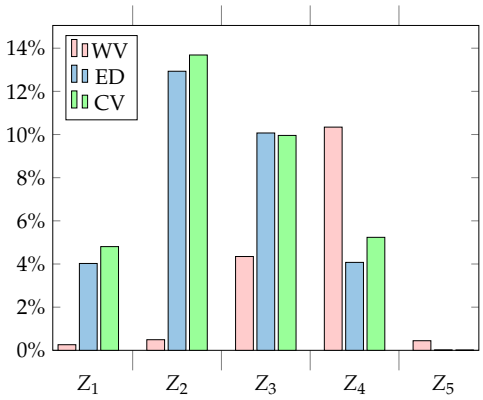
The results for dataset D are presented by Figure 8.6. In Figure 8.6a we see an overall increase on the target hit achievement when compared to the previous datasets (A, B and C). As with dataset C, the ED objective function displayed poor performance for Z_2 but had better performance on three other objectives (Z_1, Z_4 and Z_5). The CV objective function presented competitive results on this dataset.

Figure 8.6b shows that, except for the ED and CV objective functions on Z_2, the gap for the solutions that have not met the target is small. Also, except for that objective, the overall average results (Figure 8.6c) show that all three objective functions present good results, with the ED

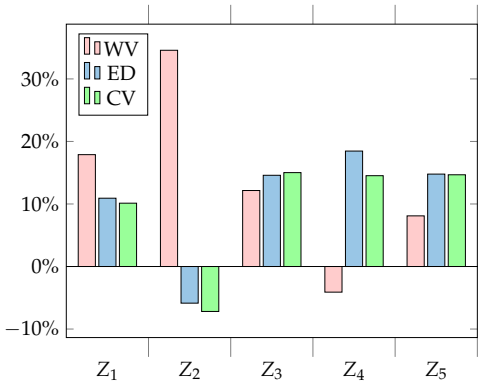
surpassing the WV on all objectives except on Z_2 and the CV presenting results slightly worse than the ED. Still, the WV objective function manages to present a higher consistency than the other objective functions, and it also manages to present good results for all objectives.



(a) Target achievement.



(b) Gap to the target.

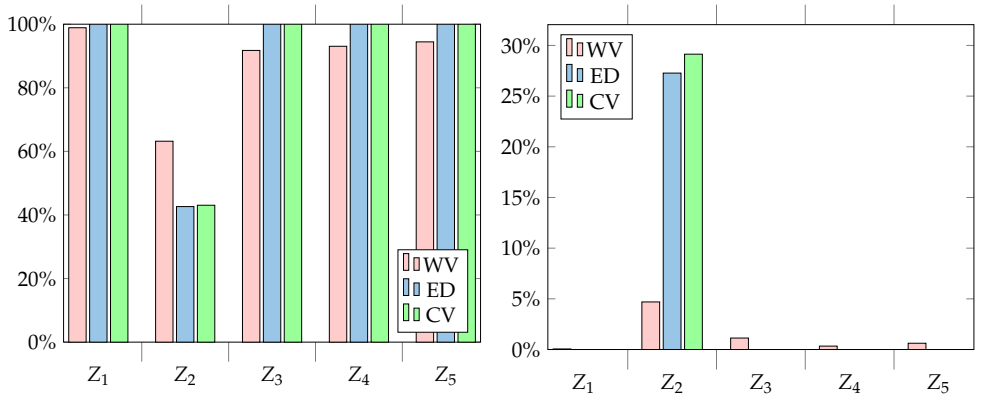


(c) Overall comparison.

Figure 8.7: Summary of results for Dataset E.

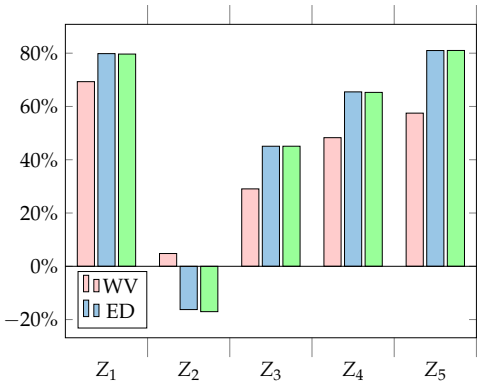
Dataset E has its results presented in Figure 8.7. Like with dataset D, a high overall target achievement rate is shown (Figure 8.7a). The WV objective function present exceptional results on Z_1 and Z_2, but fail to beat the other objective functions on the remaining objectives. On Figure 8.7b it is clear that the ED and CV objective functions present higher gaps to the target solutions. The last figure (Figure 8.7c) shows that all objective functions perform well on this dataset. The WV objective function presents better performance on Z_1 and Z_2, while the ED objective function presents better results on Z_4. The CV objective function presents the

best results on Z_3.



(a) Target achievement.

(b) Gap to the target.



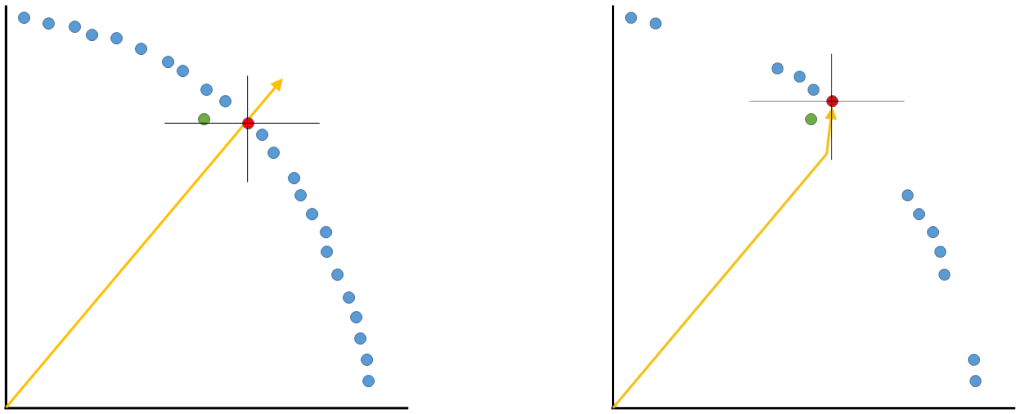
(c) Overall comparison.

Figure 8.8: Summary of results for Dataset F.

Lastly, the results for dataset F are presented in Figure 8.8. The target achievement rates (Figure 8.8a) show that all three objective functions perform well on this dataset, however, on Z_2 the WV objective function is noticeably better. Figure 8.8b reinforces that claim as the gap for objective Z_2 is much larger for the ED and CV objective functions. Finally, Figure 8.8c shows that on average, the solutions provided by the WV objective function are better than the target solutions for all objectives, and, while the ED and CV objective functions present better results than the WV on Z_1, Z_3, Z_4 and Z_5, on Z_2 the average results are worse than the target values.

8.5 Discussion

It was shown that both the WV and ED objective functions provide good results on the majority of the scenarios, with the WV objective function being slightly better than the ED objective function on average. Also, for the first three scenarios, the CV objective function clearly presented inferior results. It is noticeable that the results were better on the largest datasets D, E and F. In Chapter 8 we showed that the smallest datasets are also the ones with solutions in all regions of the objective space and that they resemble the independent MOMKP set. Also, it was shown that datasets D, E and F present unique fitness landscapes with several regions containing no solutions and several local relationships. Therefore, in the smallest datasets, if the objective function is not accurate enough (regarding reaching the target solution), the search can deviate from the region of interest more easily because the objective space is smoother. On the largest datasets, because the objective space contains gaps and local relationships, it may be enough for the search (driven by the objective function) to reach the same region as the target solution.



(a) Well-spread Pareto front. **(b)** Pareto front with gaps.

Figure 8.9: Example of an inaccurate objective-function on a Pareto front that is well-spread (a) and on a Pareto front with gaps (b).

Take for instance the example presented in Figure 8.9. The blue dots

represent the Pareto front, the green dot is the target solution, the red dot is the solution found by the search and the yellow arrow is the search direction given by the objective function. Note that the search direction is not accurate, hence it does not point towards the target solution (this is a valid assumption given that we are either estimating the weight vector or approximating the euclidian distances). In Figure 8.9a, we have a well spread Pareto front and because there exists a solution aligned with the search vector (red dot), that solution is taken as the best solution. In the given example, the chosen solution is better than the target regarding the objective presented in the x axis, but it is worse than the target regarding the objective represented by the y axis.

In Figure 8.9b, we have a Pareto front with gaps and local relationships. The search direction is the same as before but in that direction, there are no solutions, hence the search seeks a closer solution that, in this case, is one that dominates the target solution (surpasses the target in all objectives).

A reason for the ED objective function to underperform when compared to the WV is the different ranges of objective-values. Distances and costs can range from about a few thousand units while the preferences are measured in dozens of units. This would lead us to believe that the ED objective function would provide better results on objectives Z_1 and Z_2, because those are the objectives with high ranges, but that does not happen, and the algorithm frequently provides better results on Z_3, Z_4 and Z_5. An explanation for this phenomenon lies in the Equation 8.15. When the search reaches local optima, it becomes very difficult for the algorithm to leave it because of the condition of that equation. If the search reaches local optima and a few (but not all) objectives fulfil the condition $Z_i > Z_i^t$, then slightly 'disturbing' the solution to escape local

optima might outcome in other objectives fulfilling the condition and making the new solution much worse. In summary: the VNS proposed in Chapter 7 was tailored for the HHC using a weighted function. The ED objective function drastically changes the objective space and the algorithm loses performance. The phenomenon is even worse for the CV objective function.

Nonetheless, it is clear that estimating a Pareto front of an unknown instance of a dataset based on a known instance is an effective way to tackle the problem. While the multiobjective algorithms required up to an hour to obtain the approximation set of the first instance of a dataset, the VNS managed to find competitive solutions in a minute. In the majority of the experiments, targets were achieved and the overall quality of results was positive.

Still, based on this investigation, it is safe to assume that although the results are good, it is possible to improve the objective functions. On future work, we aim to investigate adaptive objective functions that can change the search direction according to the current state of the search and possibly reach improved results.

8.6 Conclusion

This part of the thesis concludes the studies on the HHC problem. With the multiobjective analysis of the HHC problem (Chapter 6), we discovered that instances of the same dataset possess similar fitness landscapes. In this chapter we proposed a solving methodology to use the efficient single-objective VNS algorithm presented in Chapter 7 to solve the multiobjective HHC problem. The proposal consists of solving an instance of the HHC problem using multiobjective techniques to obtain an approx-

imation set and then having the decision-maker to choose an adequate solution and use goal programming to solve unknown instances of the same dataset using the selected instance as the target. Additionally, we proposed two different objective functions to guide the VNS to reach the target solution. The first attempts to obtain a weight vector of the target solution. The second uses euclidian distances to the target solution.

We showed that the methodology is capable of finding good solutions in a fraction of the time required by multiobjective algorithms to process the approximation set. Also, because of the extra performance of the VNS, it was common for solutions to provide better values than the target solution.

The fact that instances of the same dataset presented similar fitness landscapes in the HHC problem should not come as a surprise because many features are recurring in home healthcare. No drastic changes in the pool of workers occur on different days and many visits are recurring: patients need daily help to get out of bed, take baths, take medications, etc. Therefore, different instances share fundamental parts of the problem, which is one of the reasons of why the fitness landscapes are closely related on different days.

Moreover, the proposed methodology can be extended to any real world problems where instances present similar fitness landscapes, and, although this seems restrictive, many real world problems share features across different instances, increasing the probability of that happening. For instance, scheduling problems can have recurring activities, the same pool of workers every day, etc; vehicle routing problems may have the same fleet every day and recurring deliveries. Hence, it is realistic to expect that on applied problems the proposed technique might be useful and that the multiobjective analysis technique proposed on Chap-

ter 5 is an effective tool to evaluate whether using goal programming to solve the problem is applicable or not.

Chapter 9

Conclusions

In this thesis, a set of problem instances for the combined optimisation problem of scheduling and routing workforce for the home healthcare sector is proposed and studied. The home healthcare scheduling and routing (HHC) optimisation problem considers creating daily work plans for workers (care workers, nurses and doctors) to deliver care to patients in their houses. Each patient has its own requirements and not all workers are qualified to perform all visits (e.g. only nurses can apply intravenous medication, etc.), hence assigning workers with the right set of skills to perform a visit is mandatory. Each visit must be performed at a strict time, and workers must arrive at the patient's premises without delays. Also, it is not always possible to visit all patients. In this case, home healthcare providers have to outsource the visits to third parties, hence keeping the number of unassigned visits to a minimum is the top priority objective. Some home healthcare companies operate in large areas, hence their workers have areas in which they may not be available. Also, workers have a time availability defining the times in which they are available to work. Although workers' area and time availabilities are important and should be respected, it is acceptable to violate

this restriction in order to fulfil a greater number of visits. Preferences are also considered in the problem. Each patient may have a list of preferred workers that are usually the workers who more often visit that patient. Also, each patient may require a set of preferred skills, or 'features', that they want the visiting worker to have. These preferred skills could be requirements such as minor optional qualifications and certificates, or personal preferences such as the requirement for non-smokers or female workers. Each worker has a maximum number of hours they can work in a week, that must be respected at all times. Also, different workers may present different incurred costs to perform the same visit (e.g. a nurse may be able to perform the visits that would require a care worker, but her hourly rate is usually higher), hence maintaining the incurred costs to a minimum is an objective of the problem. Finally, reducing distances not only means a reduction of expenses of mileage costs, but also reduces the overall travel time and improves the quality of work. Hence, the HHC problem combines features from both worker scheduling and vehicle routing problem, both well know NP-Hard combinatorial optimisation problems.

9.1 Review of Contributions

We now revisit the main contributions of this thesis.

9.1.1 Real World HHC Problem Datasets

Six HHC problem datasets arising from different real-world home healthcare companies were presented and publicly made available for the research community. Analysis of the datasets showed that each one presents distinctive characteristics, not only regarding size (number of visits and

workers) but also regarding constraints and objectives. For instance, in datasets A, B, D and F the routing part of the problem is more challenging than in the other datasets; in dataset C, the reduction of incurred costs is more challenging than in the other datasets; dataset E presents a scenario where the biggest challenge is the preferences matching, as it is not hard to find solutions that cover all visits and present a small number of time and area availabilities violations.

9.1.2 Lower Bounds for the HHC Problem

This thesis presents a technique to calculate the lower bounds for the HHC problem that separates the routing and scheduling constraints to form two subproblems. Each subproblem is relaxed and solved by a tailored method. To obtain the lower bound values, a tailored exact algorithm that is capable of finding the optimal solution of the relaxed scheduling subproblem is proposed. Then, to improve the lower bounds a conflict graph is created to estimate the number of unassigned visits in the optimal solution of the HHC problem. A similar technique is used to better estimate the number of time and area availabilities violations. A graph algorithm is used to provide lower bound values for the routing subproblem.

To the best of our knowledge, the techniques proposed in this thesis to obtain lower bounds for the HHC problem are not common in the literature. Relaxing problems to obtain lower bounds is a common practice, but it is usually followed by the use of established exact methods. The contributions of this work consist of splitting the main scheduling and routing problem and separately relaxing and tackling the two versions. The algorithms themselves inspire a similar approach on related problems, such as scheduling, assignment or timetabling. The proposed

technique is particularly useful for providing lower bounds when the size of the data is so large that MIP solvers are unable to load the problem into memory.

9.1.3 Analysis of Objectives Relationships in Multiobjective Problems Using Trade-Off Region Maps

We also presented a method to analyse and visualise the structure of the fitness landscape for many-objective problems, providing researchers with additional tools to aid the understanding of the structure and trade-offs in a set of solutions which approximate the Pareto optimal front. It has been identified that a specific issue is the locality of relationships between solutions, such that these may not hold globally. But, identifying the fact that such relationships are present in specific areas of the objective space can be valuable for the development of solution methods that could find solutions that are acceptable to the practitioners. Indeed, it may be the case that a solution method which attempts to find an improved approximation to the Pareto optimal front may benefit from a design which takes into account the fact that objectives may be positively or negatively correlated in different areas of the objective space, perhaps even changing its objective function accordingly.

In addition to a demonstration of the capabilities of the proposed analysis technique, this thesis gives insights into the generation of multi-objective optimisation problems' benchmark scenarios. It was seen that constraint setup variance, commonly used in the literature, has limited impact on the fitness landscapes, hence relying solely on that may not be sufficient to generate varied scenarios. On the other hand, using data dependency to generate the instances was able to achieve higher diver-

sity and obtain truly unique fitness landscapes, as was shown with the multiobjective multidimensional knapsack problem.

Moreover, the analysis of the HHC datasets provided us with useful insights. Clearly, even though all datasets are equally defined and equally presented in a data format, their fitness landscapes greatly differ. Some datasets (A and B) presented solutions throughout a large region of the objective space, while other datasets presented large areas without any solution. Several local relationships could be spotted as well as trade-offs involving multiple variables. Additionally, instances from the same dataset present similar fitness landscapes, even if the data represent dates separated by months, meaning that by knowing the multiobjective relationships and characteristics of one instance is enough to understand the entire dataset.

9.1.4 A Variable Neighbourhood Search for the HHC Problem

We presented a variable neighbourhood search (VNS) with three tailored local search structures that are general enough to present a good performance on all datasets. The analysis of the objectives relationships on the datasets led us to believe that search structures based on constraints would not be efficient in all datasets because of their unique multiobjective natures. Therefore two of the proposed search structures are independent of specific constraints. The first, the Priority-Based Search essentially attempts to limit the exploration of the search space to interesting areas, by avoiding assignments that are less likely to be in the optimal solution because of a high penalty cost. The second, the Flat Costs Search, estimates the objective value of all assignments and use these

values instead of the original objective function, hence although it loses precision (the estimation is not 100% accurate), it performs so fast that it allows the search to explore a much larger region of the search space. Finally, the Availability Violations Seeker is the only search neighbourhood that is based on constraints. This is because the constraints that it is based are the two with the highest cost penalty excluding unassigned tasks. The search eliminates time and area availability violations (if possible) and rearranges the solution solving eventual time conflicts.

Two of the proposed search neighbourhoods are independent of constraints. Also, time availability is commonly found in this kind of optimisation problems and it is usually a hard constraint [45] or a highly penalised soft constraint (as presented in this thesis). Hence, the proposed VNS is likely to perform well on different workforce scheduling and routing problems (WSRP) with a similar structure. Also, not only the datasets are challenging because of their unique characteristics and complex objective relationships, some datasets are very large and computationally expensive algorithms will not present satisfactory performance. The proposed techniques are capable of performing well even on very large scenarios. Lastly, the VNS presents the current benchmark results for the studied datasets.

9.1.5 Using VNS on Estimated Pareto Fronts to Solve the Multiobjective HHC Problem

When approaching the problem through a multiobjective perspective, on the majority of the datasets, the multiobjective algorithms (MOA) managed to find competitive results at the cost of a high computational time. However, on the largest datasets, the MOAs failed to provide good ap-

proximation sets. Nonetheless, the objective relationship analysis of the HHC datasets allowed us to exploit an interesting characteristic of the data: that all instances of the same dataset present similar fitness landscapes. Hence, this thesis proposes a methodology to approach multiobjective problems by using computationally expensive MOAs on only a single instance of a dataset, presenting the obtained approximation set to the decision maker as the approximation set of all instances, and using the decision maker's choice, goal programming and the VNS (which can reach the solution much quicker than MOAs) to reach an equivalent solution on the other instances not solved by the MOAs.

This methodology can be applied to any optimisation problem where MOAs are computationally expensive or inefficient (which often occurs on many-objective problems); and when instances of the same dataset present similar fitness landscapes, which we showed is not uncommon, as illustrated by the multiobjective multidimensional knapsack problem, the multiobjective nurse scheduling problem, the multiobjective vehicle routing problem with time-windows and the multiobjective HHC problem studied in this thesis. Particularly, the methodology is likely to be applicable on other WSRPs because part of the data is shared across instances: the staff pool is not likely to change in a short period of time and on many WSRPs visits are recurring – e.g. patients might need medicine, bath, feeding every day at the same time; companies might need security personnel on a regular basis, etc. Therefore, the likelihood that instances of the same dataset present similar fitness landscapes are increased in these problems.

9.1.6 Collaborative Research and Development (R&D) Projects for Optimisation Solutions

Because part of this research required collaboration with an industrial partner, we decided to resort to the software engineering literature seeking answers on how to better approach the collaborative project. We found out that the literature presents few works on the subject, hence we proposed a collaborative framework for the development of optimisation solutions for enterprise systems based on a data-centric approach. This framework consists of a human-readable data model, a data extractor and validator and a solution visualisation platform and repository. The framework can easily be applied to any collaborative R&D environment involving optimisation solutions.

Later, we proposed the use of an application programming interface (API) with improved performance to aid in the development of R&D optimisation solutions. The API combines a set of useful and computationally efficient operations that enables researchers with different backgrounds to make use of efficient implementations. Also, the API integrates a data parser and a solution builder that allows efficient evaluations. Finally, we gathered efficiency programming design patterns from the literature to promote improved data structures implementation.

Although this thesis has its main contributions to the computer science of operations research, the framework and the API are contributions on the borderline between applied operations research and software engineering. As optimisation solutions become more common, especially with the advent of software as a service and cloud computing, collaborative R&D development of optimisation solutions are increasing, hence contributions in this field of research are valuable to the liter-

ature.

9.2 Final Remarks

The HHC problem introduced in this thesis presents some interesting challenges:

- the problem is a highly constrained combination of scheduling and routing and the current MIP model cannot be solved to optimality by mathematical solvers on most instances;
- each dataset presents distinctive characteristics as some constraints and objectives are more or less important than in other datasets, thus domain-specific solutions might not be widely effective; and
- the size of the largest datasets prohibits computationally expensive techniques, even when using heuristic methods.

However, in the current form, the problem bears more similarity to a VRP rather than to a scheduling problem as the majority of the constraints can be found in VRP. Hence, an interesting research direction is to investigate ways to bring new scheduling constraints to the HHC, and in particular to the proposed datasets, in order to differentiate the current problem from the VRP.

The analysis of objective relationships of the HHC datasets gave us key information on how to efficiently tackle the problem. The analysis indicated that each dataset has a distinct objective space and objective relationships. Hence, designing solving techniques based on specific constraints (routing, preferences, costs, etc.) potentially would be effective only on part of the datasets. Then, the usefulness of such specific solving techniques would be restricted to problems where the tack-

led constraints would be similarly evident. The analysis led us to design two constraint-independent local search mechanisms that perform well on all datasets and can be adapted to other related problems. The constraint-dependent local search proposed is based on a constraint that is usually hard or a highly penalised soft constraint, hence it is likely to exist and to be important for related problems.

Yet, the analysis technique for multiobjective problems has been crafted for the HHC problem and tested on benchmark problems only. Questions about how useful the technique can be for other real world problems and applications are unanswered and could constitute an interesting research direction.

The estimation of the objective function used by the local search neighbourhoods proposed, as shown by the results, proved to be an efficient technique to tackle the larger datasets, as it allowed a better exploration of the search space while not significantly compromising the convergence to good solutions. In fact, even with the estimated costs, the algorithms were able to find the optimal solution in the instances where the optimal solution was known.

Finally, the multiobjective analysis showed us that instances of the same dataset present similar fitness landscapes. This led us to propose a solving methodology for the multiobjective HHC problem. We showed that it is more efficient to estimate the Pareto front and employ goal programming with the proposed VNS to solve the multiobjective HHC problem rather than directly apply MOAs. We conjectured that other real world problems might present a similar fitness landscape behaviour, i.e. being consistent across different instances; but we have not investigated this further. Therefore, a proper study on different real world problems to assess whether the proposed solving methodology is appli-

cable or not consists of an interesting research direction.

9.3 Future Work

In the begining of this project when the datasets were obtained, a few decisions were taken that impacted the entire project. Namely, the optimisation of a single day and the lack of consideration of time-windows. The optimisation of a single day was chosen because the literature supported it [43] and because it allowed us a deeper understanding of a daily plan. However, increasing the optimisation time horizon to a week implicates the addition of new constraints and the increased importance of the maximum working hours constraint. Because the number of maximum working hours in a week is limited, deciding which day a worker should work can have a substantial impact on the overall weekly plan. Additionally, working regulations such as rest time and breaks must be managed. Hence, not only the problem becomes substantially larger in terms of number of workers and visits, but it also gains constraints that may change its structure. Investigating this weekly problem should pose an interesting challenge and will be part of future work.

Time-windows commonly appear in the WSRP literature [43, 140]. However, the system used to collect the datasets studied in this thesis leaves the decision of booking the visit times to the user. Castillo-Salazar et al. [45] showed that time-windows add new challenges to the problem, as several synchronisation constraints appear and the decision of visiting times substantially increases the search space. Hence, we aim to use time-windows profiles from the literature to create versions of our datasets with the time-windows constraints. Comparing the multiobjective analysis of the datasets with a version with time-windows could

provide useful insigths about that constraint. Also, Castillo-Salazar et al. [45] proposed a WSRP dataset collected from the literature that includes time-windows. Applying the developed algorithms to these and other datasets could provide an interesting research direction. Adapting the current VNS to efficiently handle time-windows could potentially require the design of new search neighbourhoods and solution structures and make the algorithms applicable to a wider variety of WSRPs.

The proposed algorithm to calculate the lower bounds for the HHC problem uses brute force to identify the cliques in the conflict graph. Although we showed that the current method works, there are more elaborate techniques available in the literature. Krishnamoorthy et al. [134] proposes a polynomial algorithm to obtain maximal cliques for a conflict graph (called interval graph by the author). The application of their algorithm could enable us to investigate larger cliques and possibly obtain improved results for the estimation of unassigned visits and especially for the time and area availabilities violations.

The use of components of the analysis of objectives relationships in multiobjective problems proposed in this thesis, such as the region maps, during the optimisation process to help guide the search towards regions of interest is subject for future work. Also, the investigation of the use of heatmaps to assess whether using these in addition to the current methods could provide additional information on the multiobjective nature of the problem in hand, is also being considered as further future work.

Finally, the methodology presented in this thesis to use an estimated approximation set and goal programming to solve multiobjective problems could improved. The methodology could be further validated using different problems and datasets, such as the multiobjective knapsack, nurse scheduling and vehicle routing problems presented in this thesis. Additionally, other techniques to reach the target solution could be investigated, such as the minimisation of the deviation to the target

objectives and the archimedian goal programming model [155] in which weights are assigned to the deviation of each objective from its target goal.

Bibliography

[1] S. Abdennadher and H. Schlenker. Nurse scheduling using constraint logic programming. In *Proceedings of the 1998 Winter Simulation Conference Proceedings*, pages 838–843. AAAI Press, 1999.

[2] 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. ISSN 03608352.

[3] M. Albareda-Sambola, E. Fernández, and G. Laporte. Heuristic and lower bound for a stochastic location-routing problem. *European Journal of Operational Research*, 179(3):940 – 955, 2007.

[4] H. Algethami, R. L. Pinheiro, and D. Landa-Silva. A genetic algorithm for a workforce scheduling and routing problem. *Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC 2016)*, 2016.

[5] R. K. Anderson and M. Dror. An interactive graphic presentation for multiobjective linear programming. *Applied mathematics and computation*, 123:229–248, 2001.

[6] V. D. Angelis. Planning home assistance for AIDS patients in the City of Rome , Italy. *Interfaces*, 28:75–83, 1998.

[7] E. M. Arkin and E. B. Silverberg. Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics*, 18(1):1 – 8, 1987. ISSN 0166-218X.

[8] P. Augerat, J. M. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi. Computational results with a branch and cut code for the capacitated vehicle routing problem. Technical report, Universite Joseph Fourier, Grenoble, France, 1998.

[9] A. Azab and B. Naderi. Greedy heuristics for distributed job shop problems. *Procedia CIRP*, 20:7 – 12, 2014. ISSN 2212-8271.

[10] M. N. Azaiez and S. S. Al Sharif. A 0-1 goal programming model for nurse scheduling. *Comput. Oper. Res.*, 32(3):491–507, Mar. 2005. ISSN 0305-0548.

[11] D. F. Bacon, P. Cheng, and V. T. Rajan. Controlling fragmentation and space consumption in the metronome, a real-time garbage collector for java. *SIGPLAN Not.*, 38(7):81–92, June 2003. ISSN 0362-1340.

[12] J. Bader and E. Zitzler. Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evol. Comput.*, 19(1):45–76, Mar. 2011.

[13] R. Balachandra and J. Friar. Factors for success in r amp;d projects and new product innovation: a contextual framework. *Engineering Management, IEEE Transactions on*, 44(3):276 – 287, 1997. ISSN 0018-9391.

[14] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2): 351–385, 2008.

[15] S. Bandyopadhyay and R. Bhattacharya. Solving multi-objective parallel machine scheduling problem by a modified nsga-ii. *Applied Mathematical Modelling*, 37(1011):6718 – 6729, 2013. ISSN 0307-904X.

[16] T. Barnes, I. Pashby, and A. Gibbons. Effective university industry interaction:: A multi-case evaluation of collaborative r&d projects. *European Management Journal*, 20:272 – 285, 2002. ISSN 0263-2373.

[17] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1996.

[18] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[19] A. Baykasoglu. Preemptive goal programming using simulated annealing. *Engineering Optimization*, 37(1):49–63, 2005.

[20] C. Bazgan, H. Hugot, and D. Vanderpooten. An efficient implementation for the 0-1 multi-objective knapsack problem. In *Experimental Algorithms*, volume 4525 of *LNCS*, pages 406–419. 2007. ISBN 978-3-540-72844-3.

[21] S. V. Begur, D. M. Miller, and J. R. Weaver. An integrated spatial dss for scheduling and routing home-health-care nurses. *Interfaces*, 27:35–48, 1997.

[22] J. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer Series in Statistics. Springer, 1985. ISBN 9780387960982.

[23] 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. ISSN 0305-0548.

[24] N. Beume, B. Naujoks, and M. Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653 – 1669, 2007. ISSN 0377-2217.

[25] K. Bhattacharya, R. Hull, and J. Su. A data-centric design methodology for business processes. In *Handbook of Research on Business Process Modeling, chapter 23*, pages 503 – 531, 2009.

[26] M. Blais, S. D. Lapierre, and G. Laporte. Solving a home-care districting problem in an urban setting. *Journal of the Operational Research Societyc*, 54(11):1141–1147, 2003. ISSN 0160-5682.

[27] V. Borsani, M. Andrea, B. Giacomo, and S. Francesco. A home care scheduling model for human resources. In *2006 International Conference on Service Systems and Service Management*, pages 449–454, 2006.

[28] J. Branke, K. Deb, and K. Miettinen. *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Genetic algorithms and evolutionary computation. Springer, 2008. ISBN 9783540889076.

[29] J. Branke, K. Deb, K. Miettinen, and R. Slowinski. *Multiobjective Optimization: Interactive and Evolutionary Approaches*. LNCS sublibrary: Theoretical computer science and general issues. Springer, 2008. ISBN 9783540889076.

[30] J. Branke, S. Greco, R. Slowiński, and P. Zielniewicz. Interactive evolutionary multiobjective optimization using robust ordinal regression. In *Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization*, EMO '09, pages 554–568, Berlin, Heidelberg, 2009.

[31] O. Bräysy. A reactive variable neighborhood search for the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 15:347–368, 2003.

[32] D. Bredstrom and M. Rönnqvist. A branch and price algorithm for the combined vehicle routing and scheduling problem with synchronization constraints. *NHH Dept. of Finance & Management Science Discussion Paper No. 2007/7*, 2007.

[33] D. Bredstrom and M. Rönnqvist. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research*, 191(1):19–31, 2008. ISSN 03772217.

[34] D. Brockhoff and E. Zitzler. Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 2086–2093, 2007.

[35] A. E. Brownlee and J. A. Wright. Solution analysis in multiobjective optimization. In *First Building Simulation and Optimization Conference, Loughborough, UK*, 2012.

[36] T. Bui and J. Lee. An agent-based framework for building decision support systems. *Decision Support Systems*, 25(3):225 – 237, 1999. ISSN 0167-9236.

[37] E. Burke, P. De Causmaecker, S. Petrovic, and G. Vanden Berghe. Variable neighborhood search for nurse rostering problems. In

Metaheuristics: Computer Decision-Making, volume 86 of *Applied Optimization*, pages 153–172. Springer US, 2004. ISBN 978-1-4419-5403-9.

[38] E. K. Burke and G. Kendall. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Springer Publishing Company, Incorporated, 2nd edition, 2013. ISBN 1461469392, 9781461469391.

[39] S. Cahon, N. Melab, and E. Talbi. ParadisEO: a framework for the reusable design of parallel and distributed metaheuristics. *Journal of heuristics*, 10:357–380, 2004.

[40] X. Cai, C. J. Goh, and A. I. Mees. Greedy heuristics for rapid scheduling of trains on a single track. *IIE Transactions*, 30(5):481–493, 1998.

[41] H. I. Calvete, C. Galé, M.-J. Oliveros, and B. Sánchez-Valverde. A goal programming approach to vehicle routing problems with soft time windows. *European Journal of Operational Research*, 177(3):1720 – 1733, 2007. ISSN 0377-2217.

