Maria Teresa Galvão Dias

A New Approach to the Bus Driver Scheduling Problem using Multiobjective Genetic Algorithms
A New Approach to the
Bus Driver Scheduling Problem using
Multiobjective Genetic Algorithms

Dissertação apresentada à Faculdade de Engenharia da Universidade do Porto para a
obtenção do grau de Doutor em Ciências de Engenharia

Orientador: Professor Doutor João Bernardo de Sena Esteves Falcão e Cunha
Co-orientador: Professor Doutor Jorge Manuel Pinho de Sousa

Faculdade de Engenharia da Universidade do Porto
2005
Resumo

Nesta tese é apresentada uma abordagem multiobjectivo baseada em algoritmos genéticos para o problema de geração de serviços de motoristas em empresas de transporte coletivo de passageiros. A elaboração dos serviços dos motoristas é uma etapa de extrema importância no planeamento operacional de transportes coletivos, que consiste na definição dos serviços diários dos motoristas por forma a garantir a realização das viagens planeadas para um conjunto de viaturas. Este é um processo complexo, que envolve diversos objectivos, frequentemente conflituosos, tais como custos, qualidade do serviço prestado e satisfação das expectativas dos motoristas. Efectuou-se uma análise detalhada do processo de decisão dos planeadores para a avaliação da qualidade de um horário de serviços. Como resultado desta análise, os principais objectivos e restrições envolvidos neste processo foram identificados.

Por forma a considerar este objectivos simultâneos e conflituosos, propôs-se uma abordagem multiobjectivo baseada em algoritmos genéticos. Dois algoritmos genéticos são considerados, baseados em dois modelos multiobjectivo: o modelo Agregado e o modelo Não Dominado. No modelo Agregado, os objectivos são agrupados numa única função objectivo de acordo com uma função de ponderação apropriada. O modelo Não Dominado utiliza o conceito de solução de Pareto para a ordenação das soluções, tendo sido desenvolvido um novo procedimento de atribuição de fitness. Ambos os algoritmos utilizam o mesmo conjunto de operadores especializados de cruzamento e mutação.

Os algoritmos genéticos encontram-se integrados numa aplicação, GenT, na qual foi dada particular atenção à interacção com o planeador. Esta aplicação, amigável e flexível, permite a um utilizador com experiência e conhecimento do problema produzir e analisar as soluções mais adequadas ao seu problema. A aplicação GenT é inteiramente compatível com o sistema GIST, um sistema de apoio à decisão actualmente em operação em várias empresas de transporte coletivo.

A nova abordagem foi testada num conjunto de problemas reais de três empresas portuguesas e as soluções obtidas foram comparadas com as soluções actualmente implementadas nessas empresas. Os dois algoritmos genéticos produziram várias soluções alternativas competitivas num período de tempo consideravelmente curto.

A abordagem multiobjectivo proposta nesta tese mostrou-se extremamente útil no apoio à decisão
em operações quotidians de planeamento de serviços, assim como na simulação de cenários alternativos em horizontes de médio prazo.
Abstract

This thesis proposes a new approach for the bus driver scheduling problem (BDSP) using multiobjective genetic algorithms (GAs). Bus driver scheduling is a critical stage of the operational planning process in mass transit companies. It consists in constructing a set of legal duties that together cover all the trips planned for a group of vehicles. This is a complex process guided by several, often conflicting objectives, involving costs, quality of service and the satisfaction of the drivers' expectations.

A comprehensive analysis of the decision process and of the drivers' schedule evaluation process was performed. As a result of this analysis, the main objectives and constraints involved in this process have been identified.

In order to tackle those conflicting objectives, a multiobjective approach based on GAs is provided. Two new GAs are proposed, based on two different multiobjective models: the Aggregate (Agg) model and