[42] J. Castillo-Salazar, D. Landa-Silva, and R. Qu. Workforce scheduling and routing problems: literature survey and computational study. *Annals of Operations Research*, pages 1–29, 2014. ISSN 0254-5330.

[43] J. A. Castillo-Salazar, D. Landa-Silva, and R. Qu. A survey on workforce scheduling and routing problems. In *Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 283–302, Son, Norway, August 2012.

[44] J. A. Castillo-Salazar, D. Landa-Silva, and R. Qu. Computational study for workforce scheduling and routing problems. In *ICORES 2014 - Proceedings of the 3rd International Conference on Operations Research and Enterprise Systems*, pages 434–444, 2014.

[45] 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. ISSN 1572-9338.

[46] J. Castro-Gutierrez, D. Landa-Silva, and P. J. Moreno. Nature of real-world multi-objective vehicle routing with evolutionary algorithms. *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pages 257–264, 2011. ISSN 978-1-4577-0653-0.

[47] J. Castro-Gutierrez, D. Landa-Silva, and J. Moreno Perez. Movrptw dataset (2010). `https://github.com/psxjpc/`, 2015. Accessed: 2015-07-28.

[48] P. Chand and J. R. Mohanty. Article: Solving vehicle routing problem with proposed non- dominated sorting genetic algorithm and comparison with classical evolutionary algorithms. *International Journal of Computer Applications*, 69(26):34–41, May 2013.

[49] A. Charnes and W. Cooper. Goal programming and multiple objective optimizations. *European Journal of Operational Research*, 1(1): 39 – 54, 1977. ISSN 0377-2217.

[50] N. Chase, M. Rademacher, E. Goodman, R. Averill, and R. Sidhu. A benchmark study of multi-objective optimization methods.

[51] E. Cheng and J. L. Rich. A home health care routing and scheduling problem, 1998.

[52] N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20(1):255–282, 1981.

[53] D. Cohn and R. Hull. Business artifacts: A data-centric approach to modelling business operations and processes. In *IEEE Data Engineering Bulletin 32*, 2009.

[54] D. T. Connors. Software development methodologies and traditional and modern information systems. *SIGSOFT Softw. Eng. Notes*, 17(2):43 – 49, Apr. 1992. ISSN 0163-5948.

[55] A. A. Constantino, E. Tozzo, R. L. Pinheiro, D. Landa-Silva, and W. Romão. A variable neighbourhood search for nurse scheduling with balanced preference satisfaction. In *17th International Conference on Enterprise Information Systems (ICEIS 2015)*, Barcelona, Spain, 04/2015 2015. Scitepress, Scitepress.

[56] J.-F. Cordeau, G. Laporte, F. Pasin, and S. Ropke. Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, 13(4):393–409, 2010. ISSN 1094-6136.

[57] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms and Java CD-ROM*. McGraw-Hill Companies,Incorporated, 2003. ISBN 9780072970548.

[58] B. Crevier, J.-F. Cordeau, and G. Laporte. The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*, 176(2):756 – 773, 2007. ISSN 0377-2217.

[59] T. Davidovic, P. Hansen, N. Mladenovic, . Hautes, and . Commerciales. Variable neighborhood search for multiprocessor scheduling problem with communication delays. In *MIC 2001, chairman J. Pinho de Sousa, Proceedings of the 4th Metaheuristics International Conference*, pages 737–741, 2001.

[60] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, Aug 2014. ISSN 1089-778X.

[61] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[62] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 1, pages 825–830, 2002.

[63] K. Deb, M. Mohan, and S. Mishra. A fast multi-objective evolutionary algorithm for finding well-spread pareto-optimal solutions. Technical report, KanGAL Report No. 2003002, 2003.

[64] F. Della-Croce and F. Salassa. A variable neighborhood search based matheuristic for nurse rostering problems. *Annals of Operations Research*, 218(1):185–199, 2014. ISSN 0254-5330.

[65] G. Deng, Z. Li, Y. Zhao, and X. Zeng. *MOEA/D for Energy-Aware Scheduling on Heterogeneous Computing Systems*, pages 94–106. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. ISBN 978-3-662-49014-3.

[66] G. Desaulniers, J. Lavigne, and F. Soumis. Multi-depot vehicle scheduling problems with time windows and waiting costs. *European Journal of Operational Research*, 111(3):479–494, 1998. ISSN 03772217.

[67] 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. ISSN 03050548.

[68] M. Dorigo and T. Stützle. *Ant Colony Optimization*. Bradford Company, Scituate, MA, USA, 2004. ISBN 0262042193.

[69] J. J. Durillo and A. J. Nebro. jMetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42:760–771, 2011.

[70] W. Elgarah, J. Haynes, and J. Courtney. A dialectical methodology for decision support systems design. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, 2002.

[71] M. Emmerich, N. Beume, and B. Naujoks. An emo algorithm using the hypervolume measure as selection criterion. In C. Coello Coello, A. Hernández Aguirre, and E. Zitzler, editors, *Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 62–76. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-24983-2.

[72] L. J. Eshelman. The chc adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In *FOGA'90*, pages 265–283, 1990.

[73] P. Eveborn, P. Flisberg, and M. Rönnqvist. Laps care–an operational system for staff planning of home care. *European Journal of Operational Research*, 171(3):962 – 976, 2006. ISSN 0377-2217.

[74] P. Eveborn, M. Rönnqvist, H. Einarsdottir, M. Eklund, K. Liden, and M. Almroth. Operations research improves quality and efficiency in home care. *Interfaces*, 39(1):18–34, 2009. ISSN 0092-2102.

[75] M. Farina and P. Amato. On the optimal solution definition for many-criteria optimization problems. In *Fuzzy Information Processing Society, 2002. Proceedings. NAFIPS. 2002 Annual Meeting of the North American*, pages 233–238, 2002.

[76] M. Farina and P. Amato. A fuzzy definition of "optimality" for many-criteria optimization problems. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 34(3):315–326, 2004. ISSN 1083-4427.

[77] P. Feliot, J. Bect, and E. Vazquez. A bayesian approach to constrained single- and multi-objective optimization. *Journal of Global Optimization*, pages 1–37, 2016.

[78] M. Fırat and C. A. J. Hurkens. An improved mip-based approach for a multi-skill workforce scheduling problem. *Journal of Scheduling*, 15(3):363–380, 2012.

[79] R. B. Flavell. A new goal programming formulation. *Omega*, 4(6): 731 – 732, 1976. ISSN 0305-0483.

[80] P. J. Fleming, R. C. Purshouse, and R. J. Lygoe. Many-objective optimization: An engineering design perspective. In *Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 14–32. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-24983-2.

[81] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization, 1993.

[82] R. Fontana, A. Geuna, and M. Matt. Factors affecting universityindustry r&d projects: The importance of searching, screening and signalling. *Research Policy*, 35(2):309 – 323, 2006. ISSN 0048-7333.

[83] D. Garrett and D. Dasgupta. Multiobjective landscape analysis and the generalized assignment problem. volume 5313 of *LNCS*, pages 110–124. 2008.

[84] J. D. Garrett. *Multiobjective Fitness Landscape Analysis and the Design of Effective Memetic Algorithms*. PhD thesis, Memphis, TN, USA, 2008.

[85] M. Garza-Fabre, G. Pulido, and C. Coello. Ranking methods for many-objective optimization. In A. Aguirre, R. Borja, and C. Garciá, editors, *MICAI 2009: Advances in Artificial Intelligence*, volume 5845 of *Lecture Notes in Computer Science*, pages 633–645. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-05257-6.

[86] K. Ghoseiri and S. F. Ghannadpour. Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. *Applied Soft Computing*, 10(4):1096 – 1107, 2010. ISSN 1568-4946.

[87] I. Giagkiozis and P. Fleming. Methods for many-objective optimization: An analysis. Research Report No. 1030, November 2012.

[88] I. Giagkiozis and P. Fleming. Pareto front estimation for decision making. *Evolutionary Computation, MIT Press*, Apr 2014.

[89] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533 – 549, 1986. ISSN 0305-0548.

[90] F. Glover. Tabu searchpart i. *ORSA Journal on Computing*, 1(3): 190–206, 1989.

[91] F. Glover and C. McMillan. Applications of integer programming the general employee scheduling problem. an integration of ms and ai. *Computers & Operations Research*, 13(5):563 – 573, 1986.

[92] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Artificial Intelligence. Addison-Wesley Publishing Company, 1989. ISBN 9780201157673.

[93] M. Gong, F. Liu, W. Zhang, L. Jiao, and Q. Zhang. Interactive MOEA/D for multi-objective decision making. In *Proceedings of*

the 13th annual conference on Genetic and evolutionary computation, GECCO '11, pages 721–728, New York, NY, USA, 2011.

[94] P. Groth, A. Loizou, A. J. G. Gray, C. Goble, L. Harland, and S. Pettifer. Api-centric linked data integration: The open PHACTS discovery platform case study. *Web Semantics: Science, Services and Agents on the World Wide Web*, 29:12 – 18, 2014. ISSN 1570-8268. Life Science and e-Science.

[95] M. Grötschel, M. Jünger, and G. Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations Research*, 32(6):1195–1220, 1984.

[96] A. Guarnaschelli, O. Chiotti, and H. E. Salomone. Decision models as software artifacts - bridging the business-software gap for decision support systems. In *ICORES 2013 - Proceedings of the 2nd International Conference on Operations Research and Enterprise Systems, Barcelona, Spain, 16-18 February*, pages 86 – 95, 2013.

[97] X. Guo, Y. Wang, and X. Wang. Using objective clustering for solving many-objective optimization problems. *Mathematical Problems in Engineering*, 2013.

[98] D. Hadka. Moea framework, a java library for multiobjective evolutionary algorithms, 2015. URL `http://www.moeaframework.org/`.

[99] D. Hadka and P. Reed. Borg: An auto-adaptive many-objective evolutionary computing framework. *Evolutionary Computation*, page 31, 2013.

[100] E. Hannan. Nondominance in goal programming. *INFOR: Information Systems and Operational Research*, 18(4):300–309, 1980. doi: 10.1080/03155986.1980.11731801.

[101] M. Haouari and R. M'Hallah. Heuristic algorithms for the two-stage hybrid flowshop problem. *Operations Research Letters*, 21(1): 43 – 53, 1997.

[102] E. Harold and W. Means. *XML in a Nutshell*. In a nutshell. O'Reilly Media, 2009. ISBN 9781449379049.

[103] P. Healy and R. Moll. A new extension of local search applied to the dial-a-ride problem. *European Journal of Operational Research*, 83 (1):83 – 104, 1995. ISSN 0377-2217.

[104] J. L. Hennessy, D. A. Patterson, and K. Asanović. *Computer Architecture: A Quantitative Approach*. Computer Architecture: A Quantitative Approach. Morgan Kaufmann/Elsevier, 2012. ISBN 9780123838728.

[105] F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research, 4th Ed.* Holden-Day, Inc., San Francisco, CA, USA, 1986. ISBN 0816238715.

[106] J. Horn, N. Nafpliotis, and D. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 82–87 vol.1, 1994.

[107] F. Huang. A New Application Programming Interface and a Fortran-like Modeling Language for Evaluating Functions and Specifying Optimization Problems at Runtime. *International Journal of Advanced Computer Science and Applications(IJACSA)*, 3(4), 2012.

[108] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *Evolutionary Computation, IEEE Transactions on*, 10(5):477–506, Oct 2006.

[109] A. Huchzermeier and C. H. Loch. Project management under risk: Using the real options approach to evaluate flexibility in r&d. *Manage. Sci.*, 47:85–101, 2001. ISSN 0025-1909.

[110] C. Hwang, S. Paidy, A. Masud, and K. Yoon. *Multiple Objective Decision Making – Methods and Applications: A State-of-the-Art Survey*. Lecture Notes in Economics and Mathematical Systems. Springer Berlin Heidelberg, 2012. ISBN 9783642455117.

[111] IBM. IBM ILOG ODM Enterprise. http://www-03.ibm.com/software/products/en/ibmilogodmente/, 2013.

[112] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evol. Comput.*, 15(1):1–28, Mar. 2007. ISSN 1063-6560.

[113] H. Ishibuchi, N. Akedo, H. Ohyanagi, and Y. Nojima. Behavior of EMO algorithms on many-objective optimization problems with correlated objectives. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 1465–1472, June 2011.

[114] G. Itabashi, M. Chiba, K. Takahashi, and Y. Kato. A support system for home care service based on multi-agent system. In *Information, Communications and Signal Processing, 2005 Fifth International Conference on*, pages 1052–1056, 2005.

[115] J. Jemai, M. Zekri, and K. Mellouli. *An NSGA-II Algorithm for the Green Vehicle Routing Problem*, pages 37–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-29124-1.

[116] D. Jones and M. Tamiz. *Goal Programming Variants*, pages 11–22. Springer US, Boston, MA, 2010. ISBN 978-1-4419-5771-9.

[117] D. Jones and M. Tamiz. *A Review of Goal Programming*, pages 903–926. Springer New York, New York, NY, 2016. ISBN 978-1-4939-3094-4.

[118] P.-C. Kanellakis and C. H. Papadimitriou. Local search for the asymmetric traveling salesman problem. *Operations Research*, 28 (5):1086–1099, 1980.

[119] M. Karnaugh. The Map Method for Synthesis of Combinational Logic Circuits. *Trans. AIEE. pt. I*, 72(9):593–599, 1953.

[120] R. M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972. ISBN 978-1-4684-2001-2.

[121] A. P. Karpenko, D. T. Mukhlisullina, and V. A. Ovchinnikov. Multicriteria optimization based on neural network approximation of decision makers utility function. *Optical Memory and Neural Networks*, 19(3):227–236, 2010.

[122] A. P. Karpenko, D. A. Moor, and D. T. Mukhlisullina. Multicriteria optimization based on neural network, fuzzy and neuro-fuzzy

approximation of decision makers utility function. *Optical Memory and Neural Networks*, 21(1):1–10, 2012.

[123] R. Kashyap and D. P. Vidyarthi. Security driven scheduling model for computational grid using nsga-ii. *Journal of Grid Computing*, 11 (4):721–734, 2013. ISSN 1572-9184.

[124] A. Kaushik and D. P. Vidyarthi. An energy-efficient reliable grid scheduling model using nsga-ii. *Engineering with Computers*, 32(3): 355–376, 2016. ISSN 1435-5663.

[125] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30 (1/2):81–93, 1938.

[126] M. Khabzaoui, C. Dhaenens, and E.-G. Talbi. A multicriteria genetic algorithm to analyze DNA microarray data. In *Evolutionary Computation (CEC), IEEE Congress on*, pages 1874–1881, 2004.

[127] V. V. Kindratenko, J. J. Enos, G. Shi, M. T. Showerman, G. W. Arnold, J. E. Stone, J. C. Phillips, and W.-M. Hwu. Gpu clusters for high-performance computing. In *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, pages 1–8, Aug 2009.

[128] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. ISSN 0036-8075.

[129] J. Knowles. Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

[130] J. Knowles and D. Corne. Towards Landscape Analyses to Inform the Design of Hybrid Local Search for the Multiobjective Quadratic Assignment Problem. In *Soft Computing Systems: Design, Management and Applications*, pages 271–279, Amsterdam, 2002.

[131] A. Konstantinidis, S. Pericleous, and C. Charalambous. *MOEA/D for a Tri-objective Vehicle Routing Problem*, pages 131–140. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-41142-7.

[132] P. Korhonen and J. Wallenius. *Visualization in the Multiple Objective Decision-Making Framework*, pages 195–212. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-88908-3.

[133] J. Kornbluth. A survey of goal programming. *Omega*, 1(2):193 – 205, 1973. ISSN 0305-0483.

[134] M. Krishnamoorthy, A. Ernst, and D. Baatar. Algorithms for large scale shift minimisation personnel task scheduling problems. *European Journal of Operational Research*, 219(1):34 – 48, 2012. ISSN 0377-2217.

[135] P. Kruchten. *The Rational Unified Process: An Introduction*. The Addison-Wesley object technology series. Addison-Wesley, 2004. ISBN 9780321197702.

[136] W. Laesanklang and D. Landa-Silva. Decomposition techniques with mixed integer programming and heuristics to solve workforce scheduling and routing problems. Submitted, 2015.

[137] W. Laesanklang, D. Landa-Silva, and J. A. Castillo-Salazar. Mixed integer programming with decomposition to solve a workforce scheduling and routing problem. In *Proceedings of the International Conference on Operations Research and Enterprise Systems*, pages 283–293, 2015. ISBN 978-989-758-075-8.

[138] W. Laesanklang, D. Landa-Silva, and J. A. C. Salazar. Mixed integer programming with decomposition to solve a workforce scheduling and routing problem. In *Proceedings of the International Conference on Operations Research and Enterprise Systems*, pages 283–293, 2015. ISBN 978-989-758-075-8.

[139] W. Laesanklang, R. L. 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, *Operations Research and Enterprise Systems*, volume 577 of *Communications in Computer and Information Science*, pages 191–211. Springer International Publishing, 2015. ISBN 978-3-319-27679-3.

[140] 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 5th the International Conference on Operations Research and Enterprise Systems*, pages 330–339, 2016.

[141] G. Laporte. The vehicle routing problem : An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59:345–358, 1992.

[142] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10:2002, 2002.

[143] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966. ISSN 0030364X, 15265463.

[144] K. N. Le. *A Study of Evolutionary Multiobjective Algorithms and their Application to Knapsack and Nurse Scheduling Problems*. PhD thesis, School of Computer Science, University of Nottingham, 2011.

[145] C.-Y. Lee and G. L. Vairaktarakis. Minimizing makespan in hybrid flowshops. *Operations Research Letters*, 16(3):149 – 158, 1994.

[146] S. C. H. Leung. A non-linear goal programming model and solution method for the multi-objective trip distribution problem in transportation engineering. *Optimization and Engineering*, 8(3):277–298, 2007. ISSN 1573-2924.

[147] S. J. Leven and D. S. Levine. Multiattribute decision making in context: A dynamic neural network methodology. *Cognitive Science*, 20(2):271–299, 1996. ISSN 1551-6709.

[148] H. Li and D. Landa-Silva. An adaptive evolutionary multi-objective approach based on simulated annealing. *Evol. Comput.*, 19(4):561–595, Dec. 2011. ISSN 1063-6560.

[149] A. Lim and F. Wang. Multi-depot vehicle routing problem: a one-stage approach. *Automation Science and Engineering, IEEE Transactions on*, 2(4):397–402, 2005. ISSN 1545-5955.

[150] A. V. Lotov and K. Miettinen. *Visualizing the Pareto Frontier*, pages 213–243. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-88908-3.

[151] M. Lukasiewycz, M. Glaß, F. Reimann, and J. Teich. Opt4J - A Modular Framework for Meta-heuristic Optimization. In *Proceedings of the Genetic and Evolutionary Computing Conference (GECCO 2011)*, pages 1723–1730, Dublin, Ireland, 2011.

[152] T. Lust and J. Teghem. The multiobjective multidimensional knapsack problem: a survey and a new approach. *CoRR*, abs/1007.4063, 2010.

[153] B. Maenhout and M. Vanhoucke. Nsplib – a nurse scheduling problem library: a tool to evaluate (meta-)heuristic procedures. In *O.R. in health*, pages 151–165. Elsevier, 2005.

[154] B. Maenhout and M. Vanhoucke. An electromagnetic meta-heuristic for the nurse scheduling problem. *Journal of Heuristics*, 13(4):359–385, 2007. ISSN 1381-1231.

[155] R. Marler and J. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004. ISSN 1615-1488.

[156] R. Martin. *Agile Software Development: Principles, Patterns, and Practices*. Alan Apt Series. Prentice Hall/Pearson Education, 2003. ISBN 9780135974445.

[157] J. Matias, A. Correia, C. Mestre, P. Graga, and C. Serodio. Web-based application programming interface to solve nonlinear optimization problems. In *Proceedings of the World Congress on Engineering 2010, Vol III*, 2010.

[158] M. McShaffry. *Game Coding Complete, Fourth Edition*. ITPro collection. Course Technology PTR, 2012. ISBN 9781133776581.

[159] P. Mestre, J. Matias, A. Correia, and C. Serodio. Direct search optimization application programming interface with remote access. *IAENG International Journal of Applied Mathematics*, pages 251–261, 2010.

[160] A. Mingozzi, M. A. Boschetti, S. Ricciardelli, and L. Bianco. A set partitioning approach to the crew scheduling problem. *Operations Research*, 47(6):873–888, 1999.

[161] S. Mishra, Prakash, M. K. Tiwari, and R. S. Lashkari. A fuzzy goal-programming model of machine-tool selection and operation allocation problem in fms: a quick converging simulated annealing-based approach. *International Journal of Production Research*, 44(1): 43–76, 2006.

[162] M. Misir, K. Verbeeck, P. De Causmaecker, and G. Vanden Berghe. Hyper-heuristics with a dynamic heuristic set for the home care scheduling problem. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8, July 2010.

[163] 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, 2011.

[164] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097 – 1100, 1997. ISSN 0305-0548.

[165] C. Mumford. Simple population replacement strategies for a steady-state multi-objective evolutionary algorithm. In K. Deb, editor, *Genetic and Evolutionary Computation GECCO 2004*, volume 3102 of *Lecture Notes in Computer Science*, pages 1389–1400. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-22344-3.

[166] A. A. Musa and U. Saxena. Scheduling nurses using goal-programming techniques. *IIE Transactions*, 16(3):216–221, 1984.

[167] K. Narukawa and T. Rodemann. Examining the performance of evolutionary many-objective optimization algorithms on a real-world application. In *Genetic and Evolutionary Computing (ICGEC), 2012 Sixth International Conference on*, pages 316–319, 2012.

[168] R. Nystrom. *Game Programming Patterns*. Genever | Benning, 2014. ISBN 9780990582915.

[169] S. Oaks. *Java Performance: The Definitive Guide*. O'Reilly Media, 2014. ISBN 9781449363543.

[170] M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991.

[171] V. Pareto. *Cours D'Economie Politique*. Lausanne, Switzerland, 1896.

[172] E. Parker, S. Zimmerman, S. Rodriguez, and T. Lee. Exploring best practices in home health care: A review of available evidence on select innovations. *Home Health Care Management & Practice*, 26(1): 17–33, 2014.

[173] A. Pillai, A. Joshi, and K. Rao. Performance measurement of r&d projects in a multi-project, concurrent engineering environment. *International Journal of Project Management*, 20(2):165 – 177, 2002. ISSN 0263-7863.

[174] R. L. Pinheiro and D. Landa-Silva. A development and integration framework for optimisation-based enterprise solutions. In *ICORES 2014 - Proceedings of the 3rd International Conference on Operations Research and Enterprise Systems, Angers, Loire Valley, France, March 6-8, 2014.*, pages 233–240, 2014.

[175] R. L. Pinheiro, D. Landa-Silva, and J. Atkin. Analysis of objectives relationships in multiobjective problems using trade-off region maps. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 735–742, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3472-3.

[176] 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: Proceedings of the 7th World Congress on Nature and Biologically Inspired Computing (NaBIC2015)*, volume 419 of *Advances in Intelligent Systems and Computing*, pages 247–259. Springer International Publishing, 2016. ISBN 978-3-319-27399-0.

[177] J. K. Pinto and J. G. Covin. Critical factors in project implementation: a comparison of construction and r&d projects. *Technovation*, 9:49 – 62, 1989. ISSN 0166-4972.

[178] M. Polacek, R. F. Hartl, K. Doerner, and M. Reimann. A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of Heuristics*, 10(6):613–627, 2004. ISSN 1381-1231.

[179] D. Power, F. Burstein, and R. Sharda. Reflections on the past and future of decision support systems: Perspective of eleven pioneers. In D. Schuff, D. Paradice, F. Burstein, D. J. Power, and R. Sharda, editors, *Decision Support*, volume 14 of *Annals of Information Systems*, pages 25–48. Springer New York, 2011. ISBN 978-1-4419-6180-8.

[180] B. Prasad. *Concurrent Engineering Fundamentals: Integrated product and process organization*. Prentice-Hall international series in industrial and systems engineering. Prentice Hall PTR, 1996. ISBN 9780131474635.

[181] R. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill series in computer science. McGraw-Hill Higher Education, 2010. ISBN 9780071267823.

[182] A. Pryke, S. Mostaghim, and A. Nazemi. Heatmap visualization of population based multi objective algorithms. In *Proceedings of the 4th International Conference on Evolutionary Multi-criterion Optimization*, EMO'07, pages 361–375, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-70927-5.

[183] R. C. Purshouse and P. J. Fleming. Conflict, harmony, and independence: Relationships in evolutionary multi-criterion optimisation. In *Evolutionary Multi-Criterion Optimization*, volume 2632 of *LNCS*, pages 16–30. 2003.

[184] R. C. Purshouse and P. J. Fleming. An adaptive divide-and-conquer methodology for evolutionary multi-criterion optimisation. In *Evolutionary Multi-Criterion Optimization*, volume 2632 of *LNCS*, pages 133–147. 2003.

[185] R. C. Purshouse and P. J. Fleming. On the evolutionary optimization of many conflicting objectives. *Evolutionary Computation, IEEE Transactions on*, 11(6):770–784, 2007. ISSN 1089-778X.

[186] I. Quesada and I. Grossmann. An international journal of computer applications in chemical engineering an lp/nlp based branch and bound algorithm for convex minlp optimization problems. *Computers & Chemical Engineering*, 16(10):937 – 947, 1992.

[187] 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. ISSN 0377-2217.

[188] P. Reed, D. Hadka, J. Herman, J. Kasprzyk, and J. Kollat. Evolutionary multiobjective optimization in water resources: The past, present, and future. *Advances in Water Resources*, 51(0):438 – 456, 2013. ISSN 0309-1708.

[189] C. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. Advanced topics in computer science series. McGraw-Hill, 1995. ISBN 9780077092399.

[190] M. G. C. Resende and C. C. Ribeiro. *Greedy Randomized Adaptive Search Procedures*, pages 219–249. Springer US, Boston, MA, 2003. ISBN 978-0-306-48056-0.

[191] Y. Rochat and É. D. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1 (1):147–167, 1995. ISSN 1572-9397.

[192] C. ROMERO. Titles of related interest. In *Handbook of Critical Issues in Goal Programming*. Pergamon, Amsterdam, 1991. ISBN 978-0-08-040661-9.

[193] V. Roshanaei, B. Naderi, F. Jolai, and M. Khalili. A variable neighborhood search for job shop scheduling with set-up times to minimize makespan. *Future Generation Computer Systems*, 25(6):654 – 661, 2009. ISSN 0167-739X.

[194] R. Salimi, H. Motameni, and H. Omranpour. Task scheduling using {NSGA} {II} with fuzzy adaptive operators for computational grids. *Journal of Parallel and Distributed Computing*, 74(5):2333 – 2350, 2014. ISSN 0743-7315.

[195] L. Santiago and T. Bifano. Management of r d projects under uncertainty: a multidimensional approach to managerial flexibility. *Engineering Management, IEEE Transactions on*, 52(2):269–280, 2005. ISSN 0018-9391.

[196] H. Sato, H. Aguirre, and K. Tanaka. Controlling dominance area of solutions and its impact on the performance of moeas. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 5–20. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-70927-5.

[197] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc. ISBN 0-8058-0426-9.

[198] S. U. Seçkiner, H. Gökçen, and M. Kurt. An integer programming model for hierarchical workforce scheduling problem. *European Journal of Operational Research*, 183(2):694 – 699, 2007.

[199] S. Shamshirband, M. Shojafar, A. A. R. Hosseinabadi, and A. Abraham. A solution for multi-objective commodity vehicle routing problem by nsga-ii. In *Hybrid Intelligent Systems (HIS), 2014 14th International Conference on*, pages 12–17, Dec 2014.

[200] F. Siebert. Eliminating external fragmentation in a non-moving garbage collector for java. In *Proceedings of the 2000 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, CASES '00, pages 9–17, New York, NY, USA, 2000. ACM. ISBN 1-58113-338-3.

[201] H. Singh, A. Isaacs, and T. Ray. A pareto corner search evolutionary algorithm and dimensionality reduction in many-objective optimization problems. *Evolutionary Computation, IEEE Transactions on*, 15(4):539–556, 2011. ISSN 1089-778X.

[202] F. J. Solis and R. J.-B. Wets. Minimization by random search techniques. *Mathematics of Operations Research*, 6(1):19–30, 1981.

[203] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2): 254–265, 1987. ISSN 0030364X.

[204] K. Sörensen and F. W. Glover. *Metaheuristics*, pages 960–970. Springer US, Boston, MA, 2013. ISBN 978-1-4419-1153-7.

[205] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evol. Comput.*, 2(3):221–248, Sept. 1994. ISSN 1063-6560.

[206] T. Stützle. Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, 174(3):1519 – 1539, 2006. ISSN 0377-2217.

[207] R. T. Sumichras and I. S. Markham. A heuristic and lower bound for a multi-depot routing problem. *Computers & Operations Research*, 22(10):1047 – 1056, 1995.

[208] R. Swaminathan, Y. Huang, S. Moosavinasab, R. Buckley, C. W. Bartlett, and S. M. Lin. A review on genomics APIs. *Computational and Structural Biotechnology Journal*, 14:8 – 15, 2016. ISSN 2001-0370.

[209] M. Tamiz, D. F. Jones, and E. El-Darzi. A review of goal programming and its applications. *Annals of Operations Research*, 58(1):39–53, 1995.

[210] M. Tamiz, S. Mirrazavi, and D. Jones. Extensions of pareto efficiency analysis to integer goal programming. *Omega*, 27(2):179 – 188, 1999. ISSN 0305-0483.

[211] A. Trautsamwieser and P. Hirsch. Optimization of daily scheduling for home health care services. *Journal of Applied Operational Research*, 3:124–136, 2011.

[212] 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.

[213] W. W. P. Tsang and H. Y. K. Lau. An artificial immune system-based many-objective optimization algorithm with network activation scheme. In *Advances in Artificial Life ECAL 2013*, 2013.

[214] T. Tusar and B. Filipic. Visualization of pareto front approximations in evolutionary multiobjective optimization: A critical review and the prosection method. *Evolutionary Computation, IEEE Transactions on*, 19(2):225–245, April 2015. ISSN 1089-778X.

[215] S. Verel, A. Liefooghe, and C. Dhaenens. Set-based multiobjective fitness landscapes: A preliminary study. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pages 769–776, New York, NY, USA, 2011.

[216] S. Verel, A. Liefooghe, L. Jourdan, and C. Dhaenens. Pareto local optima of multiobjective NK-landscapes with correlated objectives. In *Evolutionary Computation in Combinatorial Optimization*, volume 6622 of *LNCS*, pages 226–237. 2011.

[217] S. Verel, A. Liefooghe, L. Jourdan, and C. Dhaenens. On the structure of multiobjective combinatorial search space: MNK-landscapes with correlated objectives. *European Journal of Operational Research*, 227(2):331 – 342, 2013.

[218] J. A. Vrugt and B. A. Robinson. Improved evolutionary optimization from genetically adaptive multimethod search. *Proceedings of the National Academy of Sciences*, 104(3):708–711, 2007.

[219] D. J. Walker, R. M. Everson, and J. E. Fieldsend. Visualizing mutually nondominating solution sets in many-objective optimization. *Evolutionary Computation, IEEE Transactions on*, 17(2):165–184, 2013.

[220] J. Walnes. Xstream, April 2016. http://xstream.codehaus.org/.

[221] Z. Wang, H. Xing, T. Li, Y. Yang, and R. Qu. An improved moea/d for qos oriented multimedia multicasting with network coding. In *Proceedings of the 8th International Conference on Mobile Multimedia Communications*, MobiMedia '15, pages 20–25. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2015. ISBN 978-1-63190-066-2.

[222] H. G. Welch, D. E. Wennberg, and W. P. Welch. The use of medicare home health care services. *New England Journal of Medicine*, 335(5): 324–329, 1996.

[223] J. Xu and S. Chiu. Effective heuristic procedures for a field technician scheduling problem. *Journal of Heuristics*, 7(5):495–509, 2001. ISSN 1381-1231.

[224] YAML. YAML Aint Markup Language. http://www.yaml.org, 2013.

[225] S. Yang, M. Li, X. Liu, and J. Zheng. A grid-based evolutionary algorithm for many-objective optimization. *Evolutionary Computation, IEEE Transactions on*, 17(5):721–736, 2013. ISSN 1089-778X.

[226] M. Yener, A. Theedom, and R. Rahman. *Professional Java EE Design Patterns*. Wiley, 2014. ISBN 9781118843451.

[227] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on*, 11(6):712–731, 2007.

[228] F. Zhao, Z. Chen, J. Wang, and C. Zhang. An improved moea/d for multi-objective job shop scheduling problem. *International Journal of Computer Integrated Manufacturing*, pages 1–25, 2016.

[229] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms a comparative case study. In A. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 292–301. Springer Berlin Heidelberg, 1998. ISBN 978-3-540-65078-2.

[230] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.*, 8(2): 173–195, June 2000. ISSN 1063-6560.

[231] E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm. Technical report, 2001.

Appendices

Appendix A

A Development and Integration Framework for Optimisation-Based Enterprise Solutions

Decision support systems are pieces of software intended to aid humans in making decisions regarding specific problems in given domains. Over the past few decades, research and development of such systems have increased considerably [179]. Nonetheless, due to their complexity, decision support systems are mostly developed jointly by collaborative teams. Academics often enter into R&D projects in partnerships with practitioners to develop decision support modules – which may consist of one or more optimisation algorithms, simulation modules, inference engines, etc. – for solving a given problem [82].

As with every software development environment, collaborative development of decision support systems presents its difficulties. The software engineering literature describes many methodologies for software development such as applying basic software paradigms [181], or adhering to the Rational Unified Process (RUP) [135], Agile methodology [156] or Concurrent Engineering (CE) [180]. However, the general scope of these methodologies often fails to address the unique challenges of R&D development.

The literature provides a few works that propose general methodologies to aid the development of R&D projects [16, 96]. Some commercial tools exist to aid R&D projects development, such as the IBM

ILOG ODM Enterprise [111]. However, adhering to such tools implies adopting black-box algorithms (which may not be in the interest of researchers), limitations in technology (as compatibility issues arise) and often a considerable cost.

This appendix presents a data-centric development framework to assist in the collaborative development of optimisation-based enterprise solutions. The framework is a tool to facilitate the communication between development teams and to foster progress on the research of the optimisation problem being tackled. Moreover, it provides a layer between the information management system and the decision support module being developed. Hence, the practitioner does not need to force the use of specific technologies and methods and the researchers focus on the development of algorithms. We stress that the proposed framework is not intended to substitute software engineering methodologies and frameworks. Instead, it is designed to provide both the researchers and practitioners with enough independence while still aiding them in the collaborative development. We also describe the application of the proposed framework to a real scenario involving the development of an optimisation-based enterprise solution.

A.1 Related Work

The use of methodologies, frameworks and guidelines aid the process of elaborate software development and can help to reduce risks and problems. In Section B.6.1 we discuss in detail some of the risks and problems that the proposed framework has helped us to address in our project. The literature describes several methodologies and guidelines to help on the development of both decision support systems [36, 54, 70] and information management systems [135, 156, 180]. However, only few works refer specifically to R&D projects and in our opinion, even fewer could be applicable to optimisation-based problems. We highlight the following ones:

- Several works focus on the management aspects of R&D projects. They discuss the advantages of a collaborative project between researchers and practitioners, how to assess such partnerships and how to evaluate the benefits. Among them, we can list the works

of Balachandra and Friar [13], Huchzermeier and Loch [109], Santiago and Bifano [195] and Fontana et al. [82].

- There are some case studies of R&D projects that aim to identify characteristics of the software development process in such R&D projects to then propose a few general guidelines. Among these works we can highlight those by Pinto and Covin [177], and Pillai et al. [173].

- Barnes et al. [16] presents a general good practice model for R&D projects, but not a methodology or framework. Hence, we recommend to adhere to their model while applying our framework, as both methodologies can be fully integrated. Their partner evaluation factors and university-industry issues practises can be addressed prior to the application of our framework. Our data modelling contributes to fulfill several of their project management topics, such as the clearly defined objectives and responsibilities, realistic aims, collaborative agreement, progress monitoring and effective communication. Besides,our data model helps in managing the cultural gap issue and our proposed solution visualisation and auxiliary platform aids with the outcome factors they defined.

- Guarnaschelli et al. [96] proposed a general methodology for integrating the development of decision models with model driven software development. They divide the methodology in four parts, namely the business modelling, the decision problem definition, the service modelling and the reference modelling. The framework proposed in this appendix can be integrated with their methodology since the elements we propose match the elements proposed in their suggested timeline. Our proposed data modelling starts after their suggested business modelling and progresses through the decision model identification and specification. Our data extractor and validator contributes to their decision problem realisation and the domain elements reference modelling. Finally our solution visualisation and auxiliary platform aids with the implementation step.

The framework consists of three components to help with the communication of teams, the understanding of the problem being tackled

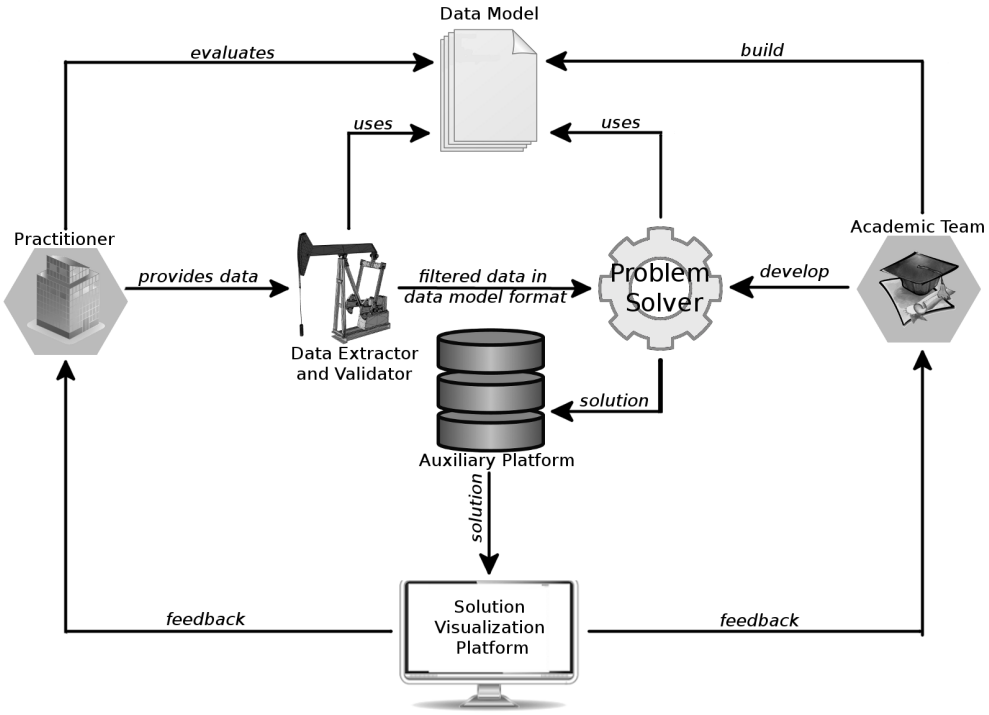


Figure A.1: Overview of the development and integration framework.

and the integration of the algorithmic solution. It is important to notice that these components are not novel and some of the suggested practices are well known to many researchers. Thus, our goal is to formalise these practices and provide a description of our experience to help researchers and practitioners in future endeavours. The first component is a data model understandable to both researchers and practitioners. The second component is a data extraction and validation mechanism. The third component is a solution visualisation and auxiliary platform. Figure A.1 presents an overall diagram of the framework. Altogether, these elements create a layer between the practitioners and the researchers to assist them in achieving a common understanding of the problem and obtaining a higher communication level.

A.2 First Component: Data Model

The data model component applies data-centric development concepts [25, 53] to optimisation scenarios. The data model is a data abstraction in a file format capable of representing an instance of the optimisation problem. It serves as a layer between details of the problem domain and details of the algorithmic solution so that those building the model

approach the problem from their own perspective. The purpose of the data model building process is that both practitioners and researchers work in collaboration to create the file format and improve mutual understanding.

A.2.1 Modelling the Data

The data can be modelled into a set of files or a single file, depending on the characteristics of the optimisation problem. Ideally, the set of files contains all the information needed to run the solving algorithms. We recommend to have in each dataset all the information required to solve one problem instance instead of having data common to several instances in one file. This is because the time required for executing the optimisation algorithms is usually higher than the time required for data processing. More importantly, having all the data for a problem instance available in this way helps researchers to better understand the problem instance.

The HHC scenario used to illustrate the framework in this appendix contains a large dataset, hence we decided to split an instance into six files, each one containing a list of some sort. Figure A.2 presents a simplified version of the dataset. Each file contains a list of the element that defines the file. Also, each file is linked to the mathematical model in order to show the correlation between the optimisation problem and the data files.

The use of a data model at the centre of the development process helps to improve the communication between researchers and practitioners [53]. The data files should preferably be human-readable and as self-explanatory as possible. This also allows researchers to easily alter the data in order to generate different test cases.

When developing the framework we opted to use eXtensible Markup Language (XML) [102] as the file format for the model. We have chosen XML because it is easily readable, XML parsers are widely available, and XML offers a schema model. Such schema allows the creation of XML files that model other XML files and files attached to XML schemas are easily validated. Also, in the case of our HHC project, both researchers and practitioners were already familiar with XML. However, an alternative is to use YAML [224], which is clearer and more human-readable

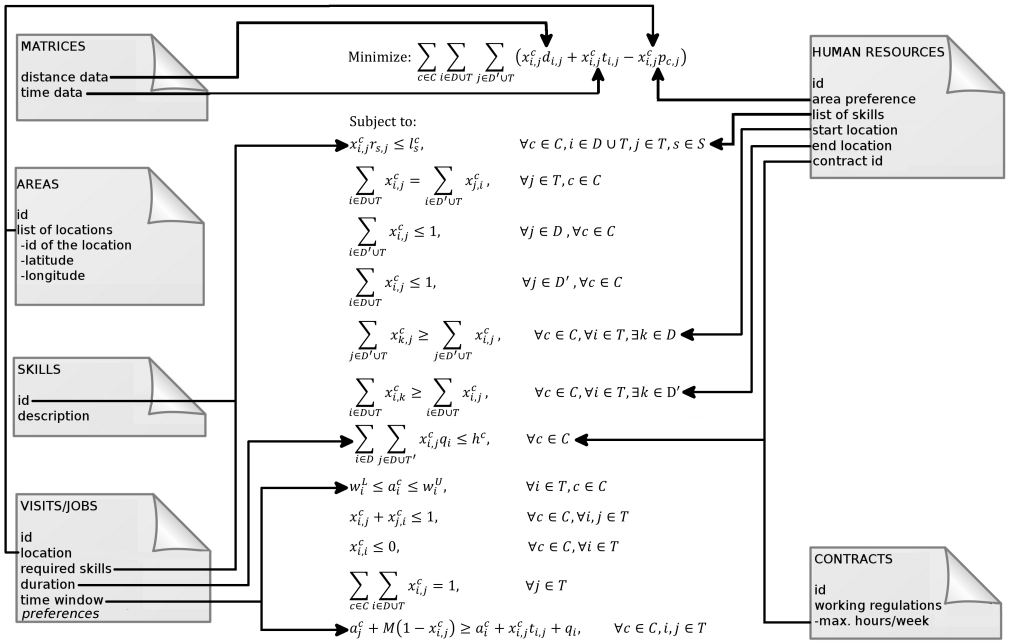


Figure A.2: The files set for the WSRP project. Each file contains a list of the named elements and each element contains several fields related to the optimisation problem.

than XML. Besides, several YAML libraries are available for the most common programming languages.

Like in any other software development process, the development of a decision support system may be subject to many changes during the development. This must be considered when modelling the files, for example, by using required and optional fields. The required fields should be used for very basic data that must exist in order to validate the data for a problem instance. The optional fields are used to capture secondary constraints and objectives such as preferences.

In the modelling stage, the format used to represent a solution should help researchers to verify that the solution is valid, i.e. meets the problem constraints and additional requirements. This also helps practitioners and researchers to understand each other early in the process, on what constitutes a good-quality solution.

The final data model obtained is a layer between researchers and practitioners. This means that practitioners can work on the information management system using the data model to interact with the decision support module (even though it may not have been developed yet). Similarly, researchers can work on the algorithmic techniques without any specific knowledge of the information management system. Further-

more, this layer helps to ensure that any solution methodology developed by the academic team can be incorporated into the company's information system by only handling the corresponding model files. Therefore, the data model plays a crucial role when integrating the company's information system and the developed optimisation algorithms.

A.3 Second Component: Data Extraction and Validation

The sooner the academic team have access to data in the model format, the sooner research on the solving algorithms can begin. Then, sample datasets or a mechanism to efficiently assist in extracting the data would be useful. Figure A.3 shows the schematic for the data extractor and validator developed as part of the framework. The figure also shows the interaction of this component with the database in order to generate the file set in the model format.

A.3.1 Data Extractor

The data extractor is the tool that retrieves the data from the information system and converts it to a set of files according to the data model. This data extractor plays an important role in the development of the decision support module. This is because a typical information system is likely to contain a large amount of data that is not needed for the optimisation problem and that should be filtered. Clearly the model itself helps to eliminate some of such data, but as expressed above, it is highly desirable that each dataset only contains the data required for a single problem instance. In our HHC project, we only extract the visits/jobs for a given planning period. This helps to eliminate the processing of unnecessary data and improve the overall system's performance. Filtering the data at the retrieval stage is usually easier, more reliable and more efficient, specially in a SQL server environment. Figure A.3 outlines this process of extracting and collecting raw data, sending it to the data validator, retrieving the validated data and finally generating the corresponding file set in the model format.

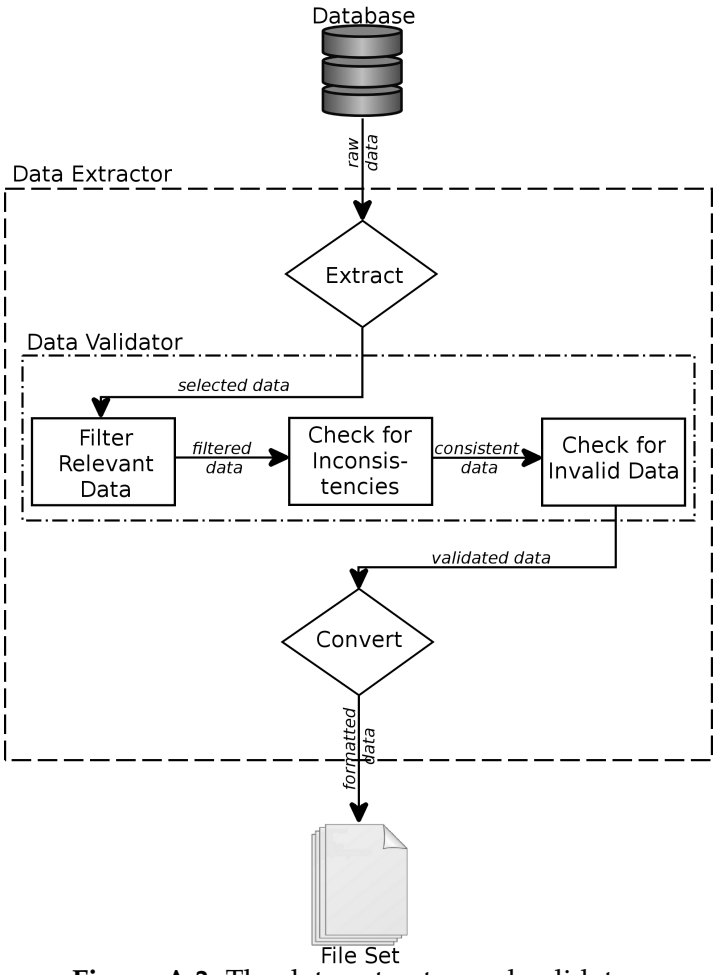


Figure A.3: The data extractor and validator.

A.3.2 Data Validator

After the data extraction, data should be validated to account for missing and inaccurate entries in the database plus other considerations. In the database for our HHC project, this means for example, filtering out staff members that are not available in the planning period considered, or those whose skills are not appropriate for the given scenario, or those only available to work on different geographical locations. We also filter out data about locations and skills not used in the given scenario. Hence, the first step of the data validator is to trim data as much as possible and ensure that data references broken as result of this data trimming are also fixed.

Moreover the validator also checks for inconsistent data, such as mistyped postcodes, which if not detected can negatively affect the performance of the solving algorithm. For example, the data validator for the HHC project checks for wrong postcodes by comparing them to the postcodes

of other locations in the same geographical area. It also checks for any numerical value that deviates too much from the expected range of values. Figure A.3 presents the proposed flow diagram for the data validator. The extractor and validator provides researchers with problem instances to assist the algorithm development, while also provides practitioners with a guiding mechanism to continue the data gathering and preparation.

A.4 Third Component: Solution Visualisation and Auxiliary Platform

The third component of the framework is a solution visualisation and auxiliary platform, a feature that is almost entirely designed to help researchers in the development. There are many information system development methodologies well consolidated in the software engineering literature. However, these are usually not fully applicable to the development of decision support systems, for which well established methodologies and frameworks are not easily identified in the literature.

This third component aims to be a communication layer between researchers and later between researchers and practitioners. Figure A.4 presents a diagram to illustrate how the visualisation and auxiliary platform interacts with the solver and the actors in the scenario, the company and the academic team. The solutions generated by the algorithmic solver can be stored into the repository, can be displayed in the visualisation platform and can be compared to other stored solutions. Through this visualisation platform, researchers and practitioners can have feedback from the solver and can evaluate the progress in the development of the solving algorithm.

A.4.1 Solution Visualiser

In an algorithm development team where each member works in a semi-independent manner with its own knowledge, tools and techniques, having a common solution visualiser can help to conduct a fair comparison between the different solution methods developed. Moreover, a mean to visualise solutions is a useful tool for algorithm developers that can help to save development time. Furthermore, having a common solution

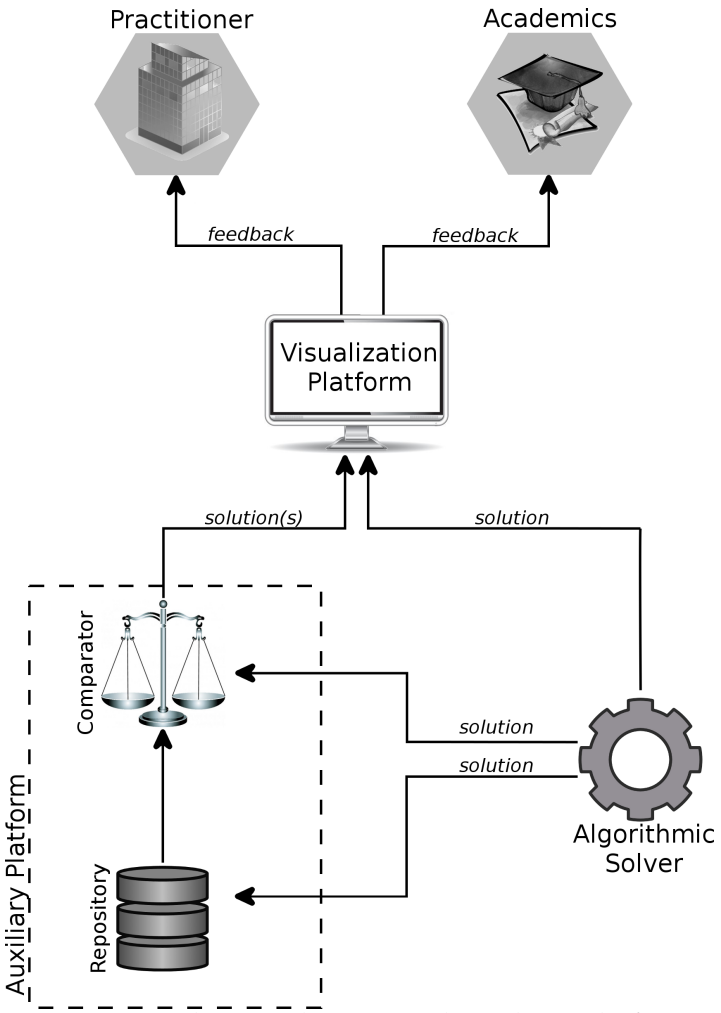


Figure A.4: The visualisation and auxiliary platform.

visualiser allows to present results obtained by different optimisation algorithms in a standardised format for practitioners to evaluate. Therefore, the visualiser is also a communication asset for the company and a tool by which researchers can present progress in their development.

Ideally the solution visualiser should be platform-free (able to run on any operating system), programming-free (not dependant of a specific programming language) and based just on the modelled format (not require extra files, configuration or parameters). To achieve these goals we can resort to some options which are a standalone application, a network-based application or a web-application. It is clear that a web-based environment presents many advantages like easy-access, good presentation and flexibility. The decision usually depends on the nature of the problem and preference of the development team. Furthermore, web-based and network applications are preferred since they

make it easy to integrate with the auxiliary platform. In our scenario, we opted to develop a web-based visualisation platform.

A.4.2 Auxiliary Platform

The auxiliary platform is another tool specially conceived to aid on the research of the optimisation techniques. It consists of a repository of problem instances and solutions, plus an automated comparator of solutions. To compare solutions is fairly easy for single-objective problems in which the optimisation objective is a single scalar value. However, research on multi-objective approaches has expanded considerably and it is now common to apply such approach to real world problems, thus making the comparison between solutions more difficult. The specifics of the comparison are subject to the problem nature and to what the decision makers aim to achieve in a solution. The auxiliary platform should preferably provide charts showing strong and weak points of the solution as well as a detailed table containing useful information. In our HHC project, the solution comparator is able to handle a pair of solutions or two sets of solutions. For the HHC we display the distance travelled by the workers, the distance time, the travel cost, the payroll cost, and preferences. Also, an integration between the visualiser, the comparator and the repository is highly desirable since the usability of the tools escalates.

A.5 Conclusion

In this appendix we have presented a methodological framework to aid on the development and integration of optimisation-based decision support modules and information management systems. The proposed framework combines a collection of good practices along with a data-centric development. The proposed development and integration framework helps to: 1) promote communication between the research team and the company, 2) provide an interface for the integration of the solving algorithms and the information management system and, 3) provide tools to develop and evaluate different algorithmic solutions. Moreover, the framework aims to bestow the academic team full independence to use whatever methods and technologies they choose and an easy manner to

integrate the designed algorithms to the management information system.

Appendix B

A Data-Centric Application Program Interface for Optimisation Problems

In the context of decision support systems for optimisation problems, the literature presents many application program interfaces (API) and frameworks to help researchers and practitioners to apply state of the art solving techniques to optimisation problems, such as ParadisEO [39], jMetal [69] and Opt4J [151].

The aforementioned APIs provide flexible implementations of state-of-the-art solving algorithms that can be applied to many optimisation problems. The user can code their representation of the problem and objective function and plug-in the algorithms to solve instances of the problem. Analogously, having an API to handle the problem data can also be very useful: users could save time by not having to code data parsers and data structures to hold the problem-related data in memory. Less experienced programmers could have access to efficient implementations of the internal structures to handle the problem data, etc.

In Appendix A, we discussed the benefits of having a R&D methodology centred on the problem data. Such benefits include obtaining a deeper understanding of the problem, a higher integration between researchers and practitioners and an improved environment for the development of the solving techniques, where multiple researchers from different backgrounds can efficiently work as a team. In that context, including an API to handle the problem data can further extend the usefulness of their proposed framework as it can increase the productivity

of researchers and developers by simplifying the data access, avoiding rework, improving computational performance and promoting standard solution comparison measures.

However, it is impractical to provide a general API for a given optimisation problem because each optimisation problem has many variants. Thus, even if it were possible to include all variants in a single API, that would have a performance impact as mechanisms for verification and validation would have to be integrated, as well as extra data fields and structures to accommodate different characteristics. Also, having such a flexible API would increase the complexity of using it. Yet, the recent literature presents few related contributions, such as an API to solve nonlinear optimisation problems [157, 159] and a new API for evaluating functions and specifying optimisation problems at runtime [107].

A methodology to allow researchers and practitioners to design and build their own data-centric API is proposed in this appendix. The methodology is composed of three components. The first is a parser for the data files that can read from and write to the modelled format. The second are the data structures containing the relevant optimisation data kept on memory. The third is a mechanism to store and manage solution objects in memory. In this work, we extend that concept by providing further insights and information on how to apply the methodology to build the API. Additionally, we bring the Object Pool, Data Locality, Matrix Ordering, Memory Alignment and the Dirty Flag programming design patterns to reduce CPU cache misses and improve the API computational efficiency. Moreover, we propose a feature called Solution Builder which centralises the objective function and provides a repository that recycles solution objects to minimise the interference of the built-in garbage collector, therefore improving memory fragmentation and further increasing computational performance.

B.1 Related Work

Many APIs and implementations available in the literature focus on the solving techniques. We can highlight ParadisEO [39], jMetal [69] and Opt4J [151] as examples. They are all frameworks that provide several solving algorithms for both single and multiobjective problems. They all have in common the fact that they are built around the solving meth-

ods, and they are flexible enough to be applied for many optimisation problems.

In the literature, we can also find frameworks and APIs with a stronger focus on the problems rather than on the solving techniques.

- Matias et al. [157] and Mestre et al. [159] propose a web-based Java API to solve nonlinear optimisation problems. The API incorporates a set of constrained and unconstrained problems and gives the user the possibility to define problems with custom-made objective functions. However, as with the many works that focus on the solving techniques, defining exclusively the objective function may be too restricting for researching the solver. Hence, our API could be integrated with this or any framework focused on the solving algorithm as we focus on how to access the data efficiently and build solution objects.

- Huang [107] proposes a new API for evaluating functions and specifying optimisation problems at runtime. Huang proposes a Fortran interface FEFAR for the evaluation of objective functions and a new definition language LEFAR for the specification of objective problems at runtime. Conceptually, our proposal differs from Huang's because we are not proposing a general API, but instead conceptualising the design of a tailored API that can help in the research and development of optimisation solutions.

- Groth et al. [94] propose an API that extends the Linked Data architecture for pharmacology data, to enable researchers to read, access and integrate multiple datasets. Their API highlights the importance of a mechanism to handle data access. However, the approach of providing an API to the community cannot be readily applied to optimisation problems because of the high computational performance required by the algorithms.

- Swaminathan et al. [208] conduct a survey of APIs for genomic related applications to improve healthcare practice. The studied APIs (Google Genomics, SMART Genomics and 23andMe) provide mechanisms to parse genomic data and sets of operations to aid researchers and developers to access and operate on the complex

genomic data. Although not related to optimisation, the authors show the importance of such tool on R&D projects.

B.2 API Concept

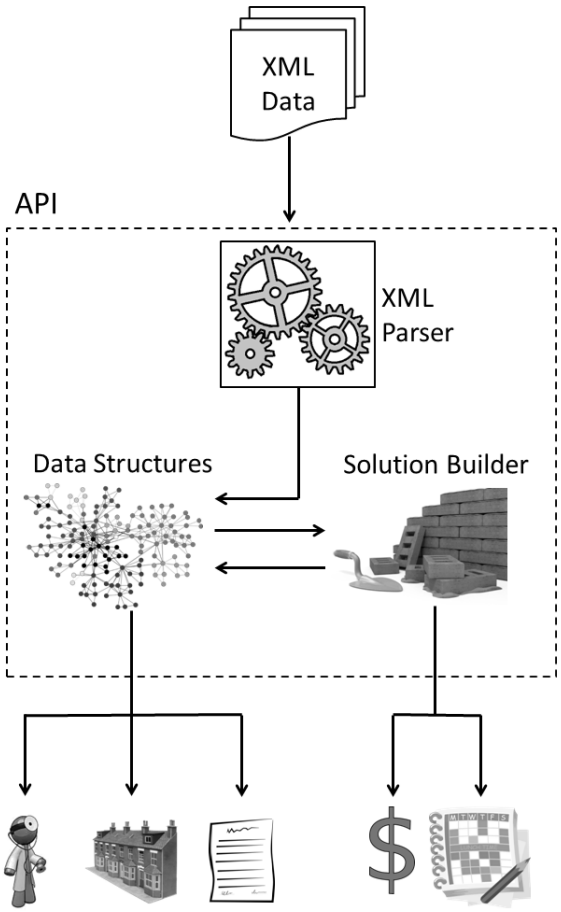


Figure B.1: Overview of the API.

The proposed API is composed of three main components that allow the user to decode the data files of a problem scenario, to load the data into efficient and easy-to-access data structures and to build and evaluate solutions in a straightforward and effective way.

Figure B.1 presents an overview of the API components. On top, we have the *XML Data* (the files containing a problem instance definition) as input for the API. The *XML Parser* decodes the files and builds the *Data Structures* that can be accessed by the user of the API. The user can also access the *Solution Builder* to instantiate and dispose of solution objects of the optimisation problem. The *Solution Builder* communicates with the

Data Structures to evaluate and update current solutions.

These features facilitate the development of both experimental solving techniques and final release versions. Additionally, they provide a reliable way for the optimisation algorithms to query the data and to compare solutions from different approaches. We describe next how an algorithmic solver interacts with the API.

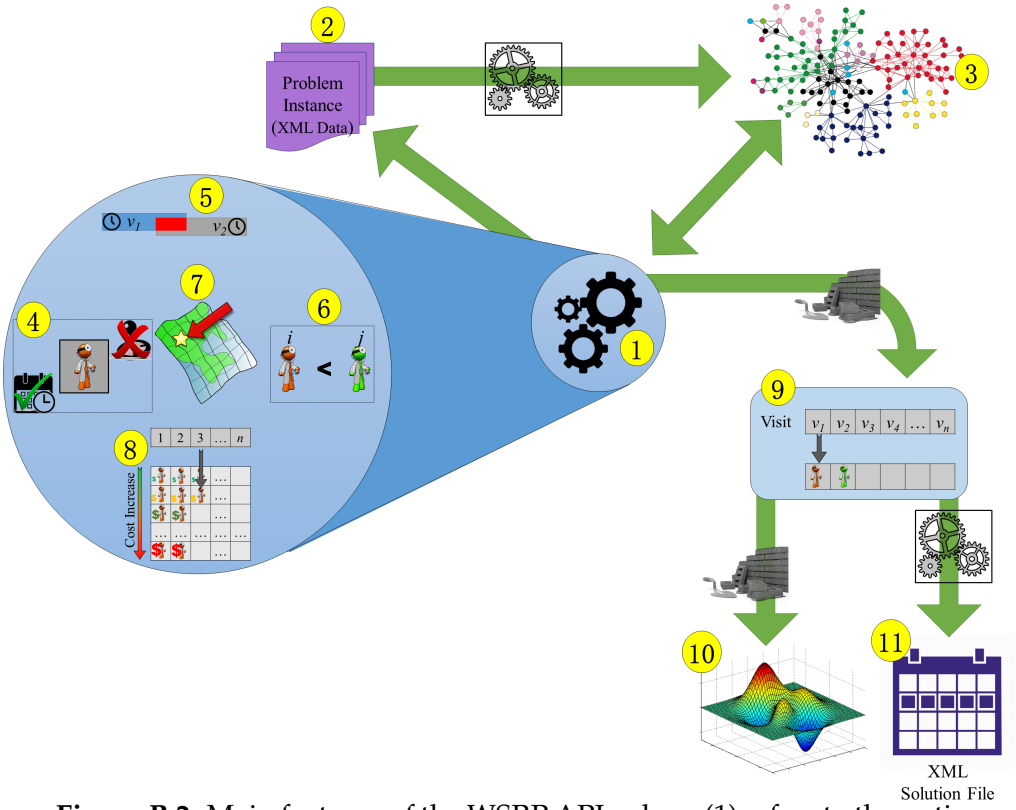


Figure B.2: Main features of the WSRP API, where (1) refers to the optimisation solver (mathematical solver, heuristic algorithms, etc.), (2) and (10) to the data parser, (3) to the internal memory structures, (4) to (8) to the operations and methods supported by the API and (9) to the Solution Dispenser.

Figure B.2 present an overview of the API employed in the HHC project and how the developed solvers interact with the API. Following, we describe each feature numbered in the figure. Some features extend to all problems and can potentially be employed by an API tackling any problem.

1. **Solver:** the API facilitates the use of different solving techniques (exact solvers, heuristic algorithms, etc.) on the problem because it provides a set of methods to quickly access various features of the problem, such as values, constraints and basic operations on the

data. Therefore, the researchers can plug in existing solving APIs and frameworks or new algorithms.

2. **Problem Data:** we consider that each problem instance can be contained in a set of data files (we use XML files) with all information related to a single instance of the optimisation problem.

3. **Internal Data Structures:** one of the main functionalities of the API is to provide a set of efficient data structures to hold the problem data in memory. After the solver selects a problem instance to load, the data parser reads the files and allocate the problem data into memory. These structures are designed to provide maximum access performance to the solver.

The API should provide secure and efficient access to the internal data structures and methods. We now present what we consider essential to be provided by the HHC API:

4. **Constraints:** the API provides several methods to assess and evaluate the underlying problem constraints related to assignments. For example, given an assignment (a visit and a worker), the API provides methods to:
   - Check if the worker is skilled for the visit.
   - Check if the worker possesses a valid contract to perform that visit.
   - Check if the worker is available at the time of the visit.
   - Check if the worker can commute to the place of the visit.

5. **Operations:** the API also provides several operations to be used by the solver. These operations can be simple checks and validations or complex functionalities. Among the operations in the HHC we highlight the following ones:
   - Check if two given visits are time-conflicting.
   - Check if a worker is qualified to perform two given visits.
   - Identify the best contract for an assignment.

6. **Preferences:** in addition to constraint methods, the HHC API also provides methods for evaluating preferences (worker, staff and patient's preferences):

- Retrieve the preferred worker to perform a given visit based on historical data.
- Identify the visits that a given worker prefers to perform.
- Select the worker that best matches a given patient preferences.

7. **Geographical Methods:** because the HHC combines scheduling and routing, we designed a set of methods related to the geographical data, including:
   - Check if a given visit is within a specific area.
   - Given a transportation mode, calculate the time to commute from a visit to another.
   - Check which transportation modes can be used within an area.
   - Retrieve all visits within an area.
8. **Overall Data Access:** it is important to notice that although the provided methods and operations are extensive, specific solvers or techniques may require a different way to access the data. Hence, the HHC API also provides basic 'get' methods for all information (Figure B.3).

Finally, the API also provides functionalities for building and maintaining solution objects for the optimisation problem:

9. **Solution Handling:** the solver may deal with a single or with multiple solutions simultaneously. The Solution Builder provides an interface for the solver to create and store solutions in memory. This component provides the following methods:
   - Create a new solution.
   - Add assignment – given a visit, a worker, a contract, start time and transportation mode, the method uses this information and creates an assignment in a given solution.
   - Dispose a solution.
10. **Evaluation Mechanism:** the builder also provides an interface so that a solution can be evaluated. The evaluation mechanism comprises of:

- A built-in objective function that calculates the values of objectives and constraint violations.
- A scalarisation function that takes the objective-values and weights and generates a solution fitness value. The objective-weights can be customised.
- Dispose a solution.

11. **Save Solution:** additionally, the data parser can save a solution object to an XML file that can later be retrieved and loaded back into memory.

Next, we detail each component of the proposed methodology to build an API for optimisation problems data.

These features facilitate the development of both experimental solving techniques and final release versions. Additionally, they provide a reliable way for the algorithms to query the data and to compare solutions from different approaches. We describe next the first feature of the proposed API, the data parser.

B.3 XML Data Model

In Appendix A we proposed the use of a data model to represent the optimisation problem features and data. Hence, the first component of the API is a parser to read the files from the data model and build the data structures. The parser is also responsible for converting a solution of the optimisation problem into the XML data file. Additionally, the parser implementation should accommodate extensions and updates in the data model.

For that purpose, we employed a serialisation library. Since we used Java, we employed the XStream Java library [220], however, most high-level languages have XML serialisation mechanisms available. The advantage of such approach is that a set of classes that corresponds to the modelled data can be used, and the serialisation library handles all the file parsing. This method is easy to develop and does not require extensive programming effort. However, it is likely that the objects in memory are not best suited for performance or intuitive access because the serialisation mechanism often requires intermediate classes and public access

to attributes. Hence, the parser is used exclusively to translate the information from and to the files. For efficient and easy access to the data, we need another set of data structures.

B.4 Internal Data Structures

The second component of the API is composed of the data structures that hold the problem-related data in memory. It is important to emphasise that while the aforementioned data model is intended to be an explicit representation of the optimisation problem, the internal data structures must be efficient for access by solving algorithms.

Therefore, we must ensure that the operations invoked during the optimisation are performed on constant ($O(1)$) time when possible. Additionally, the API should be flexible enough to accommodate different solving techniques easily, as we are not only concerned about the final product but with the entire process of executing the R&D project. Hence, in our work, we divided the API into groups of objects, following the data model orientation:

- Visits – containing information about the requirements of each visit, e.g. number of workers required, start time, duration, location, skills required, preferences, etc.
- Workers – containing information related to the workers themselves, e.g. skills, availability, home location, preferences.
- Areas and locations.
- Transportation modes.
- Contracts – containing information regarding maximum and minimum working hours and costs.

In order to hold the objects in memory, we first use arrays. The advantage of using arrays is that the random access using indexes is very efficient ($O(1)$). The disadvantage is that memory allocation requires to estimate the size of the data or use dynamic memory allocation which could also hinder performance. To increase the loading performance, we added a new XML file to the model, called 'metadata.xml' that accommodates several information and statistics of the problem instance,

such as the number of workers, tasks, the date format used in the files, etc.

Using arrays may be sufficient for most solving algorithms as it allows fast random access. However, it may be a problem with specialised heuristics or external software accessing the API. Such systems are often linked to a database. Hence, they handle elements using an identification number, which is stored in the XML, but is not consistent with the index of the arrays. To solve this matter, we employ a second data structure, a hash table, linking the identification numbers of the database entries to its respective objects. To provide better usability with both data structures, we encapsulated both the hash table and the arrays, for each type of objects, into a single class representing the set of elements.

Finally, to improve the usability of the API, we define a naming convention according to the operation performance. All methods starting with the words 'get', 'is' and 'has' are guaranteed to perform in $O(1)$ time. All methods starting with the word 'calculate' are guaranteed to perform in $O(n^k)$ time in the worst case.

Figure B.3 presents a class diagram of the data structures. For simplicity, we included only the main class that defines the optimisation problem and the classes that define the tasks and the set of tasks. The main class, *WorkforceSchedulingAndRoutingProblem*, is composed of sets of elements included in a problem instance, namely areas, tasks, human resources and contracts. This class provides an interface such that the user can retrieve each set and its elements. Also, this class allows the user to obtain the Solution Dispenser, explained in the subsequent section. Note that the *calculateMinimumNumberOfAssignments* method, as aforementioned, starts with the 'calculate' word, hence in the worst case it performs in $O(n)$, while all 'get' methods performs with $O(1)$ time complexity.

The *Tasks*, *Areas*, *HumanResources* and *Contracts* classes contain both the arrays of elements and the hash table linked by each element's identification number. Hence, when using these classes it is possible to interact through all elements or retrieve a specific one given its identification number or index, as we can find in the *Tasks* class. We see in this class that it is possible to identify an ordered list of tasks *ls* representing the array and the hash table *hashTable* containing the mapping of identification numbers. Finally, the class *Task* contains all the methods to access

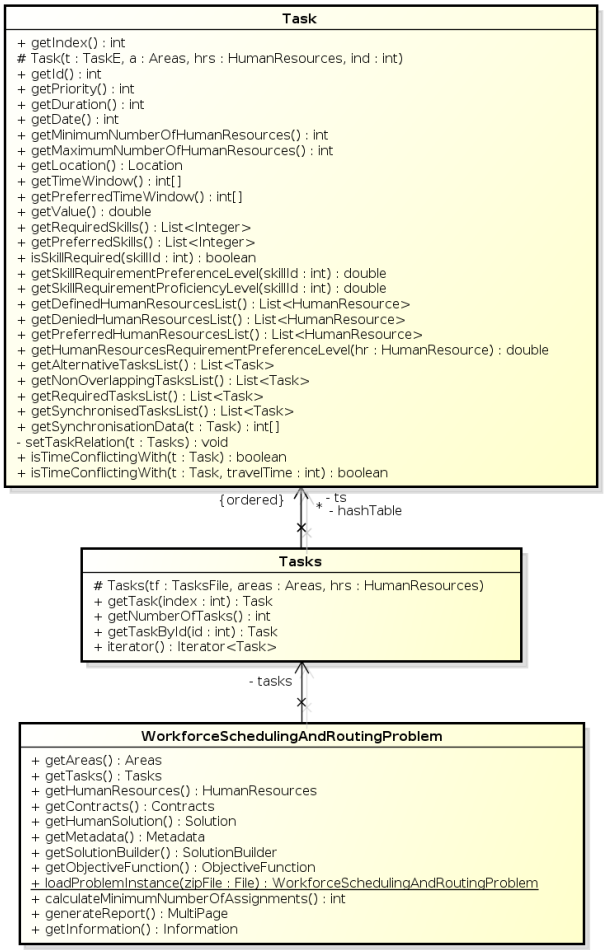


Figure B.3: Class Diagram for the Main Problem Class and the Tasks-related Classes.

the data from a single task plus some useful operations, such as *isTimeConflictingWith* which checks if a second given task conflict in time with the current task (hence they cannot be performed by the same worker).

B.4.1 Code Optimisation

McShaffry [158] states that the performance of an application can be influenced by the data structures employed and by how the data itself is organised and accessed by the code. Additionally, a good use of the processor's internal cache can potentially increase performance by up to 50 times [168]. Hence, we now present some techniques found in the literature that can improve the performance of the data access and subsequently of the algorithms that make use of the API.

Data Locality One of the most overlooked ways to increase (or decrease) performance in an application is due to the internal processor's cache memory. Modern computers possess an internal processor memory (cache memory) that bridges the access to the main RAM memory to increase the system performance. In summary, when the application requires some data in the memory, the CPU loads an entire section of the main memory into the faster internal cache. When requesting the next data, it first checks if it is already in the cache. In case it is (*cache hit*), the access is very fast as the data is promptly available. In case it is not in the cache, we have a *cache miss* and a section of the memory containing the required data is loaded into the cache [104].

[168] proposes the *data locality* design pattern that attempts to reduce the number of cache misses in an application. The design pattern consists of sacrificing some abstraction and object-oriented concepts to better arrange the data inside of an object such that, when accessing the object attributes, the number of cache misses is minimised.

Take for instance the *Task* class in the sample API. If we sequentially define the variables for id, priority, duration and date in the class and we always read them in that order (let us say in the objective function), we are promoting cache hits because they are likely to be loaded together into the cache.

Another way to improve cache hits is to use arrays of homogeneous data. Suppose some massive calculations require the costs of all tasks to be processed. Instead of having an attribute 'cost' inside task objects, it is more efficient to have an array containing all the costs for all tasks such that when a calculation is performed, the sections loaded into memory will include the costs for multiple consecutive tasks, hence actually promoting multiple cache hits.

This pattern affects the way getter and setter methods are implemented. Employing such methods is considered to be a good object-oriented practice, as it promotes encapsulation, error control and readability. However, it can cause cache misses because of the indirect referencing. Languages such as Java have in-line optimisations which can convert a setter or getter method into direct access to the attribute during the compilation of the code [169]. Hence setters and getters can be used without fear of hindering the performance. It is important to know beforehand the characteristics of the programming language used and

the compiler employed to establish whether making use of setters and getters impacts performance or not.

Matrices Ordering Another pattern aimed to improve cache memory access focuses on an issue that may be overlooked: the ordering of matrices in memory [158]. Optimisation problems data is often organised in matrices. Accessing this data in the correct order to avoid cache misses can potentially improve performance. Suppose the data from a matrix is stored in memory ordered by row (i.e. [0,0],[0,1],[0,2] are respectively adjacent). If we follow the row ordering to access the data, entire sequential sections of the memory will be loaded into the cache, hence promoting cache hits. Now consider the opposite scenario, if we accessed the matrix in column order and given that a row is larger than the section that will be loaded into memory, we have the worst case scenario where every access results in a cache miss.

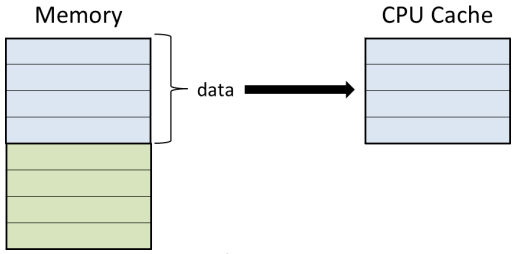


Figure B.4: Memory mapping from memory to CPU cache of an aligned chunk of data.

Memory Alignment McShaffry [158] notes that the CPU reads and writes memory-aligned data noticeably faster than misaligned data. A given data type is memory-aligned if its starting address is evenly divisible by its size (in bytes). An aligned chunk of data is promptly loaded into the cache while an unaligned chunk of data must be read in parts, shifted to the target frame and then loaded. Figure B.4 presents a diagram of a memory mapping of aligned data. The mapping is direct, meaning that the data is directly transferred from the RAM memory to the internal cache memory. Figure B.5 presents the copy of misaligned data. The CPU reads the two chunks containing the parts of the required data, shifts each piece to select the requested information and merge into a single chunk which is then copied into the cache. Clearly, the second case represents a much slower mechanism.

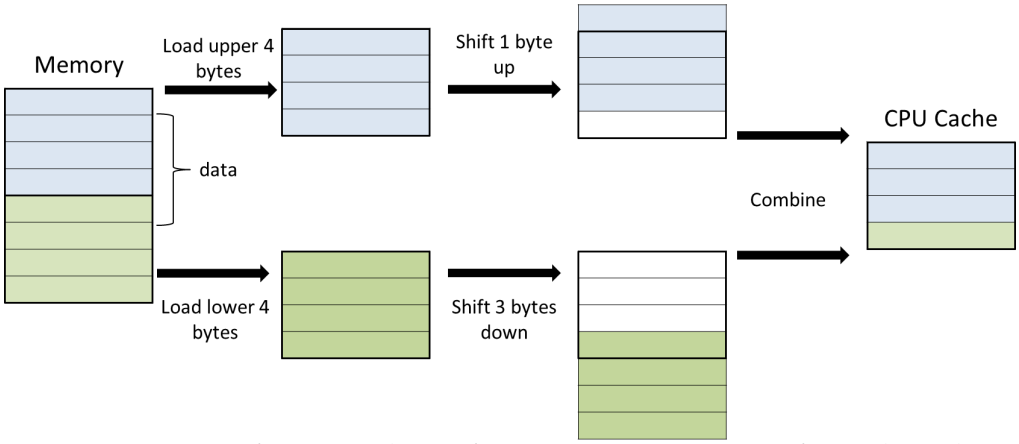


Figure B.5: Performance loss of a memory mapping of misaligned data.

The best way to take advantage of memory alignment is to make sure that the internal data types, structures and classes of the API have a number of bytes that is a power of 2. In the case a data type has fewer bytes, dummy variables can be added to force the structure (or object) to have the desired value. Also, it is imperative to be aware of the overhead of space required by the programming language for classes and data structures before computing the total size.

B.5 The Solution Builder

The last component of the API is the Solution Builder (SB). The SB have two roles:

1. To provide a standard mechanism to calculate the fitness (objective-value) of a given solution to the optimisation problem.

2. To provide an efficient way to handle solution objects in memory.

The SB provides an interface for the user to build and assess a solution to a given problem instance. Once the problem is loaded into an object, the user can invoke the SB to create a new solution object. A new empty solution is created, and an identification number is returned to the user. This number can be used to access the solution, add new assignments and evaluate the solution fitness according to multiple criteria (preferences, objectives or constraints). The user can also invoke the objective function to evaluate the solution. Figure B.6 presents a schematic of the SB component.

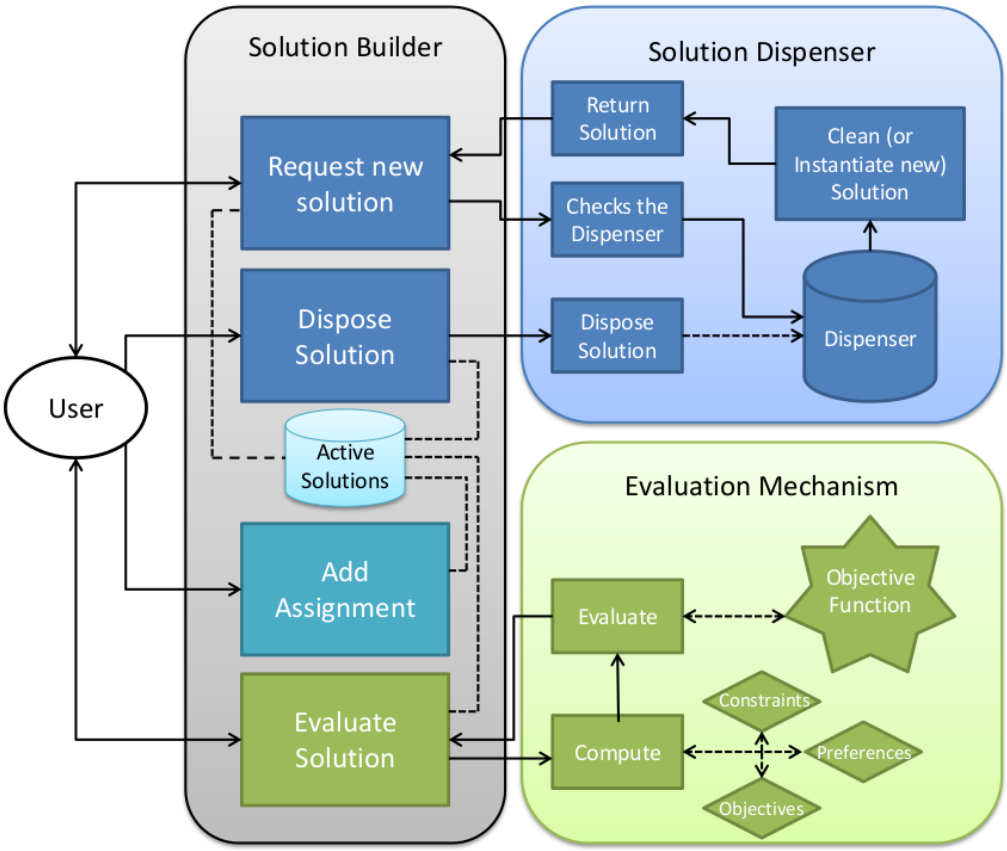


Figure B.6: Flow Diagram for the Solution Builder. The Rectangles Inside the Solution Builder Box Represent the Functionalities Given to the User. Arrows Represent Data Flow.

B.5.1 Centralised Evaluation Mechanism

Integrated into the SB is the solution evaluation mechanism. Pinheiro and Landa-Silva [174] pointed at the importance of a mechanism to ensure fairness in the comparison of results from different techniques implemented. Having a centralised objective function is beneficial and helps to avoid re-work and maintain consistency.

In the HHC project, the weights of the objective function are initialised with standard values, but the SB allows the user to set them according to specific needs. Additionally, the user can evaluate specific aspects of the problem (total distance, total costs, constraints violations, preferences, etc.) or obtain the overall fitness of a given solution.

B.5.2 Solution Dispenser

Modern programming languages, such as Java and C#, provide a convenient way to handle objects: a garbage collector. The user only needs to

dispose links and pointers to objects, then the garbage collector looks for objects not linked by the user's program and frees the memory. That can lead to two problems: memory fragmentation and extra processing time to seek, to free memory and later to allocate new objects [11, 200].

Leaving the disposal of objects to the garbage collector can lead to a decrease in performance that, aside from being marginal for most applications, can have an unacceptable impact on optimisation algorithms. Hence, the SB internally implements an Object Pool design pattern [168] to recycle objects. We employ a factory object implemented using the *factory* design pattern [226] for easy creation of the objects. This factory is responsible for creating new solution objects.

When a new solution request is invoked (Figure B.6), the factory seeks its internal solution repository (a list of disposed solutions). If there is a solution available in the repository, it retrieves it, clears the solution data and returns it to the solver. The solver now has an empty solution that it can use. When the solver does not need the solution anymore, it can dispose of the solution by invoking a specific method in the SB. The factory then receives the disposed solution and stores it in the list.

Potentially, the use of the solution dispenser can provide significant performance gains. Take for instance a population-based algorithm that processes one generation per second with a population of 100 individuals. That means 100 solutions being disposed of per second. After ten minutes running, the algorithm will have disposed of 60,000 solutions, which potentially could fragment the memory and cause several garbage collection calls. Now, when using the dispenser, considering the worst case, when a new population is created before disposing of the old one, we need 100 active solutions per population, totalling 200 active solutions that will be recycled during the execution. Thus, in this hypothetical scenario, we could have a decrease of 99.6% on the number of objects used, which could represent a reduction of 97.5% on the processing time and memory consumed by the garbage collector.

B.5.3 Code Optimisation

The SB itself is a component to improve the efficiency of the API with the dispenser being a specialist implementation of an Object Pool design

pattern. On the evaluation mechanism, however, programming techniques can be applied to improve the overall evaluation performance.

Dirty Flag Nystrom [168] proposes a design pattern called *dirty flag*. This pattern consists of a mechanism to avoid unnecessary recalculations when you have nested operations, usually on recursive calls. In the solution evaluation context, the objective function can be very costly, and recursive calculations may be needed. Additionally, algorithms may require specific parts of the objective function to be calculated at different points and in multiple times. This pattern consists of having a flag (boolean variable) to define if the state of an object has changed. If so, the values that rely on that object must be re-calculated. Thus, in a recursive operation where a value would always be calculated, it will now be calculated if the flag indicated that. Essentially, after a solution is built, only one evaluation of its values are made. If an algorithm calls the evaluation (or partial evaluation) on an unchanged solution, the last calculated value is returned.

B.6 Summary

It is difficult to provide an API to handle the data of an optimisation problem that is generic enough to support many variants of the problem (hence be widely useful) and be computationally efficient. Therefore, in this appendix, we proposed a methodology to help researchers to design and implement a tailored API for specific optimisation problems. Initially, we proposed the use of a data parser to read the files from disk and load them into memory. We defined a set of data structures to hold the data in memory and provided examples of methods and operations that further increase the usefulness of the API. We brought programming design patterns to our model to improve its effectiveness, such as the Data Locality, Dirty Flag and Object Pool. Also, we proposed efficient techniques to improve matrix access performance and to reduce cache misses during the execution of the code. Finally, we introduced a Solution Builder which centralises the objective function, hence promoting fairness on the comparison of solutions arising from different solving techniques. The Solution Builder also provides an object repository that handles the memory allocation and disposal of memory objects to avoid

garbage collector calls.

B.6.1 Improvements on the Research and Development

In this section we present the results obtained by applying the framework proposed in Chapter A to the HHC project. We highlight the problems found during the development process and how applying the framework helped us to tackle these problems.

Better Collaboration Between Researchers and Practitioners

Communication Issues. Our experience showed that communication between the actors of the scenario can potentially be problematic. Information, requirements and expectations can be different from the perspective of researchers and practitioners, leading to misunderstandings and misconceptions. The data model building process played a major role in the researchers' understanding of the optimisation problem details. During the process we were able to identify gaps in our conceptualisation and fill them during a phase where changes caused minimal drawbacks. We were able to identify more constraints for the optimisation problem as well as alter some features to better suit the needs of the practitioner.

Knowledge Background Issues. There is obviously difference between the background knowledge of the company developers and that of the academic team. Moreover, researchers in the team may have different expertise including maths, management science, computer science, statistics, etc. This variety could lead to some difficulties such as different interpretation of problem data, unfamiliarity with certain technology and different assessment of results. With the data model we created a structured set of files in which researchers were able to easily browse the problem data without knowing any specific technology employed on the information system. Besides, the visualisation platform proved to be a valuable asset to give feedback to the company. Thus, the framework aided the academics on minimising the knowledge-gap difficulties.

Integration Issues. Integrating a decision support module to an information management system can be difficult and tricky. The first issue is the problem data retrieval. The second issue is applying the devel-

oped algorithms to an existing information management system. Integrating the solver can be difficult, specially if it uses a specific tool, such as mathematical solvers or specific software libraries. The data model provided a layer between our developed algorithms and the information system and the researchers found themselves in a comfortable environment where they employed different technologies from the ones used by the company.

Better Research and Development Process

Teamwork. In our example project, the academic group was formed by several PhD students, each one using the HHC as a background for its PhD research. The application of the framework helped considerably to improve communication among academics. The optimisation problem was discussed and a common understanding achieved. Besides, building the tools in the proposed framework provided a good opportunity for teamwork and knowledge sharing.

Rework. When developing algorithmic solutions for a given optimisation problem, researchers tend to create their own versions of tools such as file models, data retrieval API's, solution viewers and representations, etc. It is rarely the case that researchers reuse these types of tools developed by other researcher. Instead, they tend to reuse algorithms in their quest to develop improved versions of those. In our HHC project, the data extractor and the solution visualisation and auxiliary platform were employed by all academics. Although the deployment of these framework tools required some time, reusing these tools saved considerable time and effort overall by eliminating rework. Along with the data model we also developed a Java API to handle the XML files and access the data. This API further extended the usefulness of the model by providing the users with an easy way to access the problem data.

Benchmarking. In real world problems, a good benchmark dataset to evaluate the solutions achieved during the research stage, is often difficult to obtain. Sometimes it is possible to estimate some close enough targets, but as the complexity of the problem grows, such approaches become less practical. Hence, it is necessary to have a fair and clear way to benchmark solutions to the real world scenarios, obtained with different algorithmic techniques. In our HHC project, the auxiliary platform provided the team with a valuable asset to compare solutions more fairly

and improve the overall benchmarking process.

B.7 Experimental Results

We now present the results of the technical analysis on the Solution Builder component and the benefits of using it. Additionally, we provide evidence of computational performance gains when using the design patterns suggested in the previous section.

B.7.1 Solution Builder

The SB is responsible for holding solution objects for the given problem. Small problems using larger encoding might consume more memory than larger problems using smaller encoding. For example, an integer array representation (an array the size equal to the number of tasks) is an example of smaller encoding and a binary array representation (a matrix which is of size *number of workers* $\times$ *number of tasks*) is an example of a larger encoding. Then, to test the solution dispenser we compared different encoding schemes, integer arrays varying from small (25 elements) to large (25,000 elements), and binary representation. Note that even though integer variables are employed to represent binary encoding (thus, a larger data type than necessary), the literature commonly present the encoding of binary variables using complex objects [69], hence, it is reasonable to use integer variables instead of binary ones in our experiments.

We defined our experiments as follows: for each encoding size, we sequentially created and disposed of one million solution objects. For the experiments using the garbage collector, the disposing process merely unlinked the objects to free them to the Garbage Collector (GC). For the SB, the internal dispose process is called and also the object data is cleared. We ran each set-up for five times and computed the average results. Additionally, to measure time and memory we used the integrated profiler available in Netbeans which can accurately measure the time spent on each method and the memory allocated during the execution of the application. The experiments were performed in Java on a quad-core Intel i7 machine with 32GB memory running Linux. The main reason for choosing Java is that it is a mature language, multi-platform

and widely employed for optimisation problems with a large number of optimisation algorithms and frameworks implemented and available for public use [39, 69, 151].

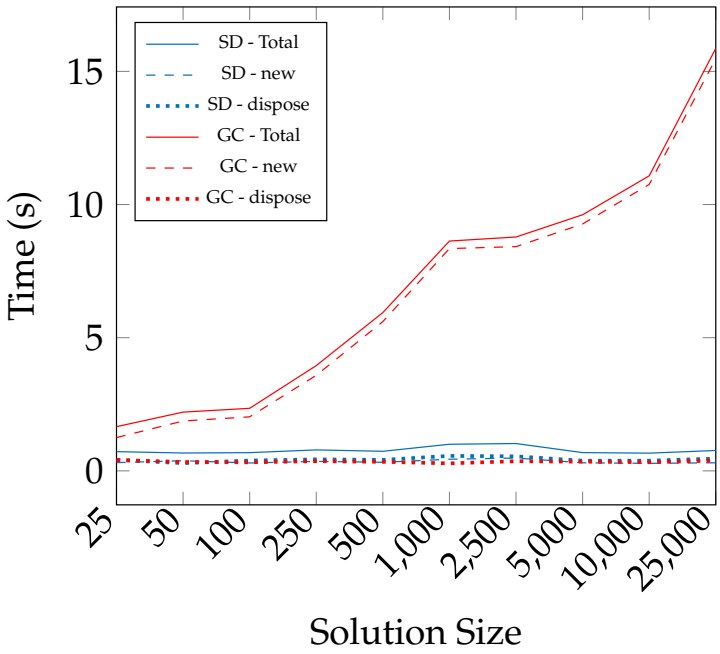


Figure B.7: Time comparison between the Solution Dispenser (SD) and the Java Garbage Collector (GC) to instantiate and dispose new solutions.

Figure B.7 presents the results of the computation time. The red lines represent the time spent in seconds on experiments using the Java garbage collector, and the blue lines represent the time spent on experiments employing the solution dispenser. The solid lines represent the total time; the dashed lines represent the time used by the *'new'* method, which allocates new solution objects, and the dotted lines represent the *'dispose'* method. The time spent by the SB follows a constant trend throughout all encoding or solution sizes. This happens because both the *'new'* and *'dispose'* operations of the solution dispenser perform in constant time and since there are no objects freed in the memory (they are being kept alive by the SB), the garbage collector (automatically activated by the Java virtual machine) just quickly checks for dead objects, finds nothing, and is deactivated without any extra processing.

Regarding the tests using the garbage collector, we see that the *'dispose'* operation is performed in constant time, but the *'new'* method requires higher time proportional to the size of the solutions. We can clearly see that relying on the garbage collector to dispose and allocate

new objects can hinder the performance of the application. Also, it is important to notice that we did not specify any parameters for the Java virtual machine. Hence the experiments had as much memory as it was required. In a real-world environment, that might not be the case. Many processes may be active in the machine, and the memory might be limited, which would make the garbage collector to be active more often than it was on the presented tests, hence further decreasing the performance.

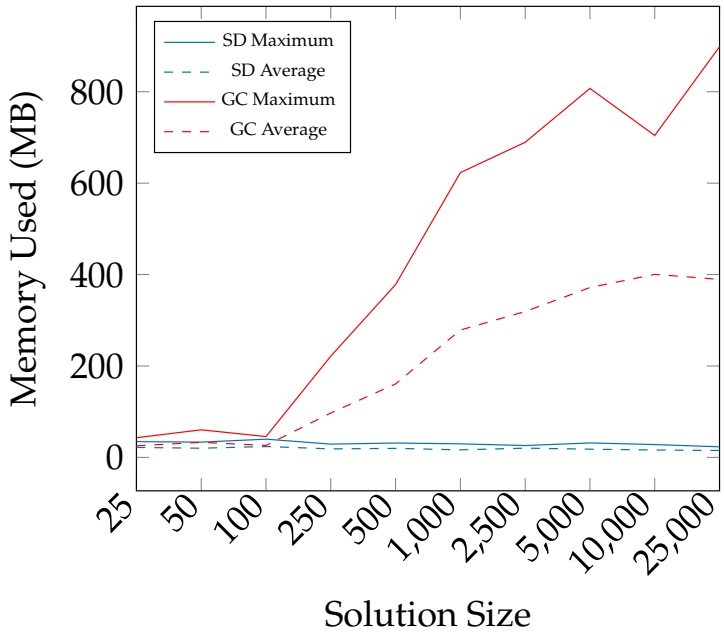


Figure B.8: Memory comparison between the Solution Dispenser (SD) and the Java Garbage Collector (GC).

In Figure B.8 we have the results of the memory allocation measurement. The red lines represent the experiments using the garbage collector and the blue lines the solution dispenser. Also, the solid lines represent the maximum memory allocated in MB and the dashed lines the average memory allocated. Analogously to the previous chart, it is clear that the memory required by the SB, not surprisingly, is constant throughout all experiments. Although the size of the array changes on each experiment, only one object is allocated in memory during runtime. However, when relying on the garbage collector, we see that it makes use of much more memory, which reinforces our previous statement that, in a scenario where the memory is limited, the garbage collector requires more frequent activation.

Finally, we conducted experiments running a Genetic Algorithm (GA)

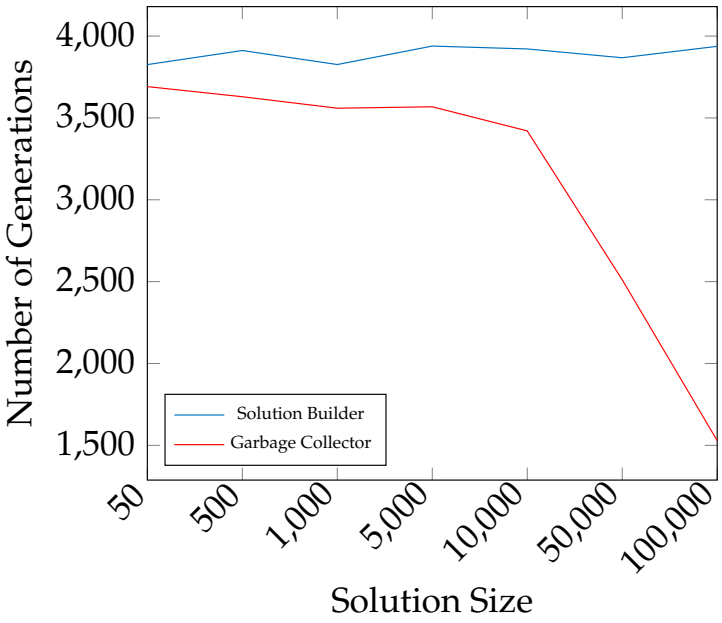


Figure B.9: Number of generations of a genetic algorithm with different solution sizes.

Goldberg [92] through the HHC API to solve the real-world scenarios. To isolate the impact of employing the Solution Builder, we ran the experiments on a single test instance varying only the size of the data structure used to represent a solution. Therefore, all the experiments required the same computational effort regarding the genetic operators and solution evaluation. Also, because of the stochastic nature of GAs, we used fixed seeds for the random number generators to increase the fairness of the comparison. We performed eight runs of the GA, both for the solution builder and the garbage collector, and computed the average number of generations after one minute. Figures B.9 and B.10 present the results. In Figure B.9, we present the total number of generations for the GA as the size of the solution representation increases. It is evident that when employing the solution builder the average number of generation is roughly constant regarding the increase in the size of the representation. It is also noticeable that if the solution size is not large enough (in this case the equivalent of 10,000 integer values), the performance gain of using the SB is below 10% (Figure B.10), but still substantial. Nonetheless, when the size of the representation increases, the benefit of employing the SB represented up to 150% rise in the number of generations.

Thus, it is apparent that by using the solution builder we can achieve substantial improvements both in time and memory consumption. This is particularly true for problems where the solution encoding is large.

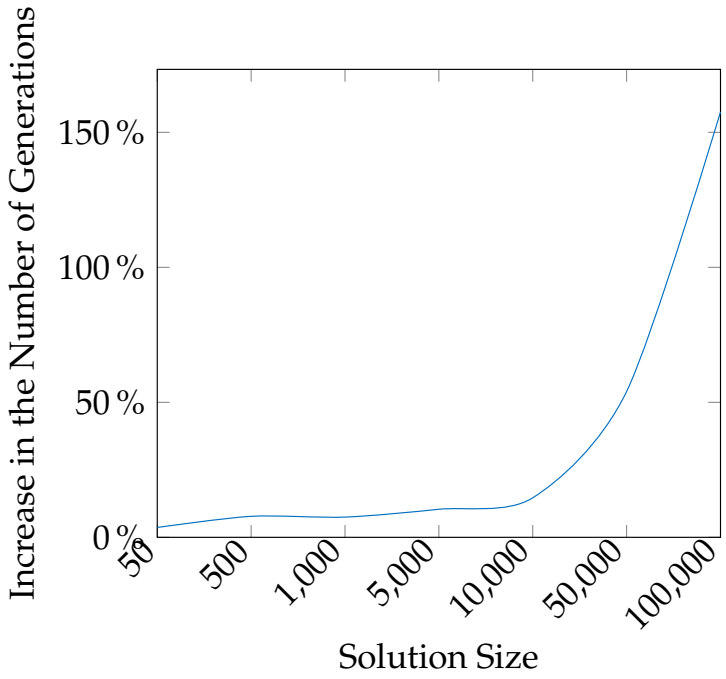


Figure B.10: Increase in the number of generation when employing the Solution Builder in detriment of the built-in Garbage Collector.

Also, the idea of recycling objects could be implemented in the solver algorithms themselves, especially in population-based algorithms (because of the high number of created and disposed solutions), to maintain their individual object pools.

B.7.2 Design Patterns

Finally, we provide evidence that the proposed programming design patterns can further increase the computational performance of the API. This set of experiments requires extra control of the hardware to ensure consistency of results. Hence, we avoided languages with a built-in garbage collector such as Java and C#. We opted to employ the C language (C11 standard) and the gcc compiler (version 5.2.1) under Linux OS (kernel version 4.2.0-19-generic) due to explicit memory and processor control, the non-existence of a garbage collector and because we could easily extend our experiment to consider GPU processing.

Data Locality

To evaluate the data locality pattern we created two types of structures: the *local structure* with all variables contained locally, hence appearing consecutively in the physical memory, and the *external structure* where the variables are pointers to the actual values, representing references (in a similar fashion to object-oriented languages) and thus appearing on different sections of the memory.

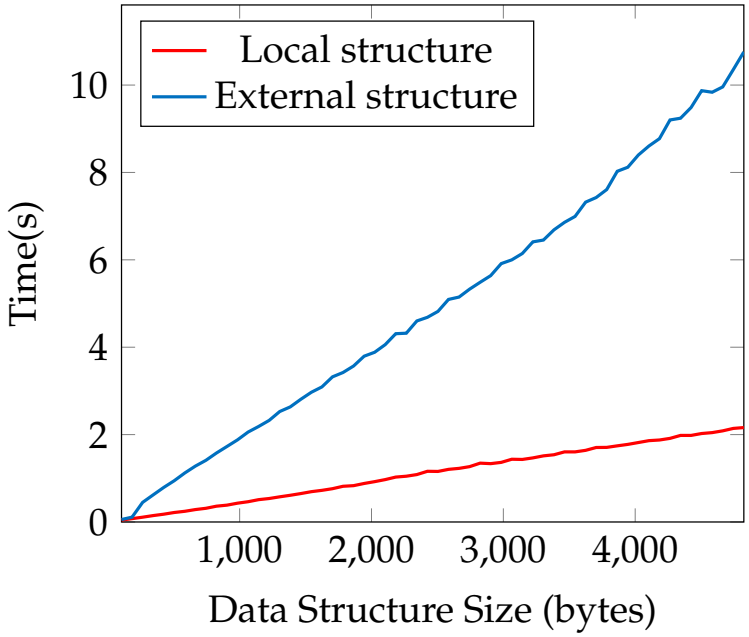


Figure B.11: Performance comparison of using values contained locally inside a structure (Local structure) against only the references to the values contained locally (External structure).

For the experiment, we created two arrays, one for each structure, each one containing 1,000,000 structure elements. For each array we sequentially iterated through all items, in sequential order, operating elementary operations on all variables within each element of the structure (thus accessing all values). We ran the iteration 1,000 times and computed the overall processing time.

Figure B.11 presents the results. We run the experiment for different sizes of structures, varying from 104 to 4,824 bytes. The x axis shows the size of the data structure in bytes. The y axis shows the average time of ten runs of the experiment, in seconds. It is clear that in both cases, as the size of the data increases, the access time increases in a linear trend. However, when we employ the use of local data, we have a reduction

of the computational time averaging 77%. It is important to notice that this is the comparison between the best and worst cases, hence these results present the maximum potential gain. However, in languages such as Java, this represents the difference between employing primitive data types (which will be stored locally within an object) and employing object data types (which will be stored as references). These results suggest that languages that are purely objected-oriented are not the best choice regarding performance for optimisation systems.

Matrix Ordering

In order to evaluate the matrix ordering design pattern we created a matrix of elements of type double. We then sequentially iterated through all elements of the matrix 1,000 times, accessing its elements in row order and computed the total processing time. Then we repeated the process using column order.

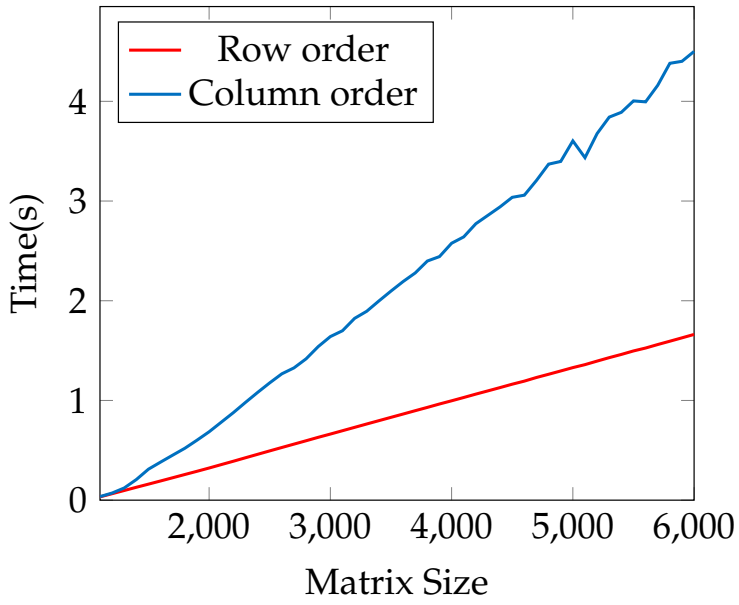


Figure B.12: Time comparison of performing sequential operations on a matrix using row order and using column order.

Figure B.12 presents the results of the matrix ordering design pattern. The x axis shows different matrix sizes in bytes, varying from 1,100 bytes to 6,000 bytes. The y axis shows the average time of ten runs of the experiments, in seconds. It is evident that performing the iteration in row order achieved significant performance gain compared to employing column order. The time reduction averages 57%. This gain happens

because the C compiler organises each line of the array sequentially in memory. Hence, when pulling data to the cache, several elements that will be read next are loaded into the cache as well. It is important to notice that these results depend on the programming language, and the results represent a potential gain. Because of that, the developer must ensure to use the right matrix ordering according to the specifications of the chosen programming language.

Memory Alignment

In order to assess the memory alignment, we created two types of structures, both consisting of a `char`, a `short`, and an `int` variable, totalling seven bytes. The *aligned* structure contained an extra byte, a padding byte, to round the structure size to eight. The *misaligned* structure did not contain the padding byte, and we ensured that by providing the `__attribute__((packed))` directive to the compiler because the C compiler automatically aligns misaligned structures.

The performance test used is similar to the one employed on the data locality pattern: we created two arrays, one for each type of structure, each one containing 1,000,000 elements. For each array we iterated through all the elements, in order, performing elementary operations on all variables of each structure element. We repeated the process 1,000 times and computed the total processing time.

Because the performance of memory alignment is dependent on the CPU architecture, we performed the experiments on multiple CPUs, on multiple architectures, and recorded the computation times. Also, we used the same OS and gcc versions in all experiments, and each executable was locally compiled to obtain architecture-specific optimisations. Also, because nowadays GPU computing is used for high-performance computing [127], we performed the experiments on two GPUs as well.

Table B.1 presents the results. The first column lists the manufacturer of the chipset: for CPUs, we have Intel and AMD and for GPU we have Nvidia. Column *Model* gives the specific model of the processor and column *Architecture Codename* gives the version of the architecture used in the CPU (note that different processors may share the same architecture). Column *Year* shows the year in which that processor was released. The double column *Time(s)* shows the computational time (in seconds) for the aligned and misaligned structures. The last column gives the perfor-

Table B.1: Results for the memory alignment pattern.

Manufacturer	Model	Architecture Codename	Year	Time (s)		Difference in Performance
				Aligned	Misaligned	
Intel	Core i7-4870HQ	Crystal Well	2014	4.4	5.1	17.4%
Intel	Core i5-4460	Haswell	2014	4.6	5.6	21.9%
Intel	Core i7-4770	Haswell	2013	4.0	4.9	21.8%
Intel	Core i5-4200U	Haswell	2013	6.5	8.0	22.4%
Intel	Xeon E5-2407 v2	Ivy Bridge EN	2014	8.6	10.4	21.3%
Intel	Core i5-3340	Ivy Bridge	2013	4.8	5.8	19.9%
AMD	A10-6800K	Richland	2013	6.2	8.1	30.0%
Intel	Core i7-3632QM	Ivy Bridge	2012	4.9	6.0	22.6%
Intel	Core i7-3820	Sandy Bridge E	2012	4.7	6.6	38.4%
AMD	FX-6100	Bulldozer	2012	7.8	12.0	53.5%
AMD	A6-3670	K10	2011	13.3	17.0	27.6%
Intel	Pentium Dual T2390	Merom	2007	17.9	25.9	44.8%
Nvidia	GTX 980M (CUDA 5.2)	Maxwell	2014	0.000314	0.000320	1.9%
Nvidia	GTX 750 (CUDA 5.0)	Maxwell	2014	0.000584	0.000586	0.3%

mance gain by applying aligned structures.

The results show that, for CPUs, there are benefits from using aligned structures. It is clear that on older architectures that gain is more substantial, going up to 53.5% on the AMD Bulldozer (2012) and 44.8% on the Intel Merom (2007). However, it is also clear that on recent architectures, the manufacturers are minimising the impact of using misaligned structures. The two bottom rows of the table present the GPU results. It is important to notice that we made full use of the GPU multithread computing capability in these experiments. Hence, the processing times are just a fraction of the processing times for the CPUs. It is clear that GPUs do not suffer significant impairments from using misaligned structures. However, it might be the case that the CUDA compiler did not recognise the pack directive sent to the compiler and the data structures were padded regardless.

Even though modern processors handle misaligned structures more efficiently, there is substantial performance gain from using aligned data. Nonetheless, to assess memory misalignment, we had to force the compiler to accept misaligned structures. Most programming languages automatically add padding bytes to align such structures, hence, unless a developer is using legacy systems or programming on low-level languages such as assembly language, memory alignment issues can be ignored.

Overall

The previous experiments showed the potential gain from using the computational efficiency design patterns presented in this work. However, these experiments consider scenarios where the evaluated features are isolated, which is unlikely to happen in practice. Hence, the answer to questions such as *"how important is the application of the proposed design patterns on optimisation algorithms?"* is still not clear. Therefore, we implemented a straightforward optimisation algorithm for a simple, well-known problem in two versions: one employing data locality and matrix ordering, and another one not using these patterns. We opted to ignore the memory alignment for the reasons aforementioned.

The optimisation problem considered in this experiment is a Multi-dimensional Knapsack Problem (MKP) [152] with n items $(i = 1, \ldots, n)$, m weights w_j^i $(j = 1, \ldots, m)$ and p profits c_k^i $(k = 1, \ldots, p)$. A set of items must be selected for packing in the knapsack in order to maximise the p profits while not exceeding the capacities W_j of the knapsack. This problem can be formulated as follows:

$$\begin{aligned} \text{maximise} \quad & \sum_{i=1}^{n} c_k^i x_i && k = 1, \ldots, p \\ \text{subject to} \quad & \sum_{i=1}^{n} w_j^i x_i \leq W_j && j = 1, \ldots, m \\ & x_i \in 0, 1 && i = 1, \ldots, n \end{aligned}$$

The solution representation for the above problem is a straightforward binary array representing the decision variables. The problem data was generated with random values of a uniform distribution [175]. The optimisation algorithm is a simple random search [202], where solutions are randomly generated, evaluated, and if the current solution is the best so far, then it is kept, otherwise it is discarded. The stop criterion is when the algorithm reaches a pre-defined number of solutions generated.

The experiment consists of performing the random search for 10,000 solution generations and recording the total computation time. We compared an implementation A, using data locality and matrix ordering against a second implementation B that ignored these patterns. The remaining of the application was kept identical. We performed the experiments for different number of items varying from 100 to 10,000. Figures B.13 and

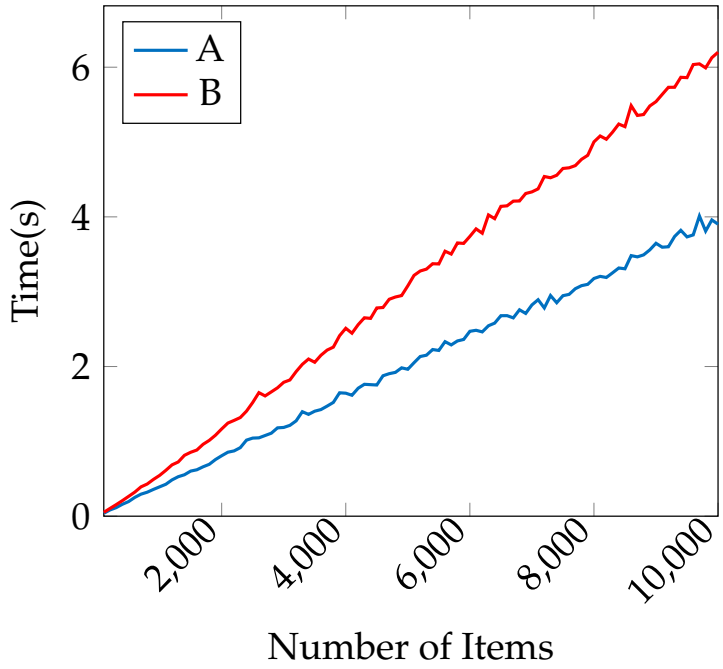


Figure B.13: Results for 10,000 generations of MKP solutions using a random search employing an implementation using the data locality and matrix ordering design patterns (A) and an implementation not using these patterns (B).

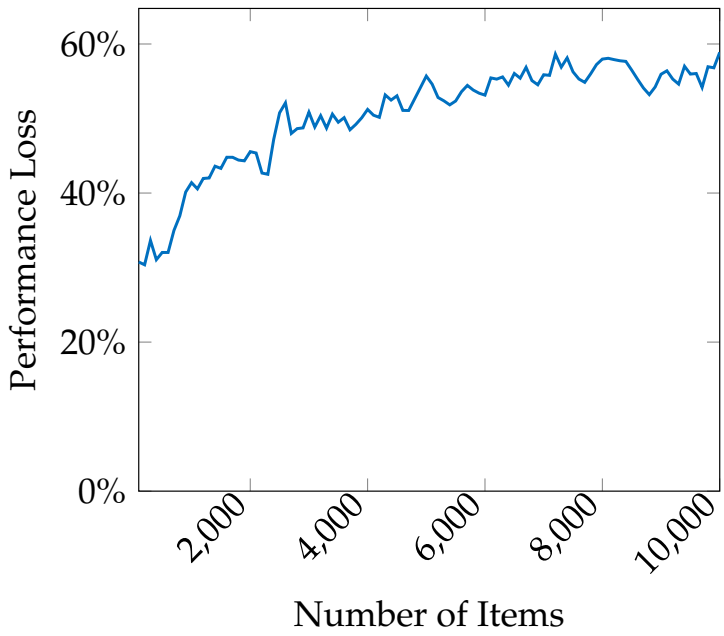


Figure B.14: Performance loss for not employing the data locality and matrix ordering design patterns.

B.14 present the results.

Figure B.13 shows that by employing the patterns, the computational time is consistently smaller. Figure B.14 indicates that the performance loss of not using the patterns can be as high as 60% and it increases with the size of the problem instances. Therefore, even in an optimisation algorithm environment, caring for matrix ordering and data locality can outcome substantial performance gains. Certainly, when combining these techniques with the SB, a higher performance increase can be achieved. Hence we recommend the use of the proposed techniques based on our experiments and the results obtained.

B.8 Conclusion

In order to illustrate the application of the framework and the API, we used a real-world R&D project to develop a decision support system based on an optimisation solution for the home healthcare problem, originating from a partnership between The University of Nottingham and a software development company. We discussed the benefits identified during the development of this project, such as:

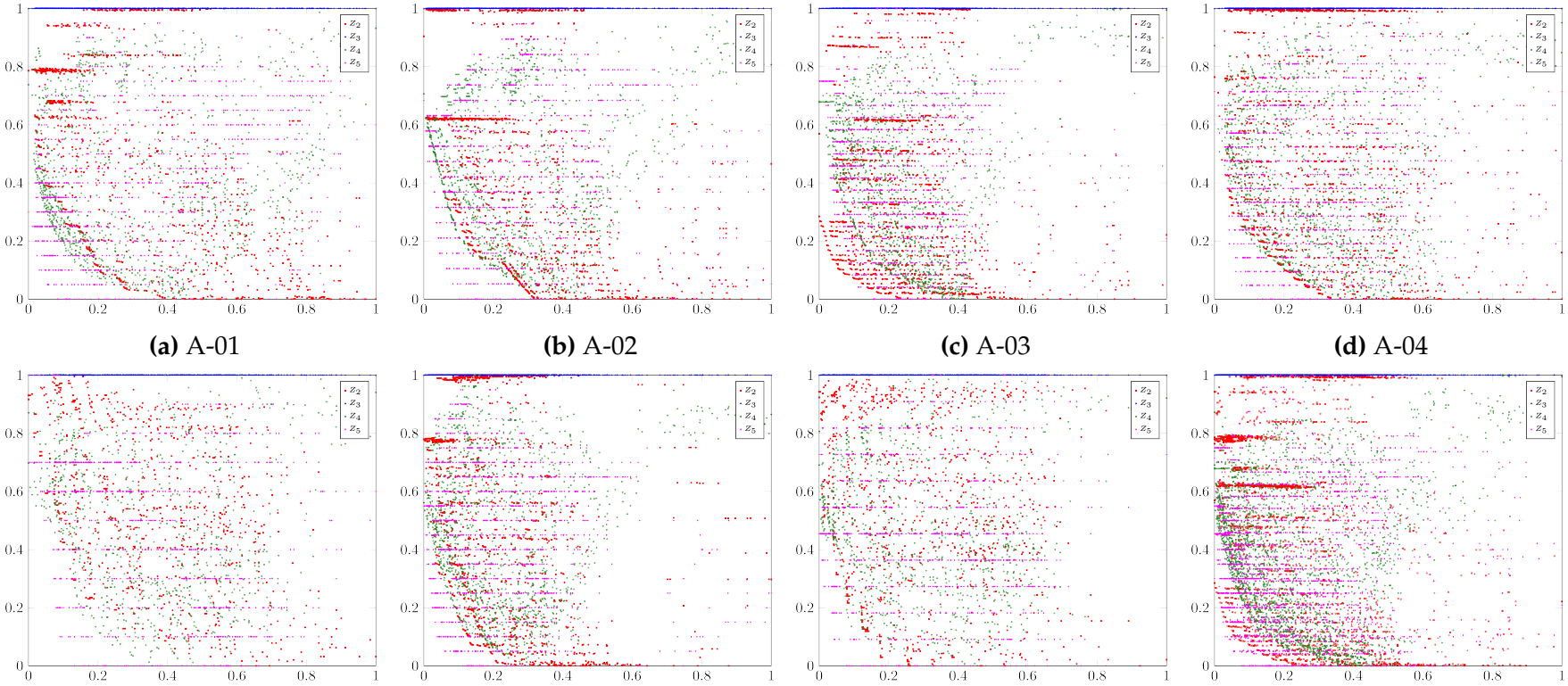
- the **reduction of rework**, because researchers and development team were using common implementations;
- increase of the **research team efficiency**, because they were not required to implement necessary (but time-consuming) programs to parse and process data;
- increase of **computational efficiency**, because all researchers used the best implementations known for the data structures and data access; and
- the guarantee of a **fair comparison** between the different techniques developed.

We also presented the empirical results from computational experiments with the solution builder, using optimisation algorithms to show that substantial performance can be gained by employing the technique instead of relying on garbage collection. We also demonstrated that the utilisation of the proposed design patterns further increases the performance of optimisation algorithms.

In summary, the methodologies presented in this work is a tool for both researchers and practitioners to obtain improved results on R&D projects based on optimisation problems. Also, the API helps to enhance the performance of the software developed while researchers with weaker programming skills can still achieve improved computational performance in their algorithms. Future work includes applying the methodology on additional R&D projects and formally measure the impact of using the API in the software development life cycle.

Appendix C

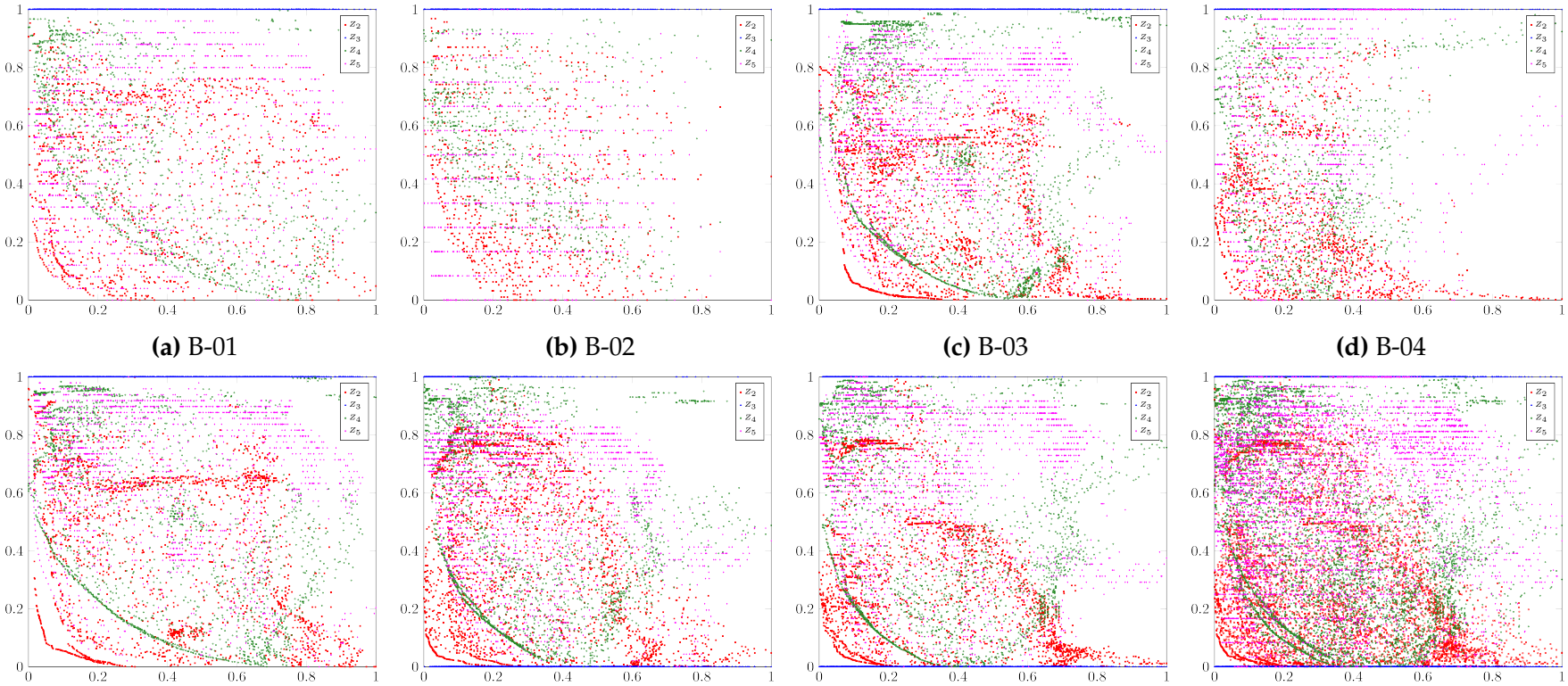
Scatter Plots of the Home Healthcare Datasets



(a) A-01 **(b)** A-02 **(c)** A-03 **(d)** A-04

(e) A-05 **(f)** A-06 **(g)** A-07 **(h)** Combined A

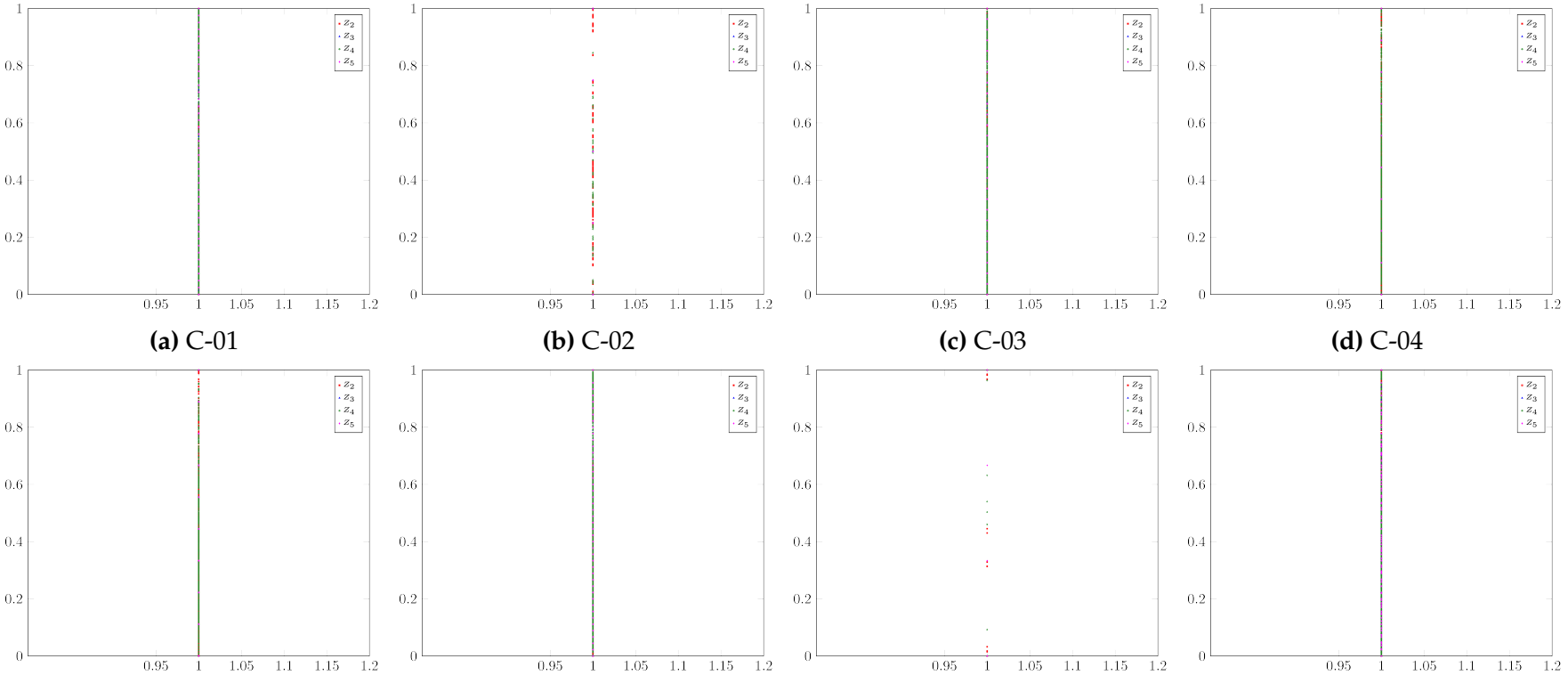
Figure C.1: Scatter plots of each of the WSRP-A instances, showing the relationship between objective Z_1 and each of the other four objectives. In each case, the x axis shows the value of objective Z_1 and the y axis shows the values of each of the other objectives in different colours.



(a) B-01 **(b)** B-02 **(c)** B-03 **(d)** B-04

(e) B-05 **(f)** B-06 **(g)** B-07 **(h)** Combined B

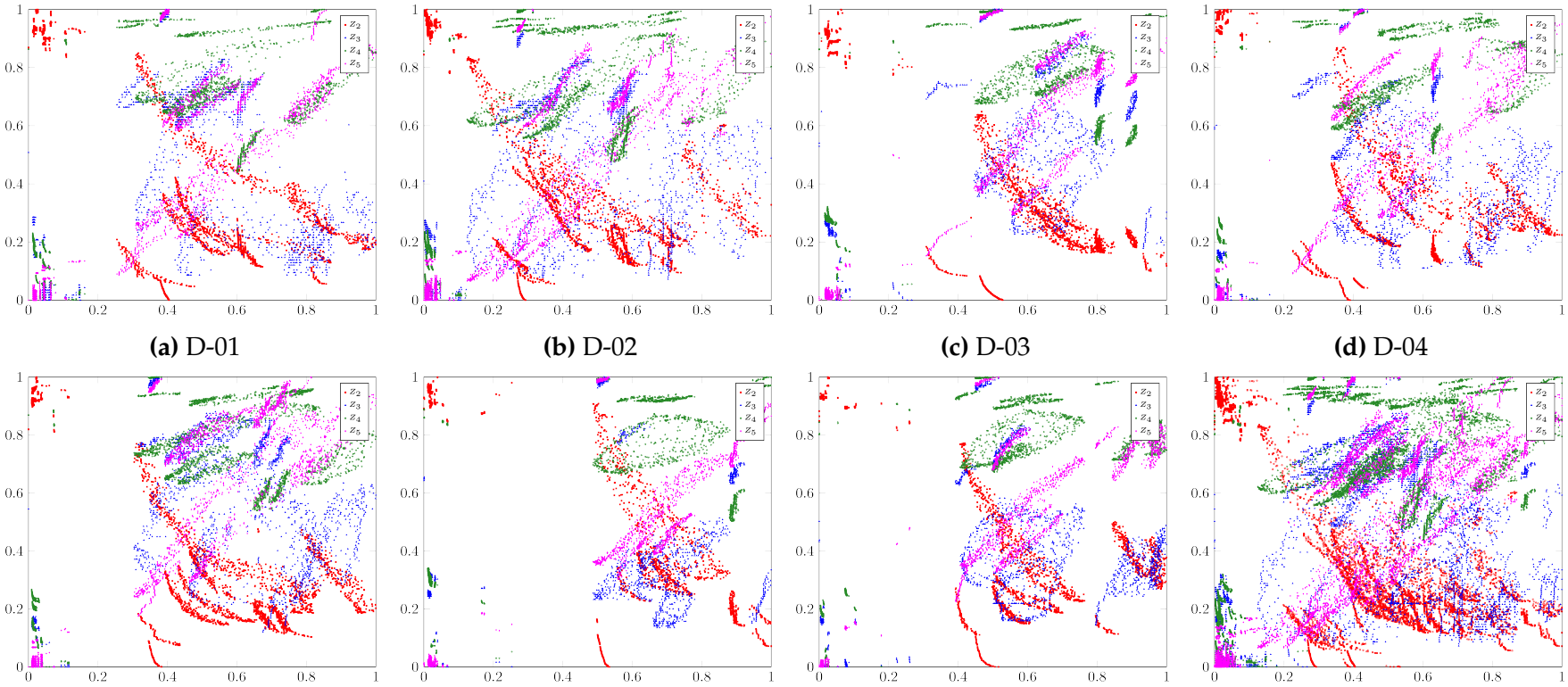
Figure C.2: Scatter plots of each of the WSRP-B instances, showing the relationship between objective Z_1 and each of the other four objectives. In each case, the x axis shows the value of objective Z_1 and the y axis shows the values of each of the other objectives in different colours.



(a) C-01 **(b)** C-02 **(c)** C-03 **(d)** C-04

(e) C-05 **(f)** C-06 **(g)** C-07 **(h)** Combined C

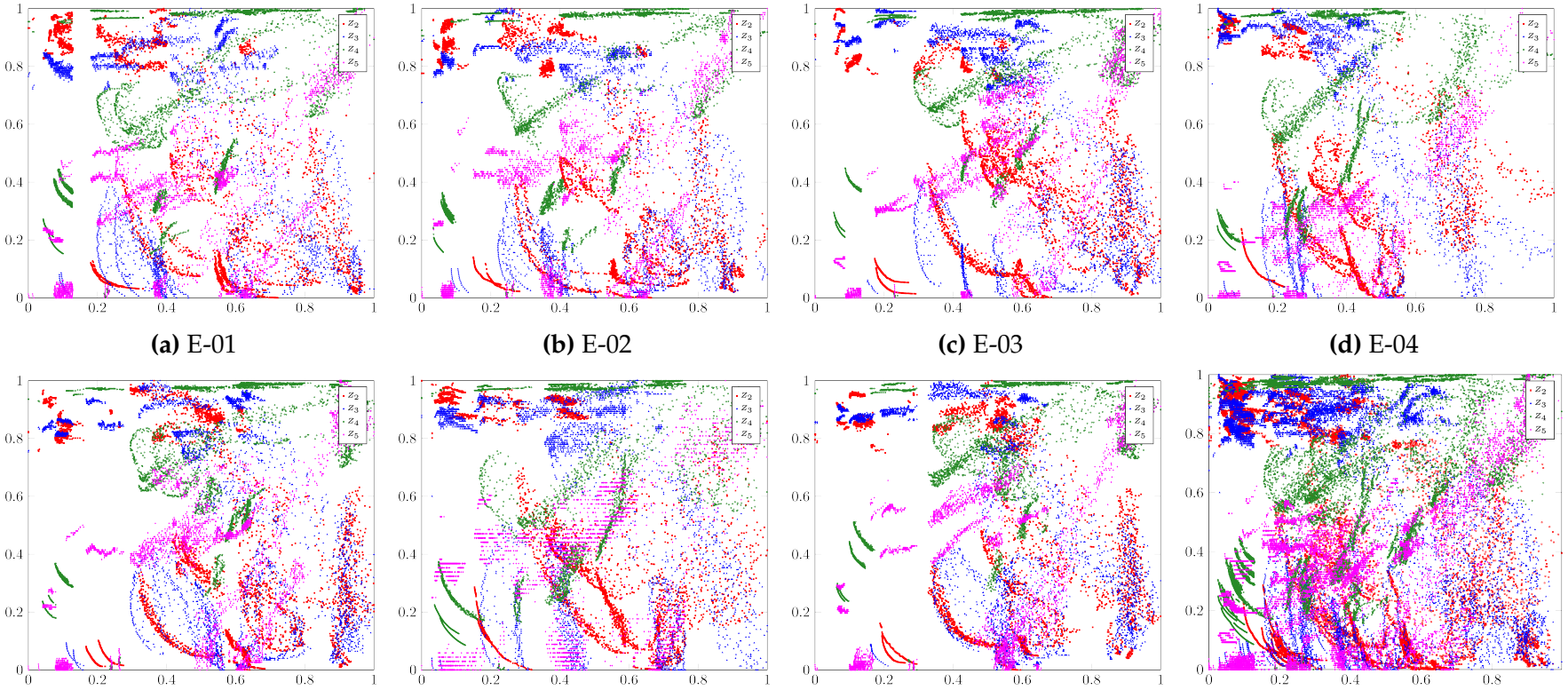
Figure C.3: Scatter plots of each of the WSRP-C instances, showing the relationship between objective Z_1 and each of the other four objectives. In each case, the x axis shows the value of objective Z_1 and the y axis shows the values of each of the other objectives in different colours.



(a) D-01 **(b)** D-02 **(c)** D-03 **(d)** D-04

(e) D-05 **(f)** D-06 **(g)** D-07 **(h)** Combined D

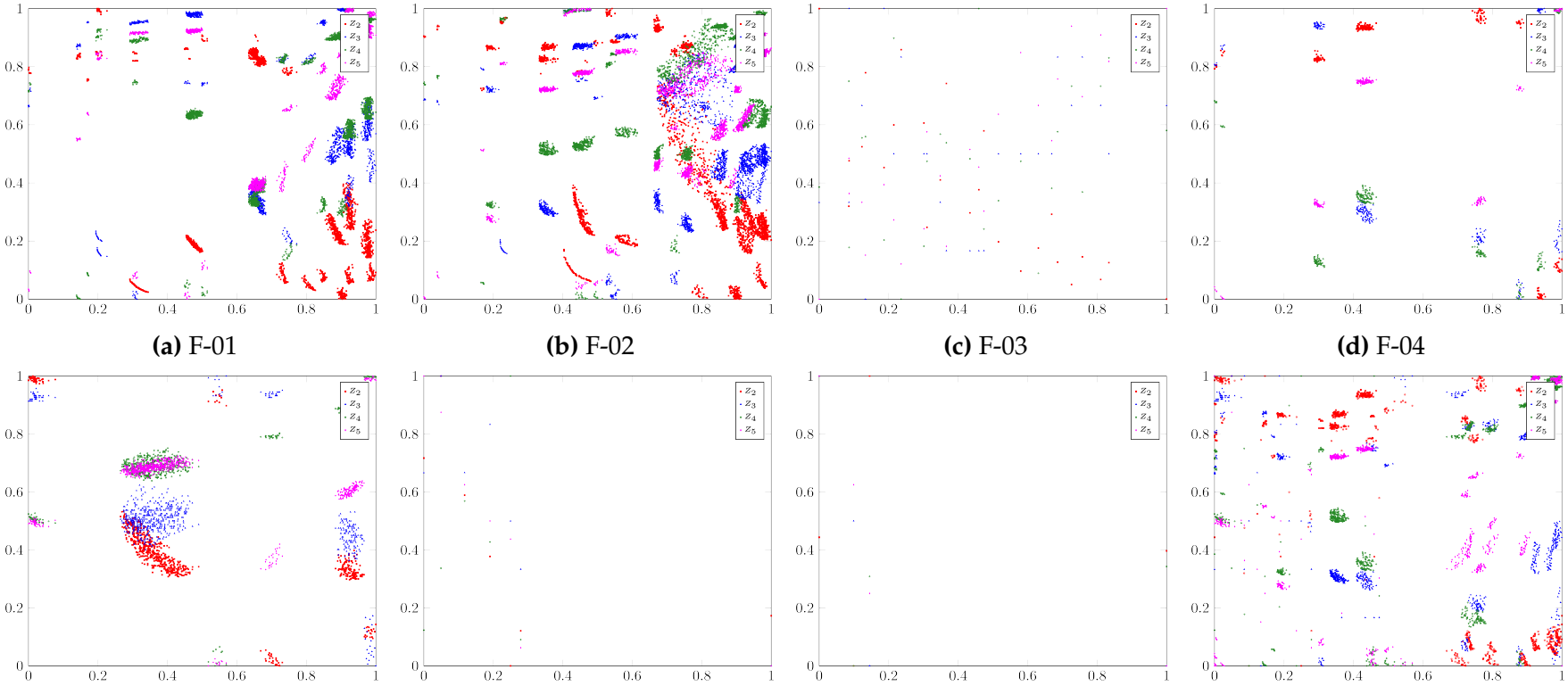
Figure C.4: Scatter plots of each of the WSRP-D instances, showing the relationship between objective Z_1 and each of the other four objectives. In each case, the x axis shows the value of objective Z_1 and the y axis shows the values of each of the other objectives in different colours.



(a) E-01 **(b)** E-02 **(c)** E-03 **(d)** E-04

(e) E-05 **(f)** E-06 **(g)** E-07 **(h)** Combined E

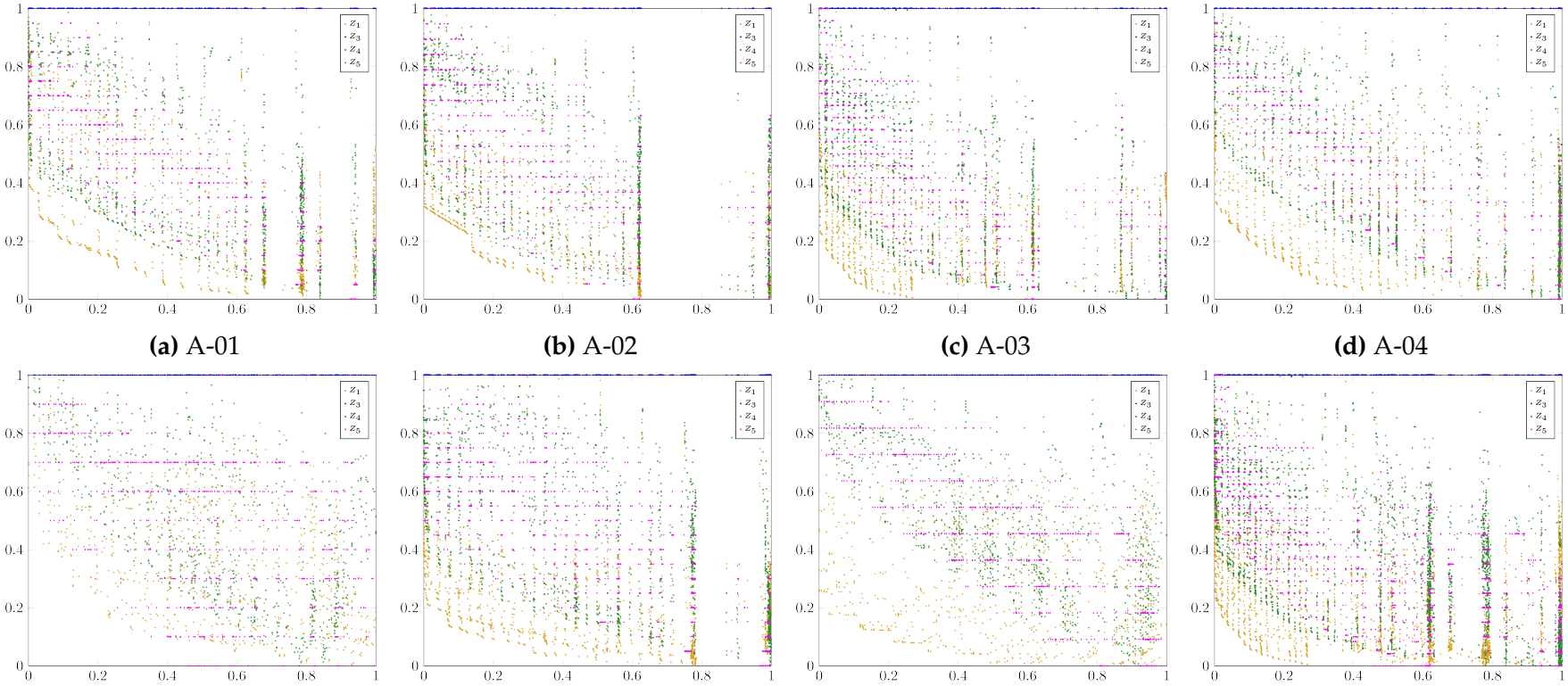
Figure C.5: Scatter plots of each of the WSRP-E instances, showing the relationship between objective Z_1 and each of the other four objectives. In each case, the x axis shows the value of objective Z_1 and the y axis shows the values of each of the other objectives in different colours.



(a) F-01 **(b)** F-02 **(c)** F-03 **(d)** F-04

(e) F-05 **(f)** F-06 **(g)** F-07 **(h)** Combined F

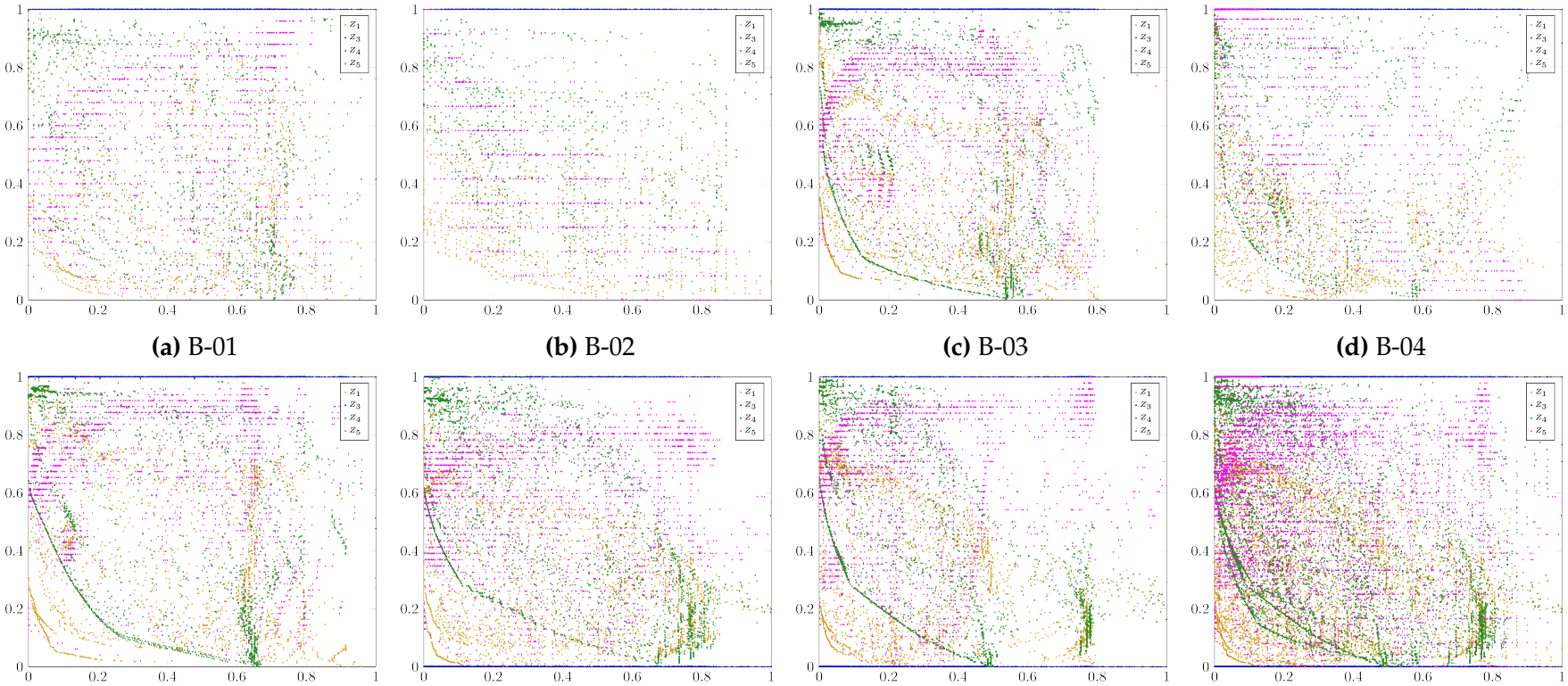
Figure C.6: Scatter plots of each of the WSRP-F instances, showing the relationship between objective Z_1 and each of the other four objectives. In each case, the x axis shows the value of objective Z_1 and the y axis shows the values of each of the other objectives in different colours.



(a) A-01 **(b)** A-02 **(c)** A-03 **(d)** A-04

(e) A-05 **(f)** A-06 **(g)** A-07 **(h)** Combined A

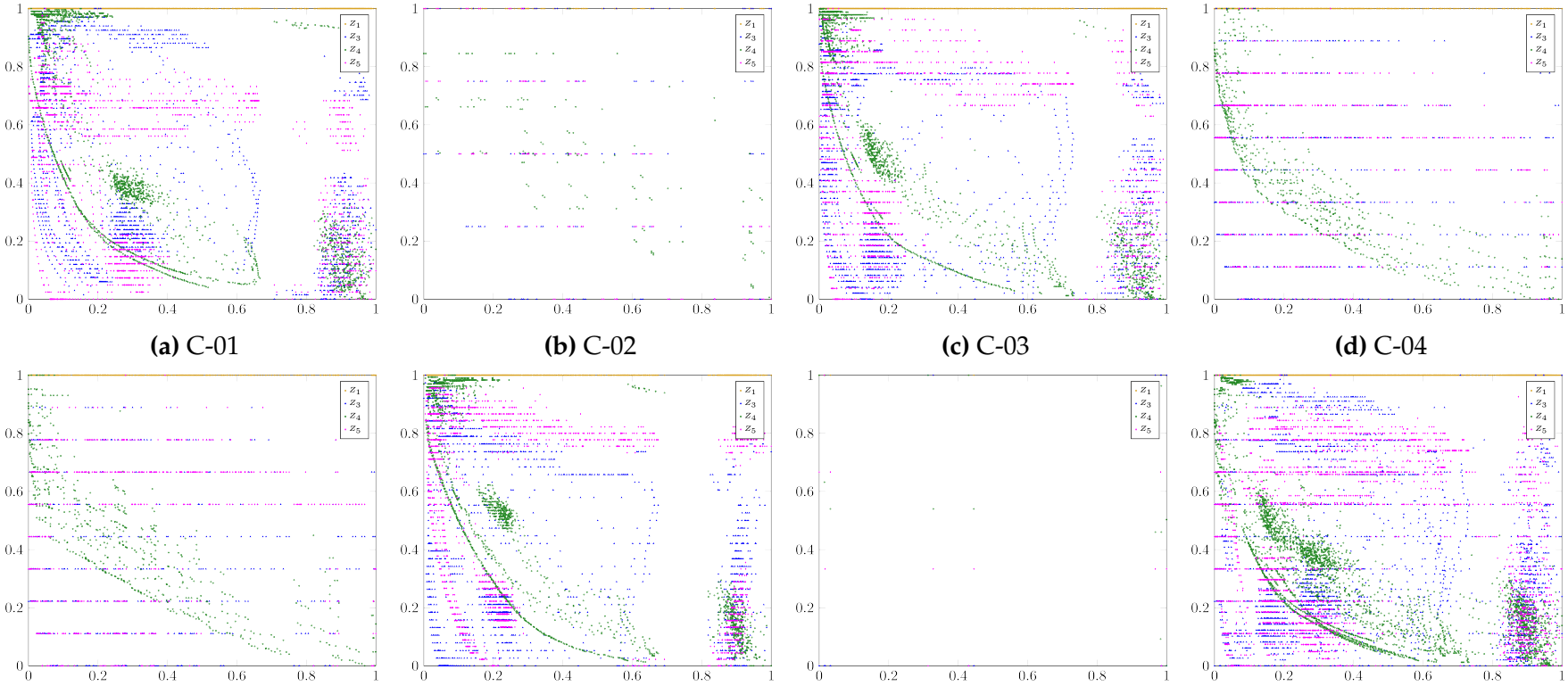
Figure C.7: Scatter plots of each of the WSRP-A instances, showing the relationship between objective Z_2 and each of the other four objectives. In each case, the x axis shows the value of objective Z_2 and the y axis shows the values of each of the other objectives in different colours.



(a) B-01 **(b)** B-02 **(c)** B-03 **(d)** B-04

(e) B-05 **(f)** B-06 **(g)** B-07 **(h)** Combined B

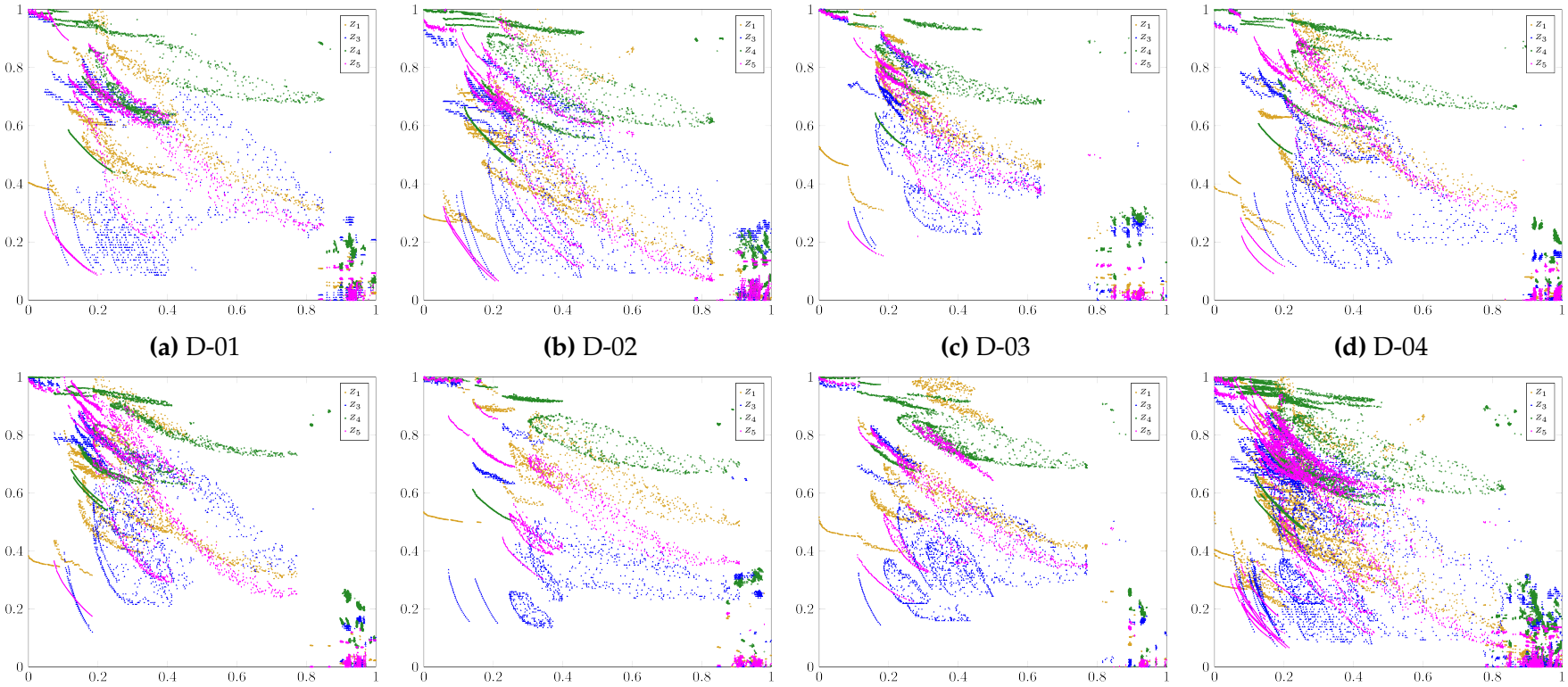
Figure C.8: Scatter plots of each of the WSRP-A instances, showing the relationship between objective Z_2 and each of the other four objectives. In each case, the x axis shows the value of objective Z_2 and the y axis shows the values of each of the other objectives in different colours.



(a) C-01 **(b)** C-02 **(c)** C-03 **(d)** C-04

(e) C-05 **(f)** C-06 **(g)** C-07 **(h)** Combined C

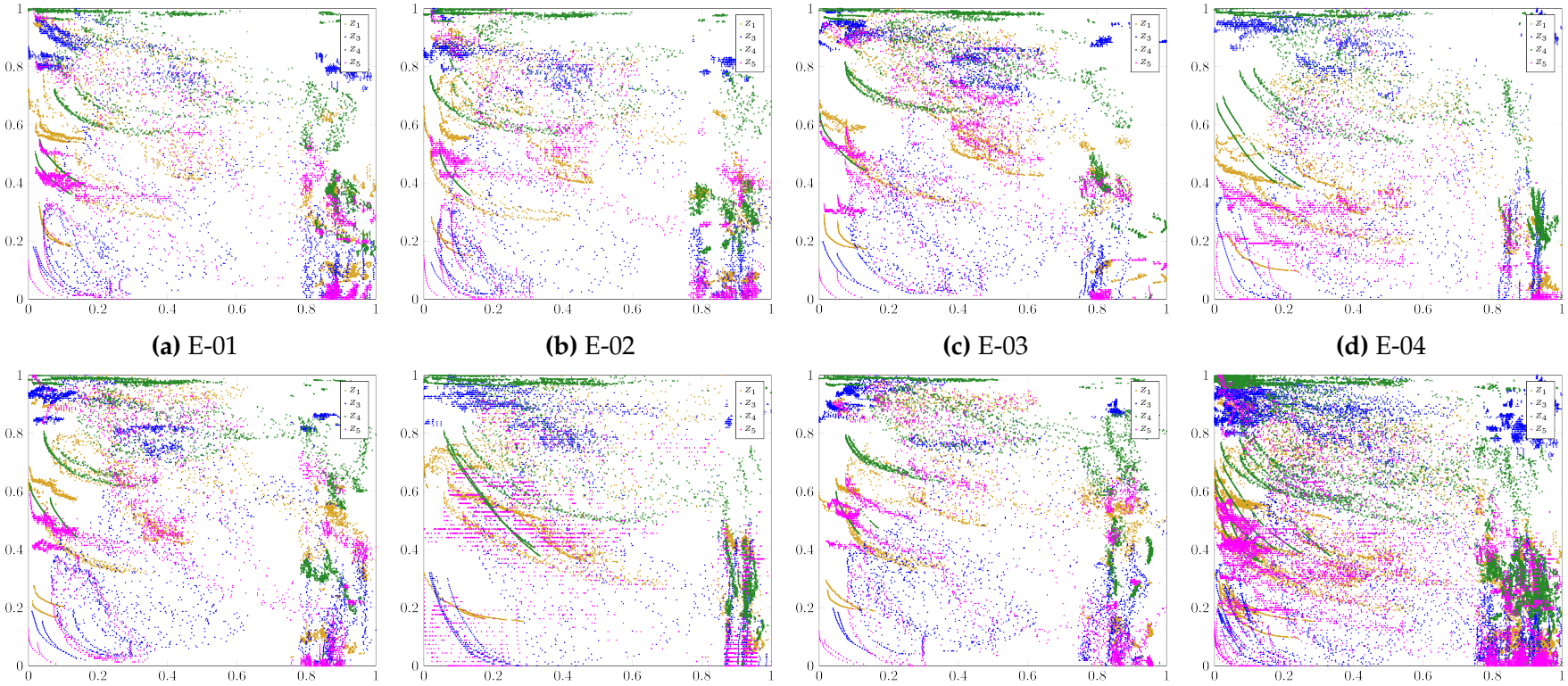
Figure C.9: Scatter plots of each of the WSRP-C instances, showing the relationship between objective Z_2 and each of the other four objectives. In each case, the x axis shows the value of objective Z_2 and the y axis shows the values of each of the other objectives in different colours.



(a) D-01 **(b)** D-02 **(c)** D-03 **(d)** D-04

(e) D-05 **(f)** D-06 **(g)** D-07 **(h)** Combined D

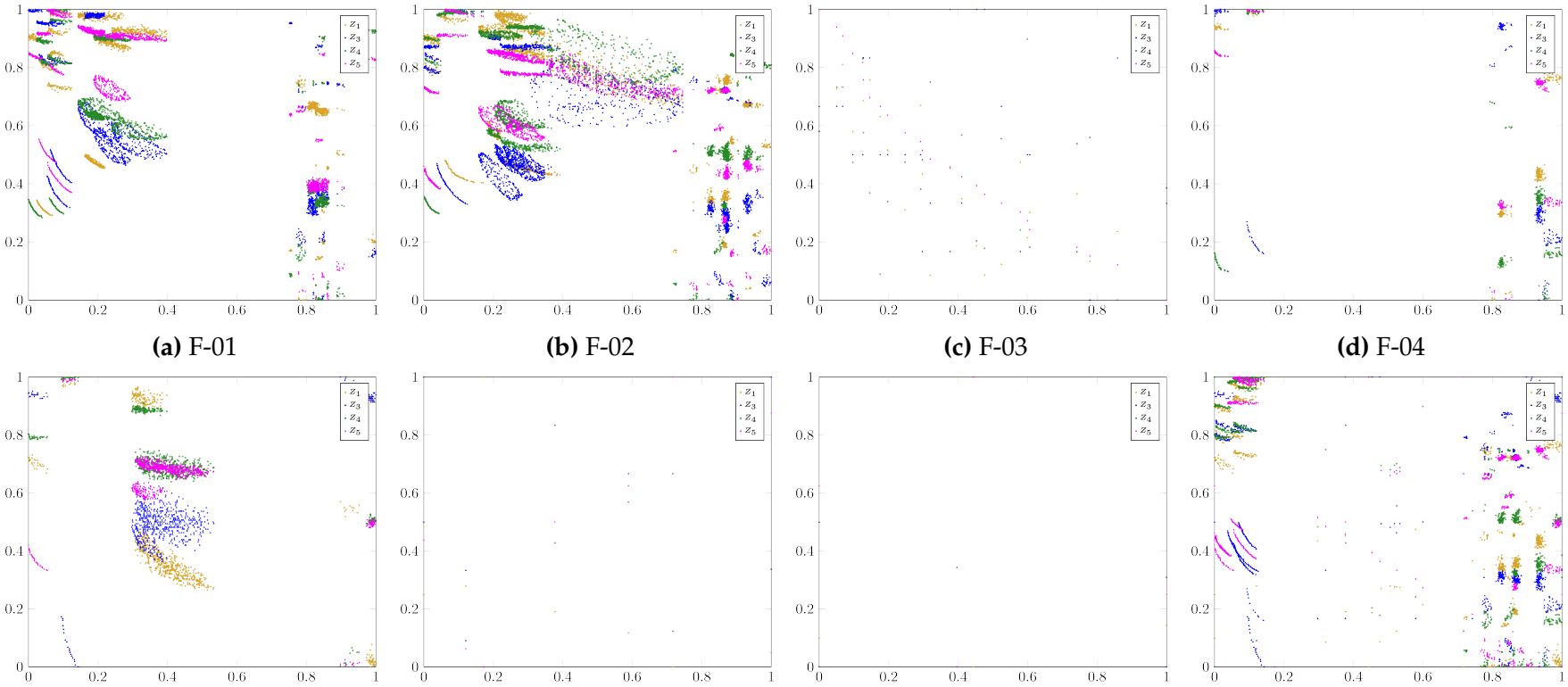
Figure C.10: Scatter plots of each of the WSRP-D instances, showing the relationship between objective Z_2 and each of the other four objectives. In each case, the x axis shows the value of objective Z_2 and the y axis shows the values of each of the other objectives in different colours.



(a) E-01 **(b)** E-02 **(c)** E-03 **(d)** E-04

(e) E-05 **(f)** E-06 **(g)** E-07 **(h)** Combined E

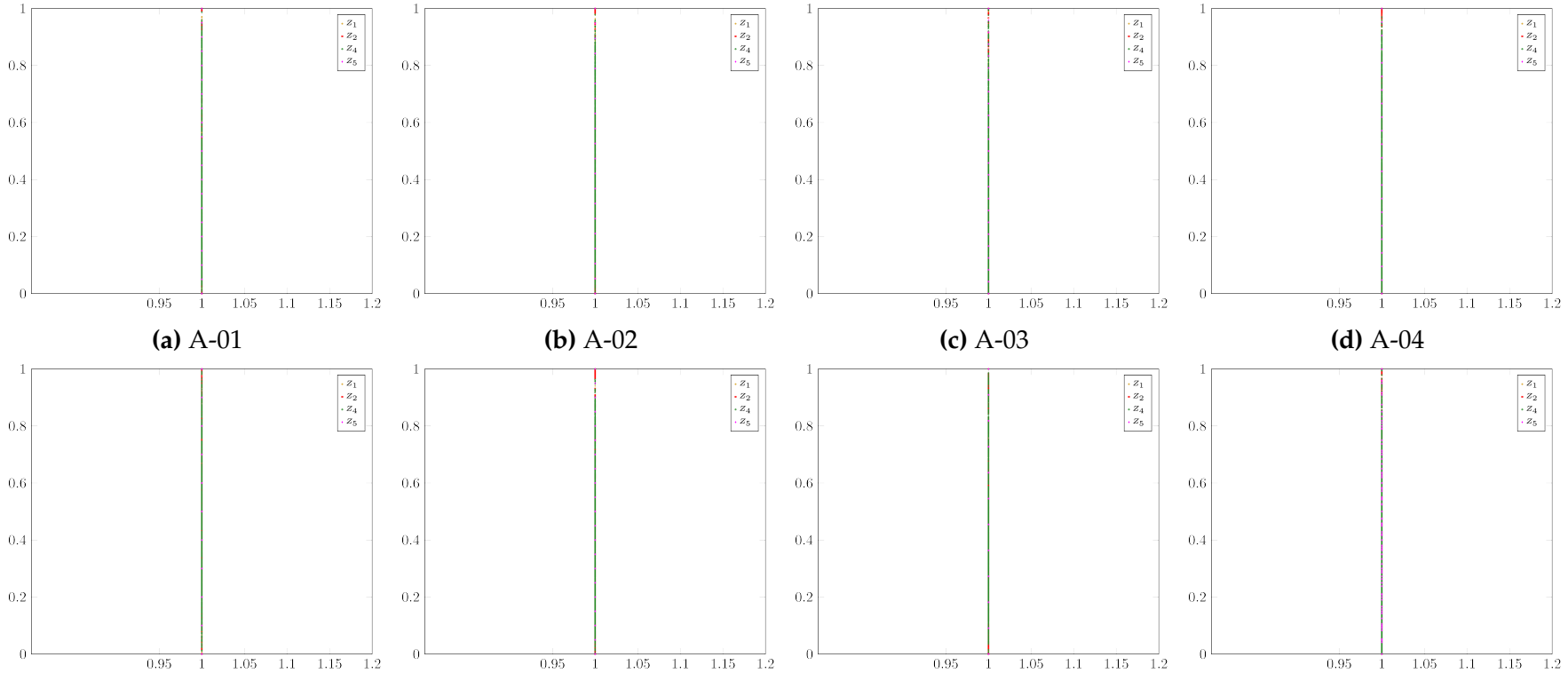
Figure C.11: Scatter plots of each of the WSRP-E instances, showing the relationship between objective Z_2 and each of the other four objectives. In each case, the x axis shows the value of objective Z_2 and the y axis shows the values of each of the other objectives in different colours.



(a) F-01 **(b)** F-02 **(c)** F-03 **(d)** F-04

(e) F-05 **(f)** F-06 **(g)** F-07 **(h)** Combined F

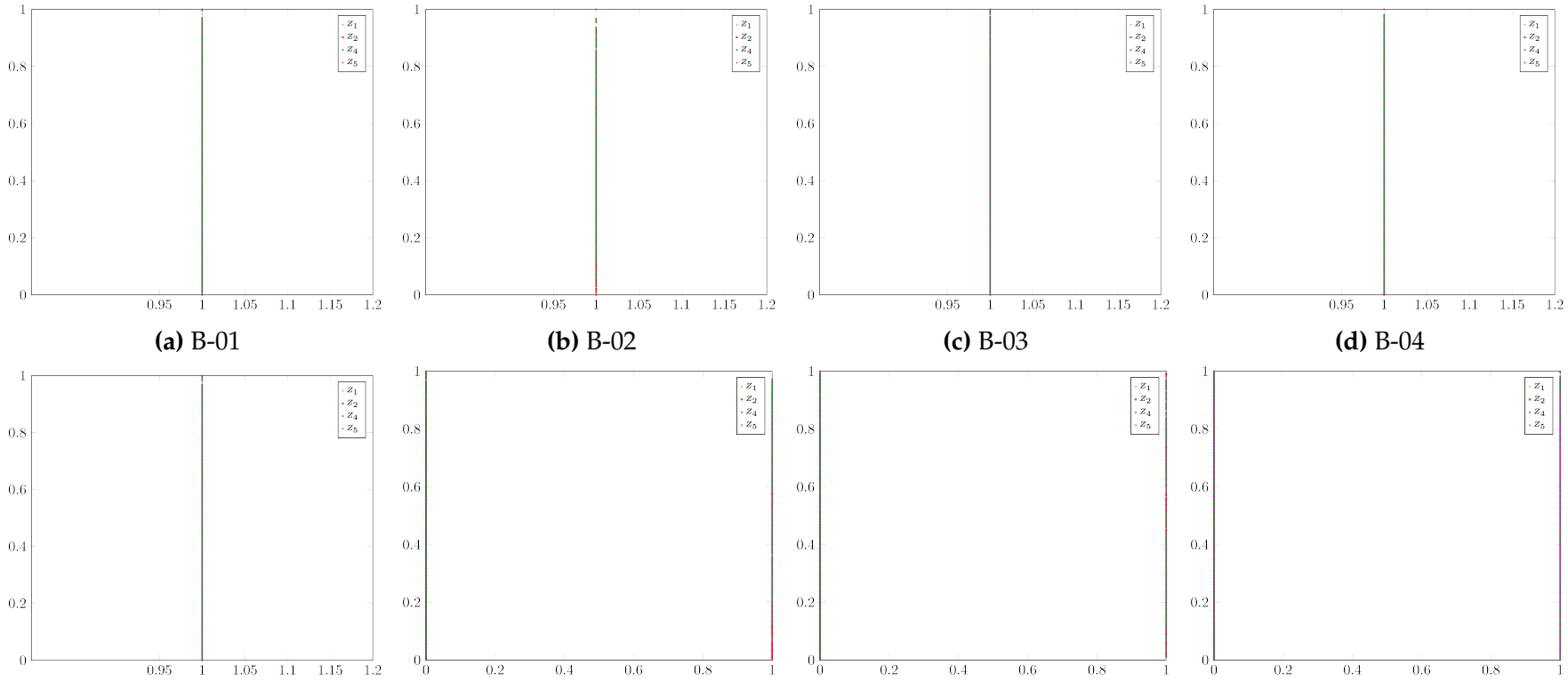
Figure C.12: Scatter plots of each of the WSRP-F instances, showing the relationship between objective Z_2 and each of the other four objectives. In each case, the x axis shows the value of objective Z_2 and the y axis shows the values of each of the other objectives in different colours.



(a) A-01 **(b)** A-02 **(c)** A-03 **(d)** A-04

(e) A-05 **(f)** A-06 **(g)** A-07 **(h)** Combined A

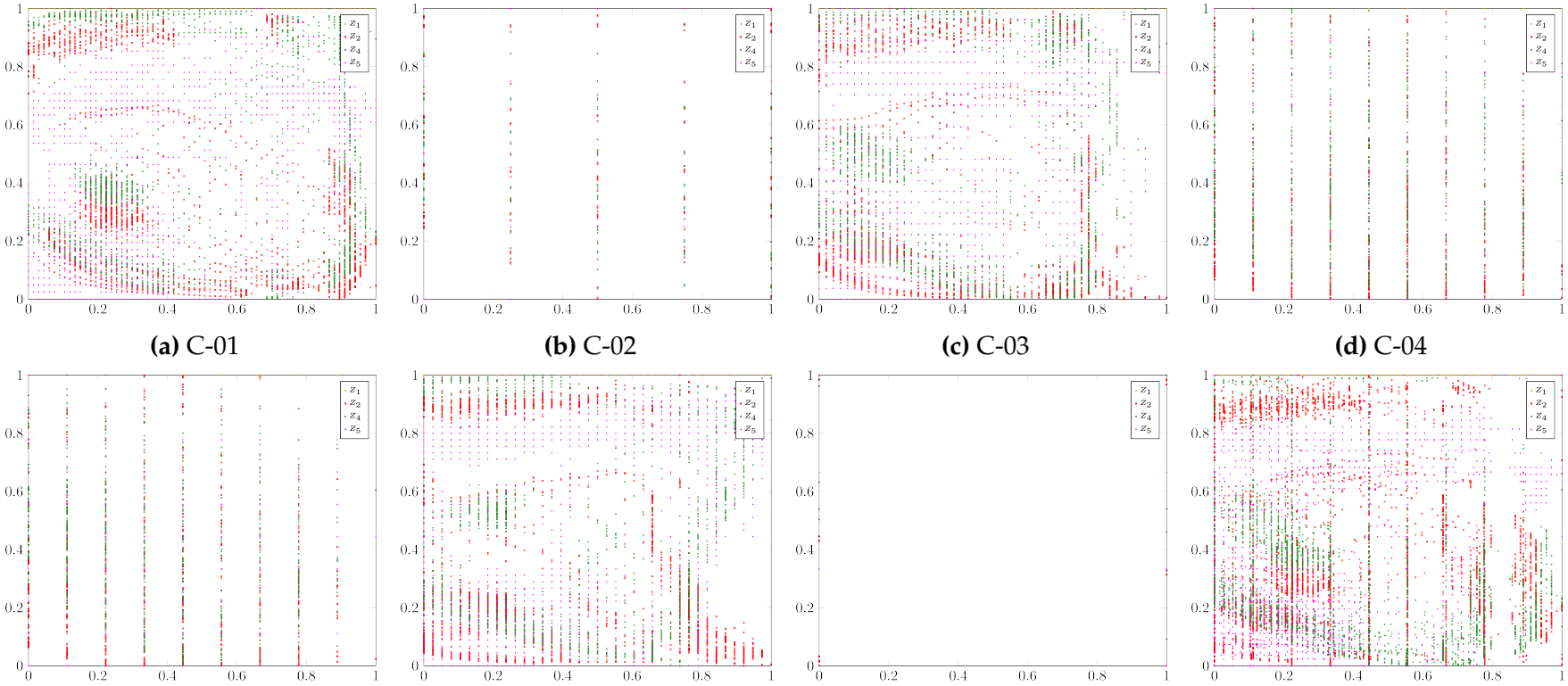
Figure C.13: Scatter plots of each of the WSRP-A instances, showing the relationship between objective Z_3 and each of the other four objectives. In each case, the x axis shows the value of objective Z_3 and the y axis shows the values of each of the other objectives in different colours.



(a) B-01 **(b)** B-02 **(c)** B-03 **(d)** B-04

(e) B-05 **(f)** B-06 **(g)** B-07 **(h)** Combined B

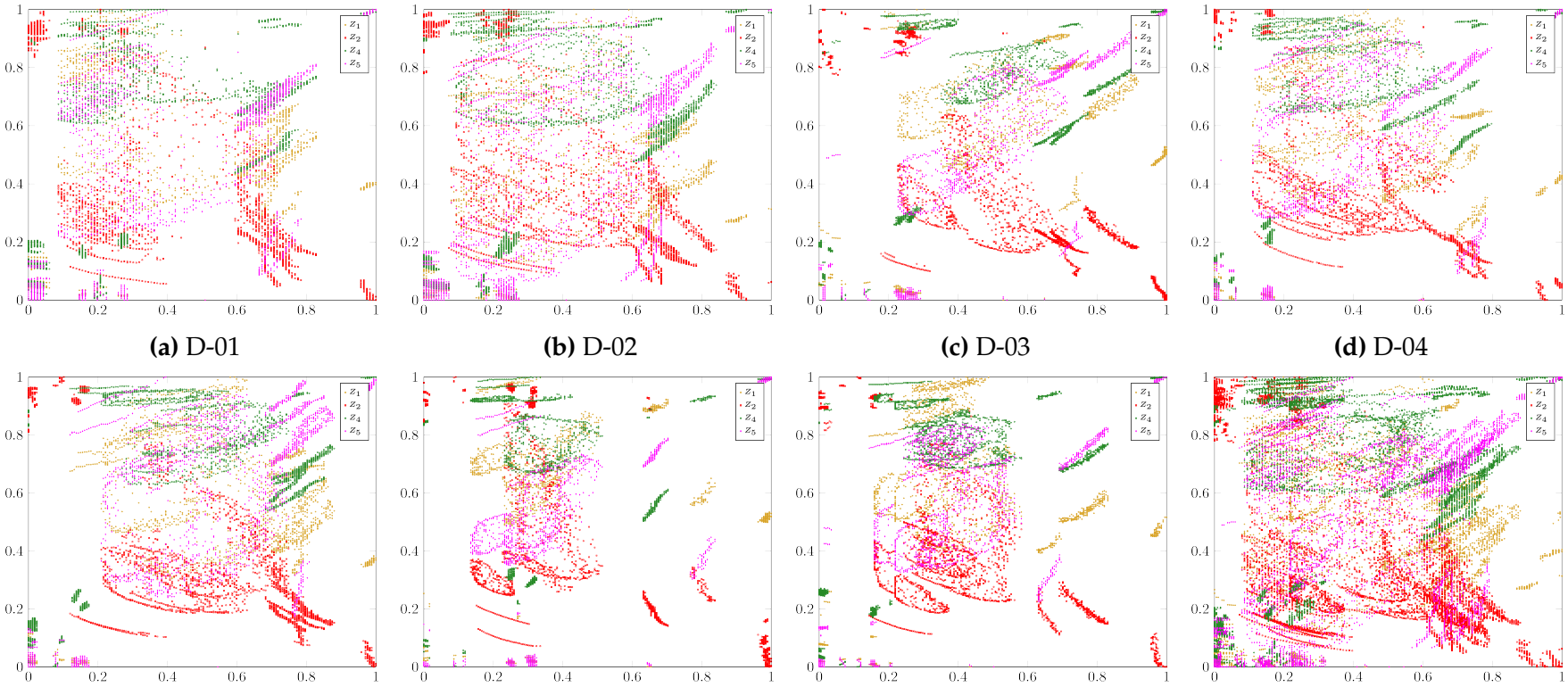
Figure C.14: Scatter plots of each of the WSRP-B instances, showing the relationship between objective Z_3 and each of the other four objectives. In each case, the x axis shows the value of objective Z_3 and the y axis shows the values of each of the other objectives in different colours.



(a) C-01 **(b)** C-02 **(c)** C-03 **(d)** C-04

(e) C-05 **(f)** C-06 **(g)** C-07 **(h)** Combined C

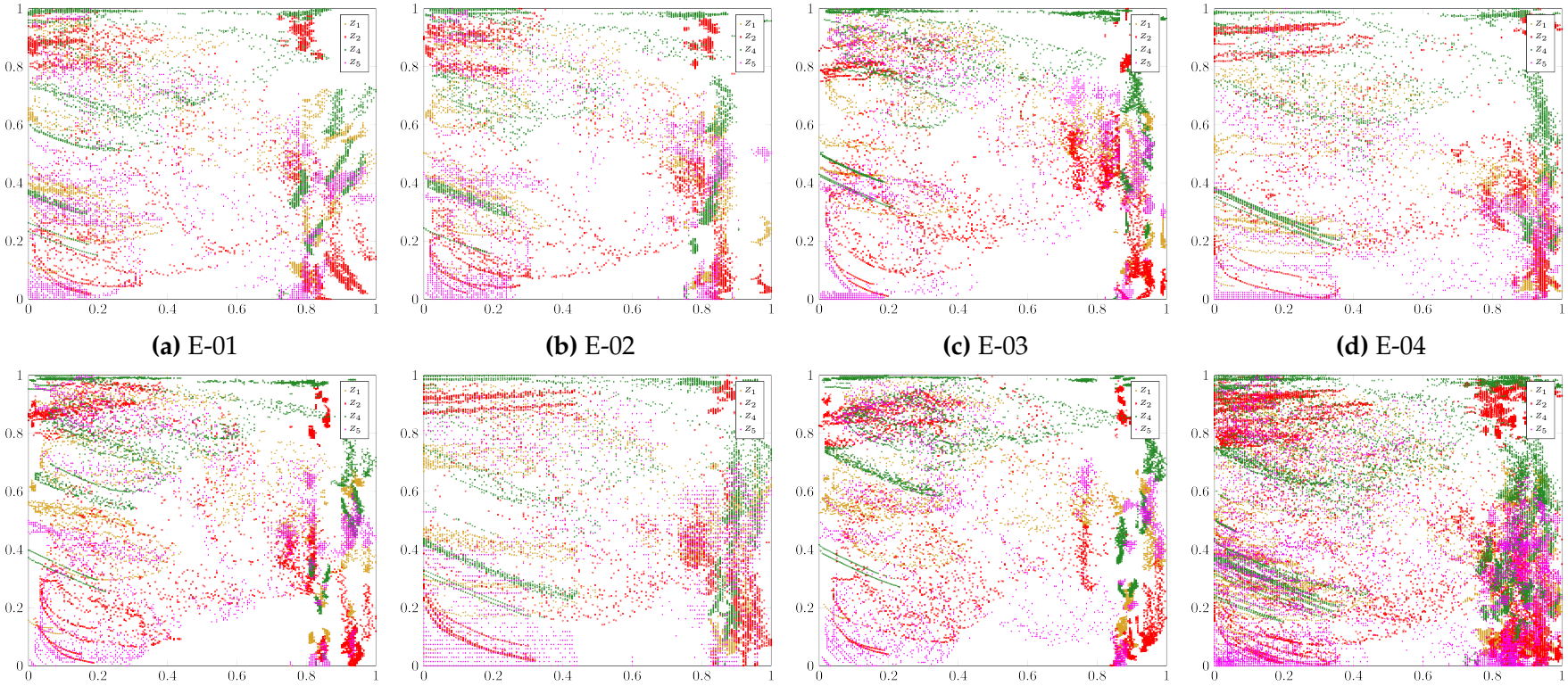
Figure C.15: Scatter plots of each of the WSRP-C instances, showing the relationship between objective Z_3 and each of the other four objectives. In each case, the x axis shows the value of objective Z_3 and the y axis shows the values of each of the other objectives in different colours.



(a) D-01 **(b)** D-02 **(c)** D-03 **(d)** D-04

(e) D-05 **(f)** D-06 **(g)** D-07 **(h)** Combined D

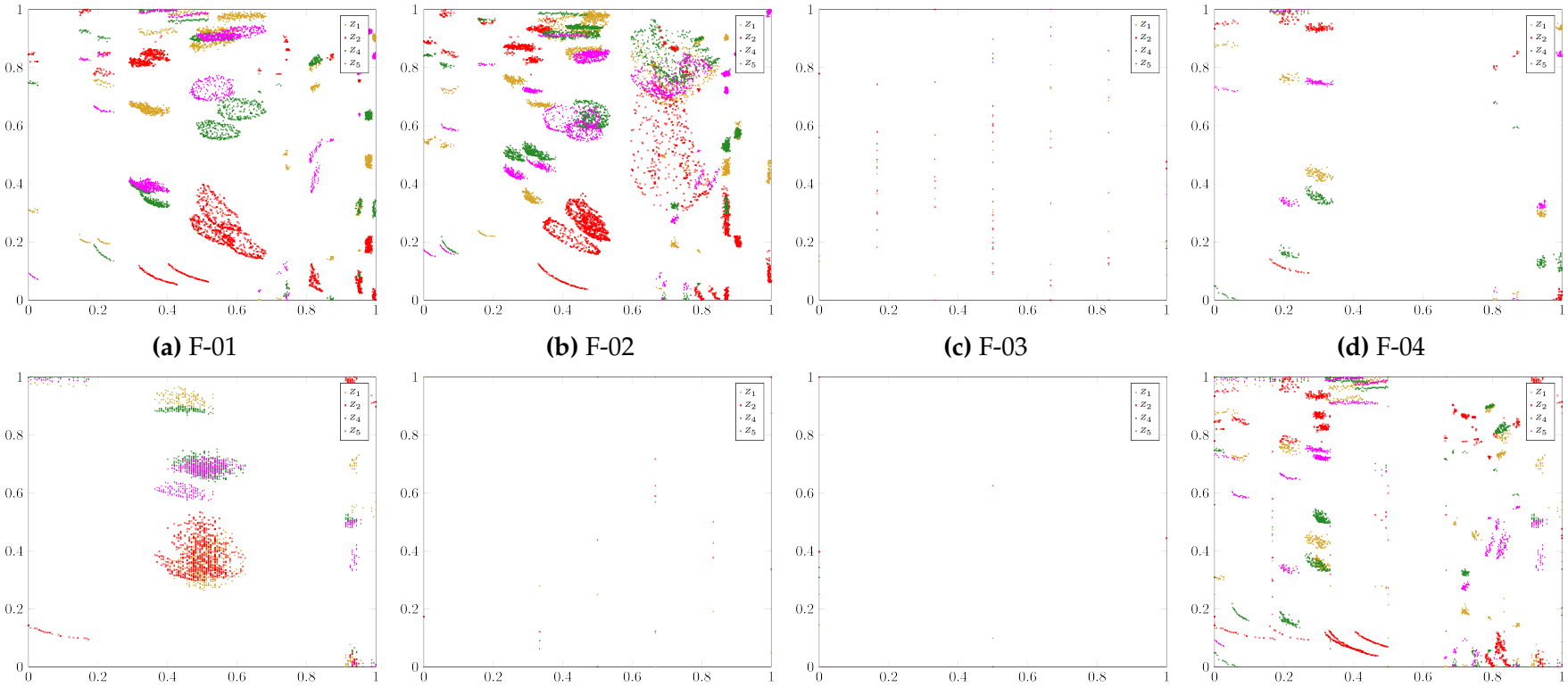
Figure C.16: Scatter plots of each of the WSRP-D instances, showing the relationship between objective Z_3 and each of the other four objectives. In each case, the x axis shows the value of objective Z_3 and the y axis shows the values of each of the other objectives in different colours.



(a) E-01 **(b)** E-02 **(c)** E-03 **(d)** E-04

(e) E-05 **(f)** E-06 **(g)** E-07 **(h)** Combined E

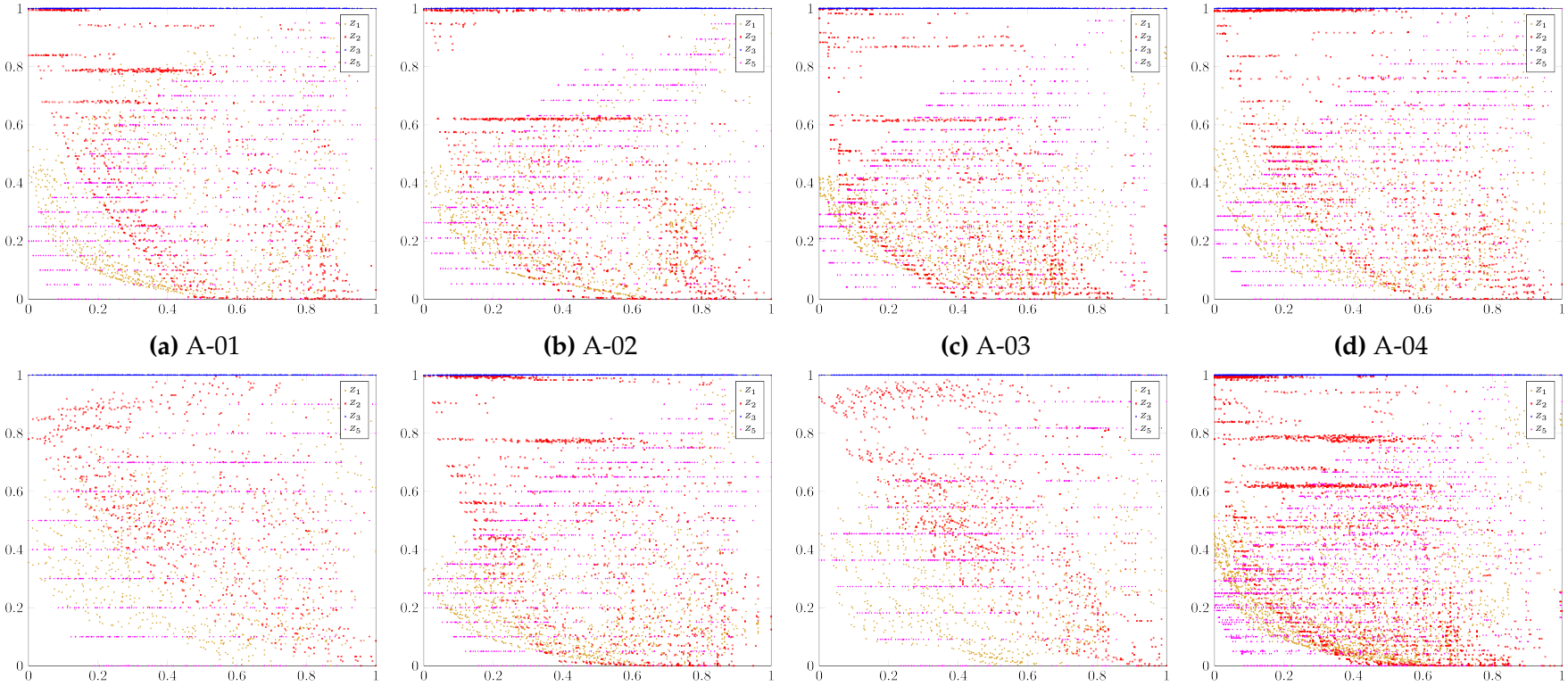
Figure C.17: Scatter plots of each of the WSRP-E instances, showing the relationship between objective Z_3 and each of the other four objectives. In each case, the x axis shows the value of objective Z_3 and the y axis shows the values of each of the other objectives in different colours.



(a) F-01 **(b)** F-02 **(c)** F-03 **(d)** F-04

(e) F-05 **(f)** F-06 **(g)** F-07 **(h)** Combined F

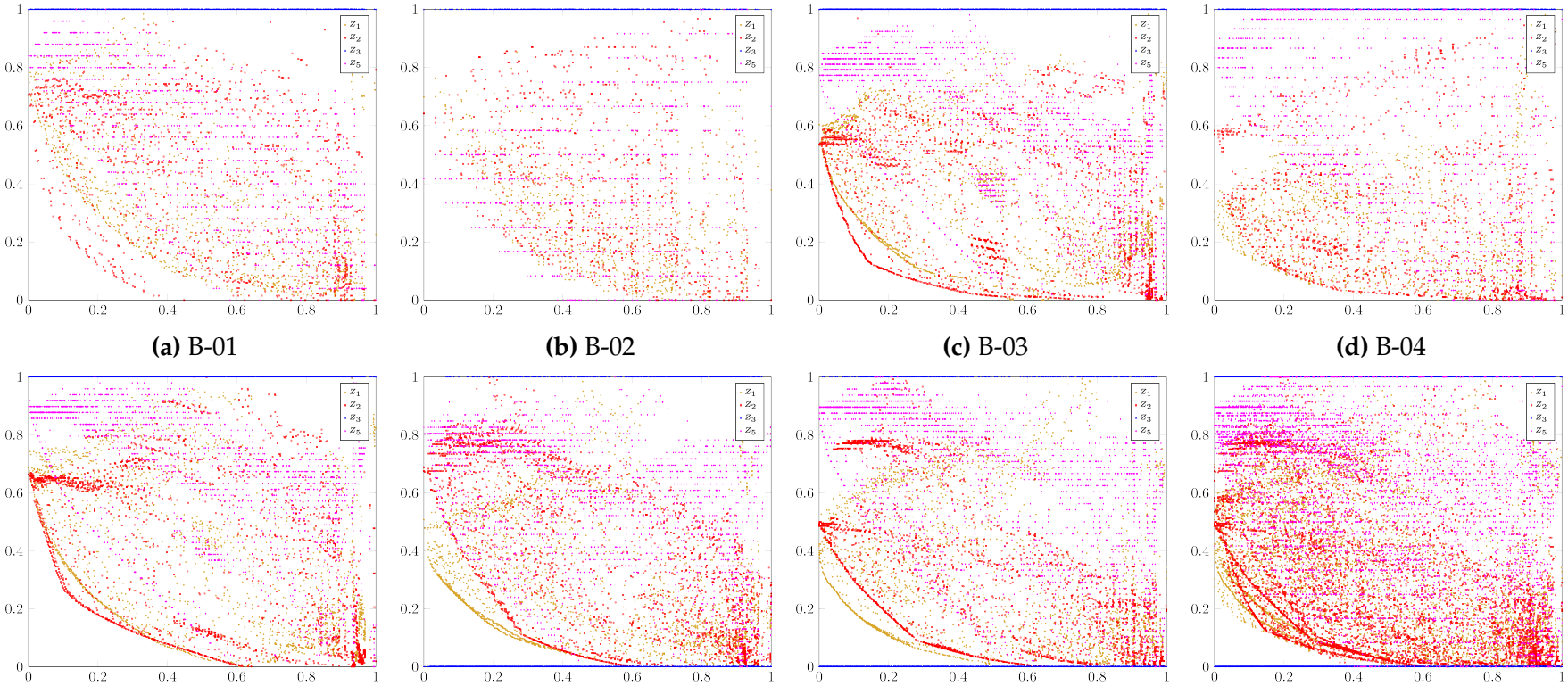
Figure C.18: Scatter plots of each of the WSRP-F instances, showing the relationship between objective Z_3 and each of the other four objectives. In each case, the x axis shows the value of objective Z_3 and the y axis shows the values of each of the other objectives in different colours.



(a) A-01 **(b)** A-02 **(c)** A-03 **(d)** A-04

(e) A-05 **(f)** A-06 **(g)** A-07 **(h)** Combined A

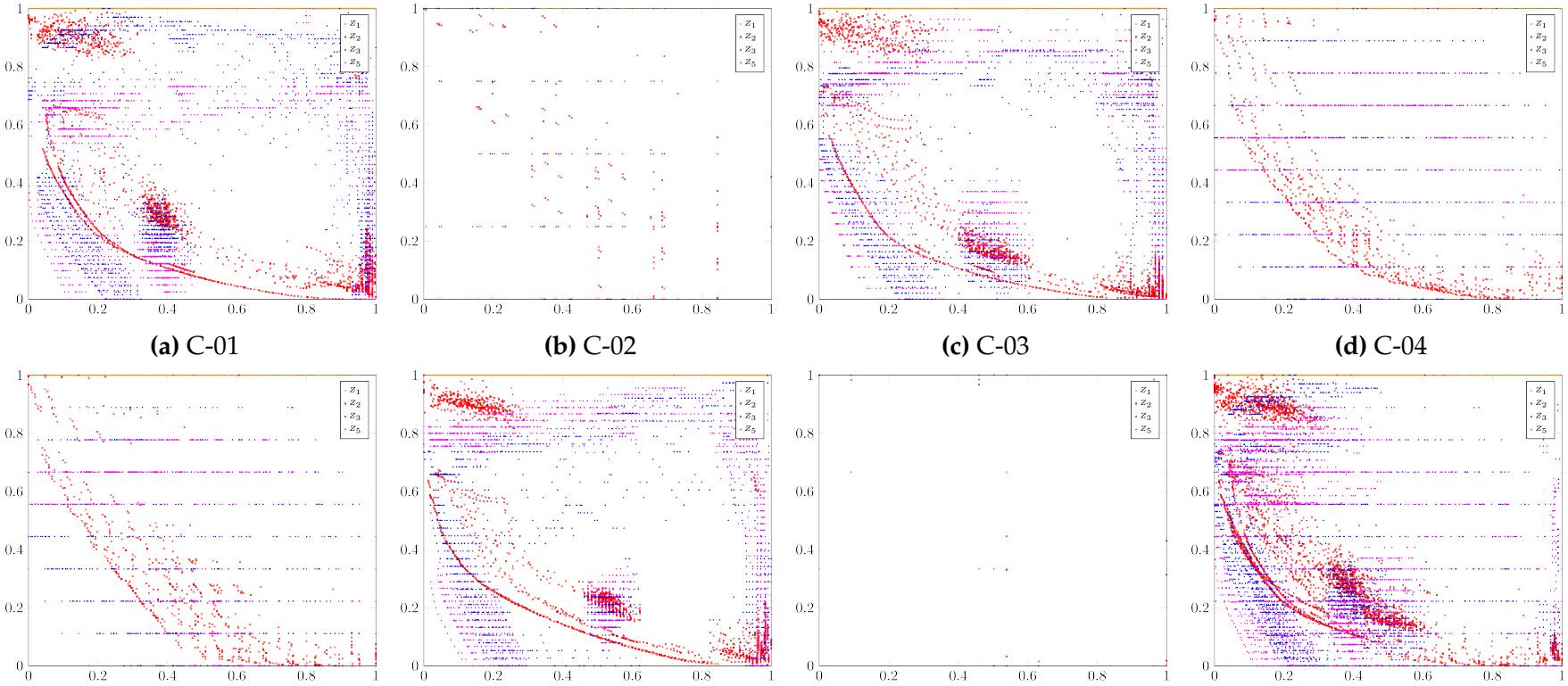
Figure C.19: Scatter plots of each of the WSRP-A instances, showing the relationship between objective Z_4 and each of the other four objectives. In each case, the x axis shows the value of objective Z_4 and the y axis shows the values of each of the other objectives in different colours.



(a) B-01 **(b)** B-02 **(c)** B-03 **(d)** B-04

(e) B-05 **(f)** B-06 **(g)** B-07 **(h)** Combined B

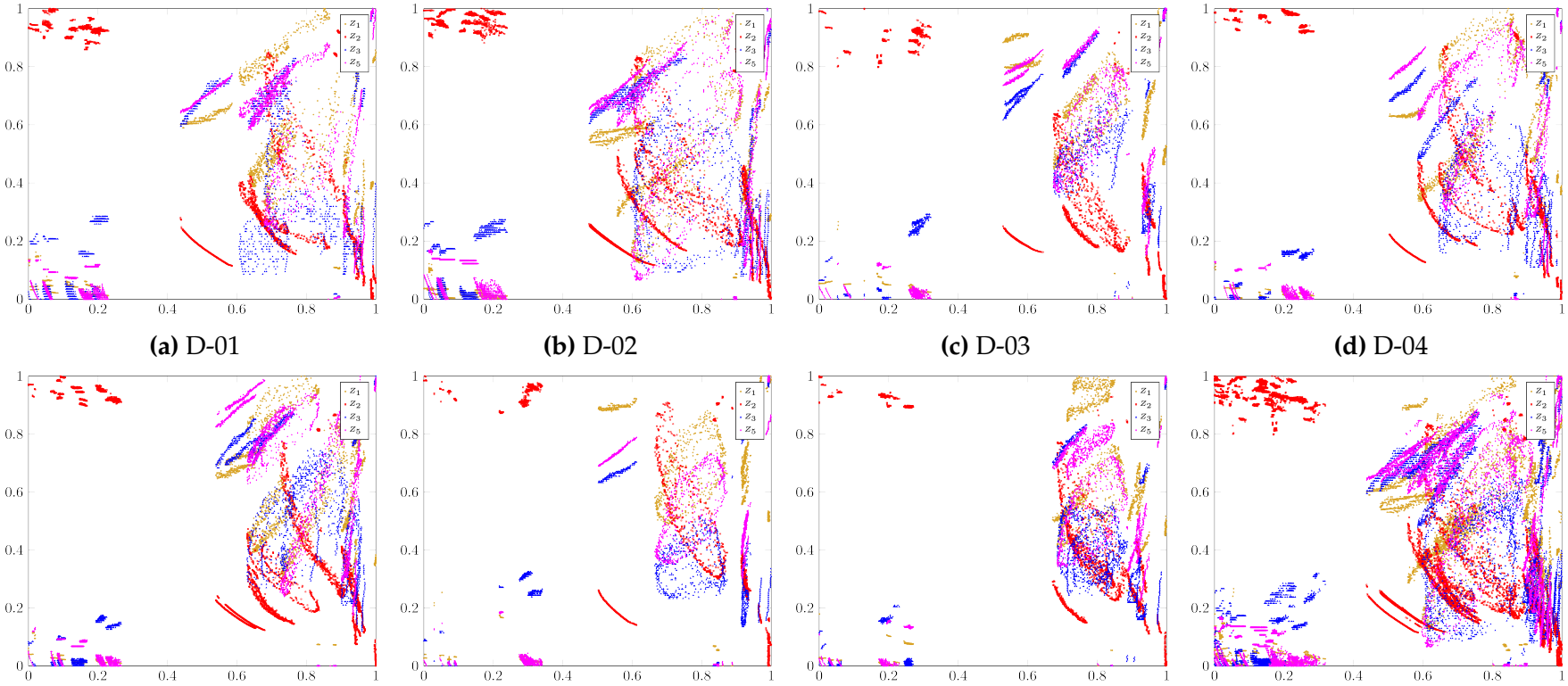
Figure C.20: Scatter plots of each of the WSRP-B instances, showing the relationship between objective Z_4 and each of the other four objectives. In each case, the x axis shows the value of objective Z_4 and the y axis shows the values of each of the other objectives in different colours.



(a) C-01 **(b)** C-02 **(c)** C-03 **(d)** C-04

(e) C-05 **(f)** C-06 **(g)** C-07 **(h)** Combined C

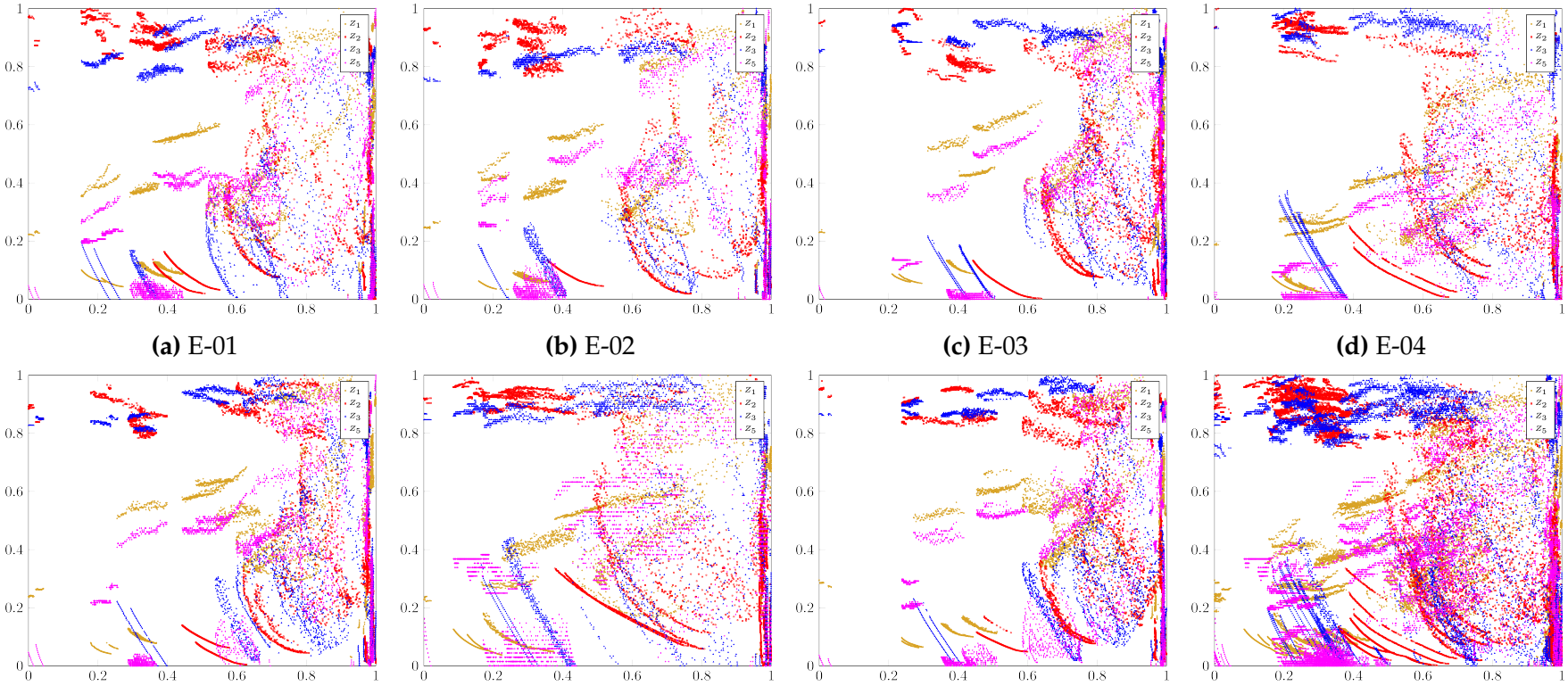
Figure C.21: Scatter plots of each of the WSRP-C instances, showing the relationship between objective Z_4 and each of the other four objectives. In each case, the x axis shows the value of objective Z_4 and the y axis shows the values of each of the other objectives in different colours.



(a) D-01 **(b)** D-02 **(c)** D-03 **(d)** D-04

(e) D-05 **(f)** D-06 **(g)** D-07 **(h)** Combined D

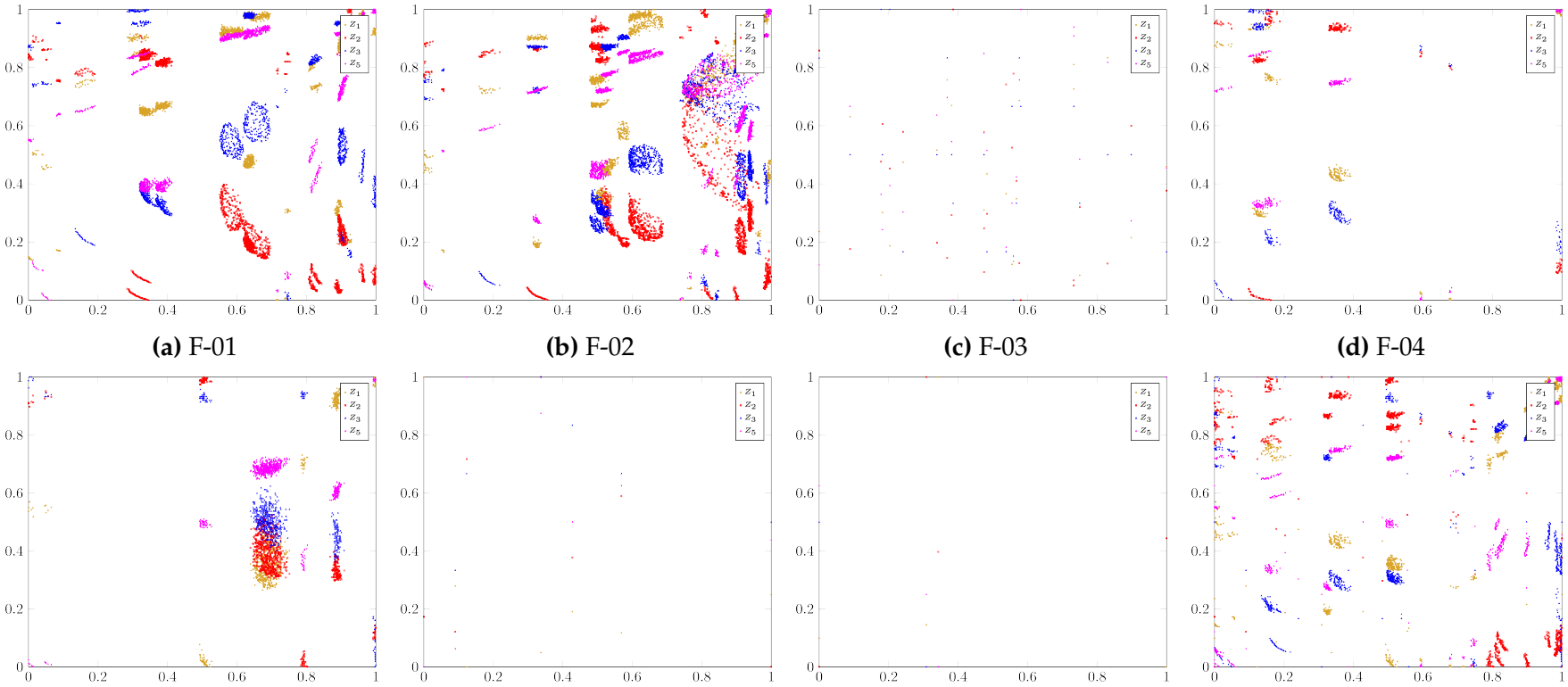
Figure C.22: Scatter plots of each of the WSRP-D instances, showing the relationship between objective Z_4 and each of the other four objectives. In each case, the x axis shows the value of objective Z_4 and the y axis shows the values of each of the other objectives in different colours.



(a) E-01 **(b)** E-02 **(c)** E-03 **(d)** E-04

(e) E-05 **(f)** E-06 **(g)** E-07 **(h)** Combined E

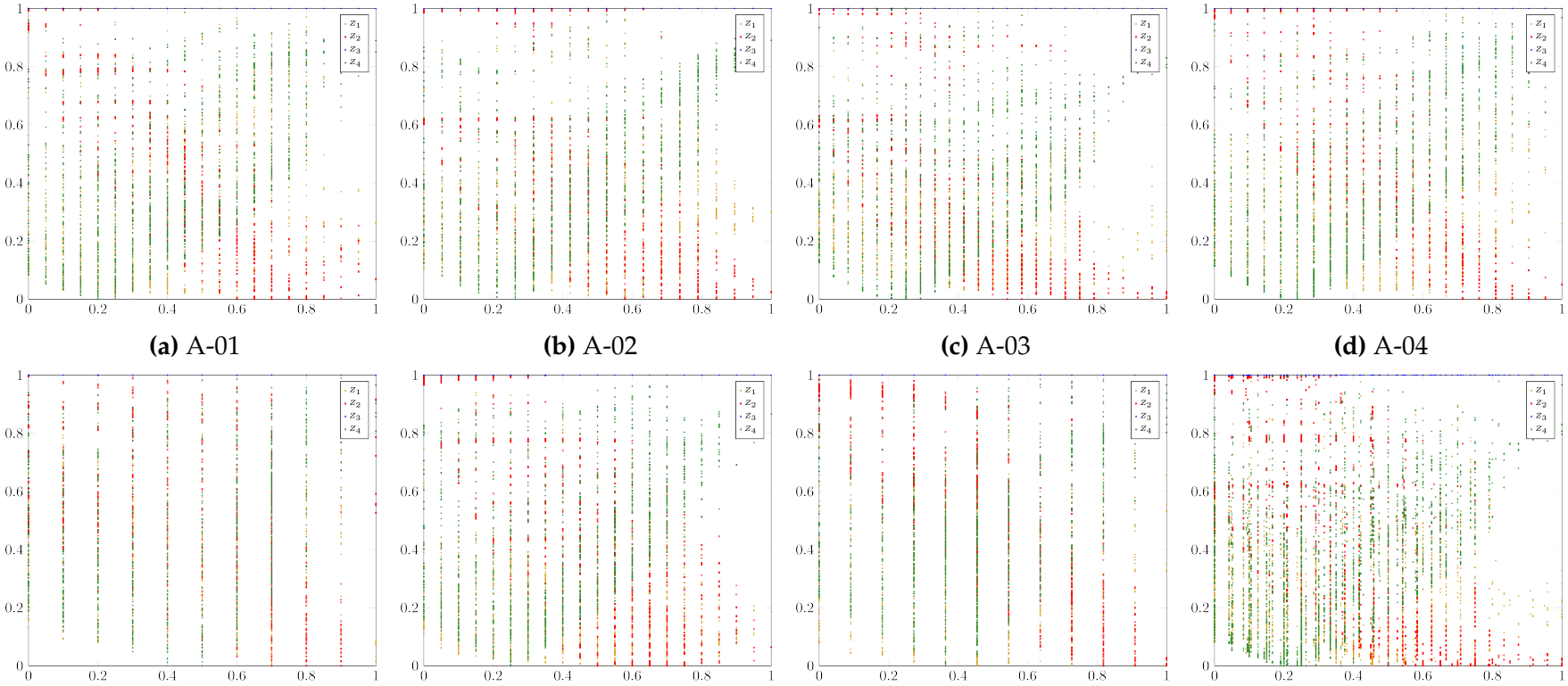
Figure C.23: Scatter plots of each of the WSRP-E instances, showing the relationship between objective Z_4 and each of the other four objectives. In each case, the x axis shows the value of objective Z_4 and the y axis shows the values of each of the other objectives in different colours.



(a) F-01 **(b)** F-02 **(c)** F-03 **(d)** F-04

(e) F-05 **(f)** F-06 **(g)** F-07 **(h)** Combined F

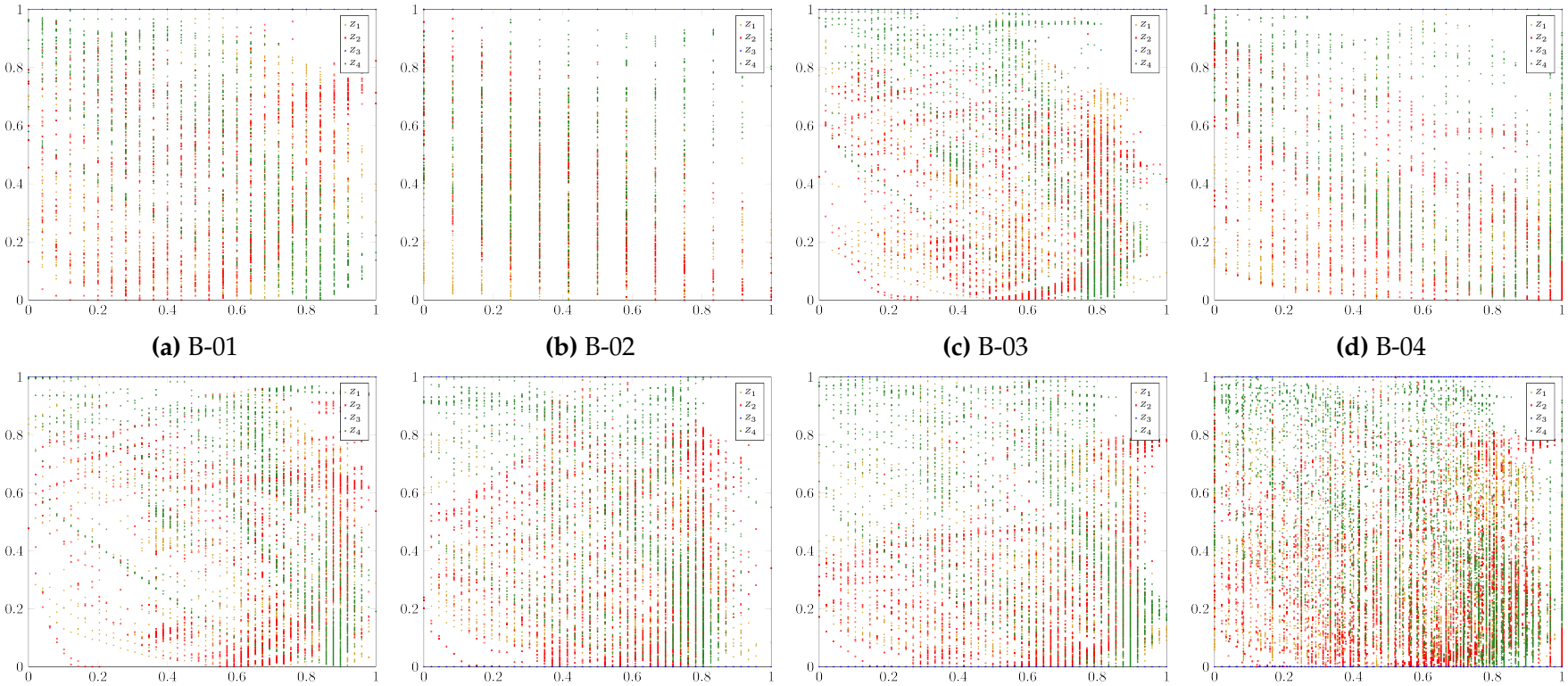
Figure C.24: Scatter plots of each of the WSRP-F instances, showing the relationship between objective Z_4 and each of the other four objectives. In each case, the x axis shows the value of objective Z_4 and the y axis shows the values of each of the other objectives in different colours.



(a) A-01 **(b)** A-02 **(c)** A-03 **(d)** A-04

(e) A-05 **(f)** A-06 **(g)** A-07 **(h)** Combined A

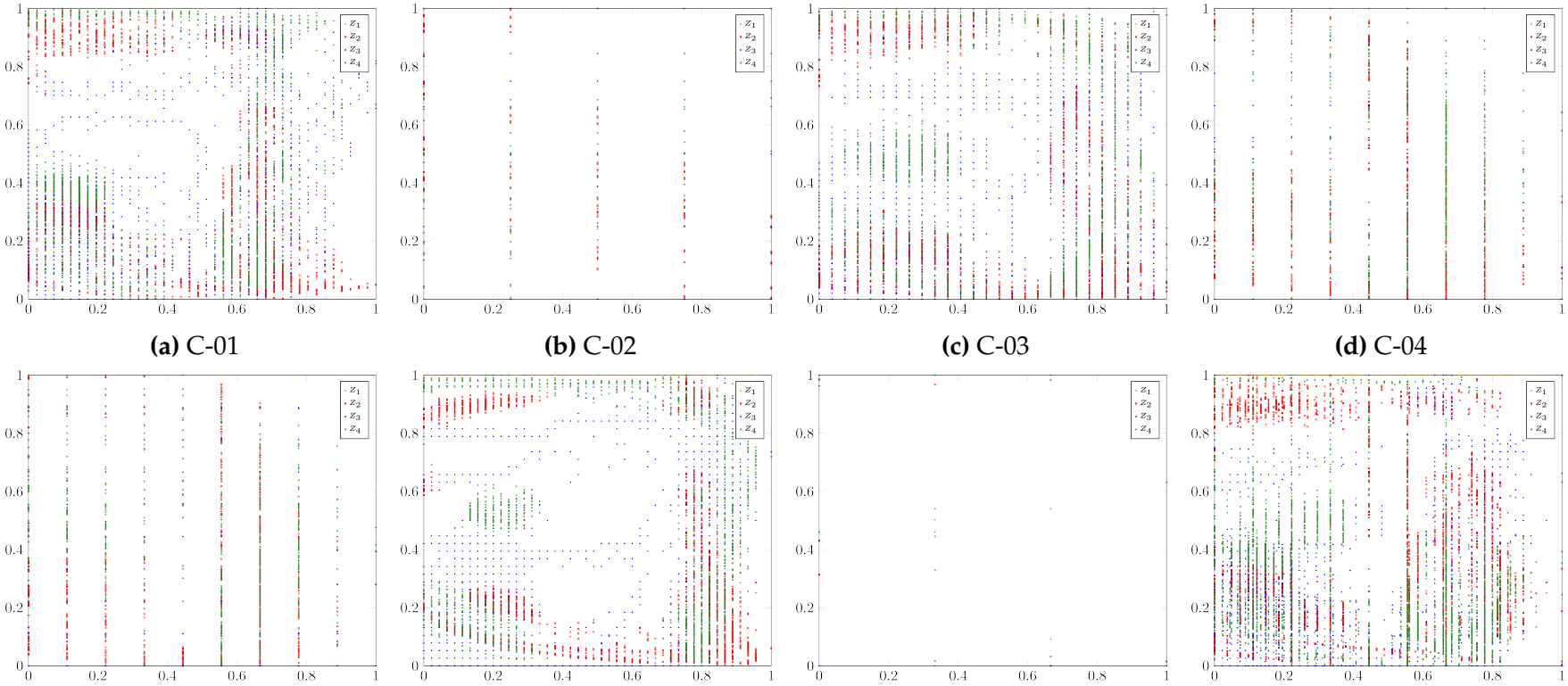
Figure C.25: Scatter plots of each of the WSRP-A instances, showing the relationship between objective Z_5 and each of the other four objectives. In each case, the x axis shows the value of objective Z_5 and the y axis shows the values of each of the other objectives in different colours.



(a) B-01 **(b)** B-02 **(c)** B-03 **(d)** B-04

(e) B-05 **(f)** B-06 **(g)** B-07 **(h)** Combined B

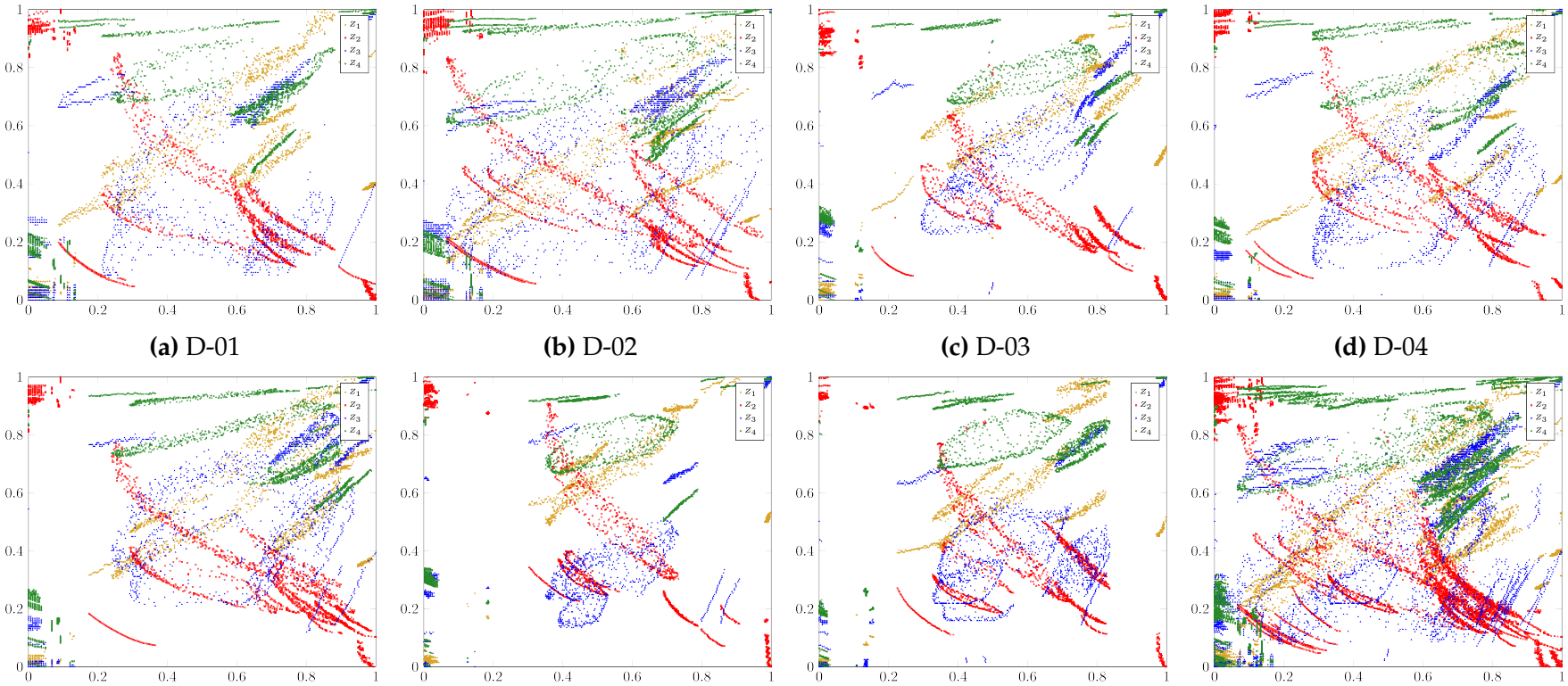
Figure C.26: Scatter plots of each of the WSRP-B instances, showing the relationship between objective Z_5 and each of the other four objectives. In each case, the x axis shows the value of objective Z_5 and the y axis shows the values of each of the other objectives in different colours.



(a) C-01 **(b)** C-02 **(c)** C-03 **(d)** C-04

(e) C-05 **(f)** C-06 **(g)** C-07 **(h)** Combined C

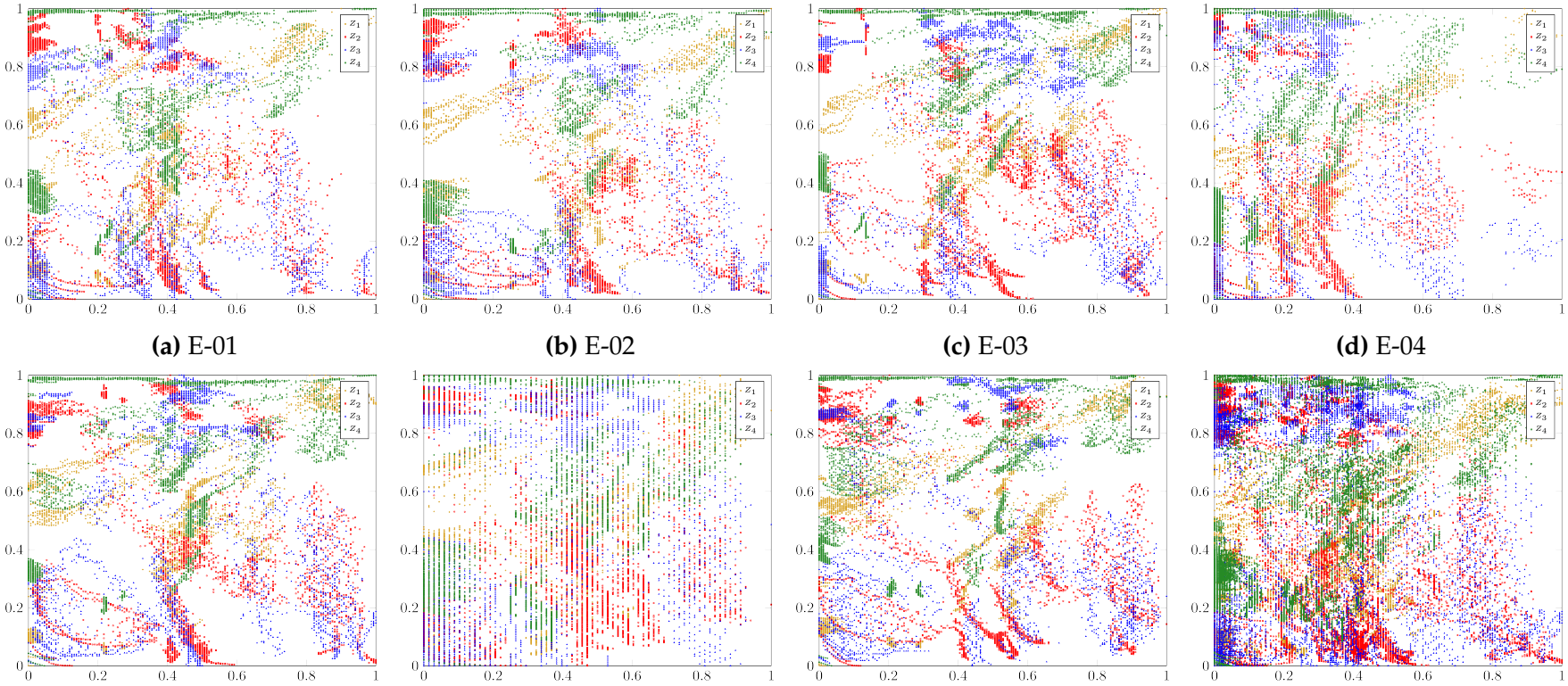
Figure C.27: Scatter plots of each of the WSRP-A instances, showing the relationship between objective Z_5 and each of the other four objectives. In each case, the x axis shows the value of objective Z_5 and the y axis shows the values of each of the other objectives in different colours.



(a) D-01 **(b)** D-02 **(c)** D-03 **(d)** D-04

(e) D-05 **(f)** D-06 **(g)** D-07 **(h)** Combined D

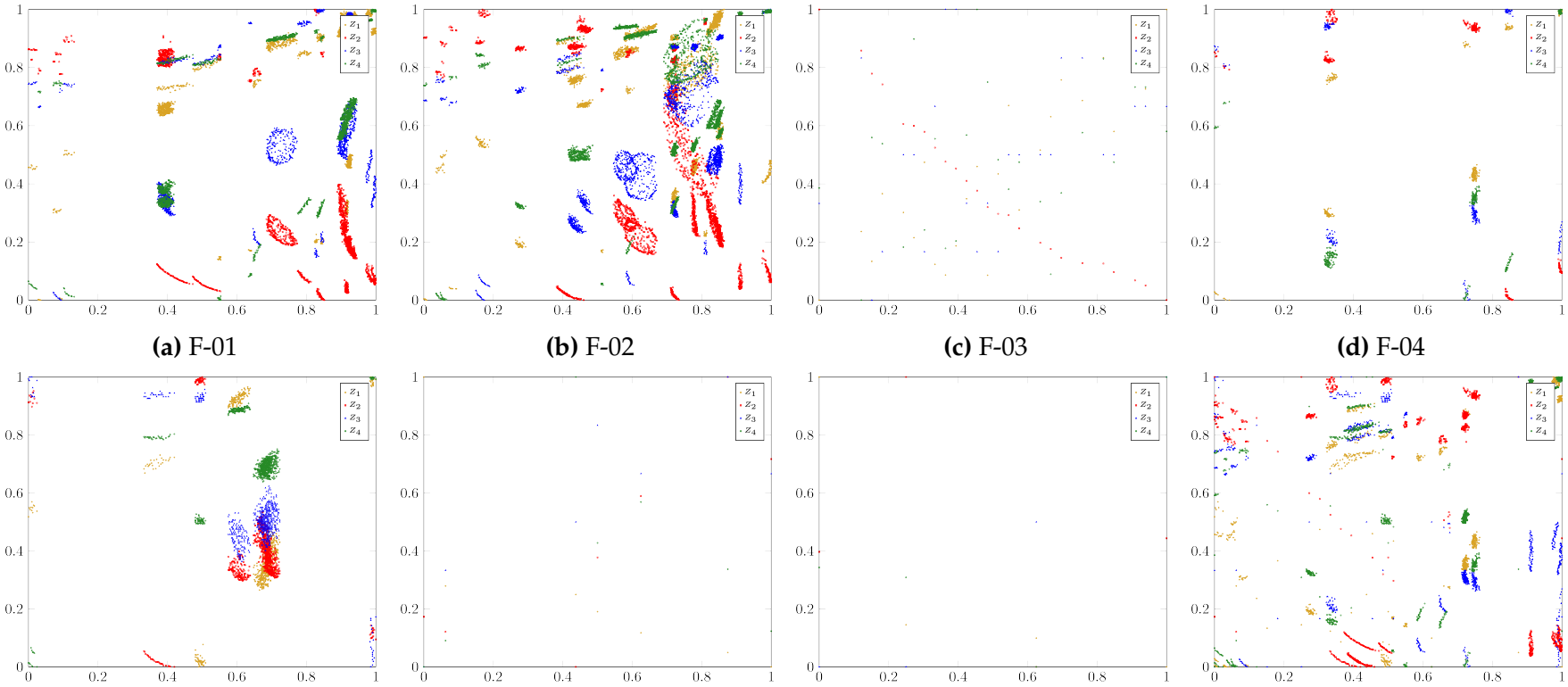
Figure C.28: Scatter plots of each of the WSRP-A instances, showing the relationship between objective Z_5 and each of the other four objectives. In each case, the x axis shows the value of objective Z_5 and the y axis shows the values of each of the other objectives in different colours.



(a) E-01 **(b)** E-02 **(c)** E-03 **(d)** E-04

(e) E-05 **(f)** E-06 **(g)** E-07 **(h)** Combined E

Figure C.29: Scatter plots of each of the WSRP-E instances, showing the relationship between objective Z_5 and each of the other four objectives. In each case, the x axis shows the value of objective Z_5 and the y axis shows the values of each of the other objectives in different colours.



(a) F-01 **(b)** F-02 **(c)** F-03 **(d)** F-04

(e) F-05 **(f)** F-06 **(g)** F-07 **(h)** Combined F

Figure C.30: Scatter plots of each of the WSRP-F instances, showing the relationship between objective Z_5 and each of the other four objectives. In each case, the x axis shows the value of objective Z_5 and the y axis shows the values of each of the other objectives in different colours.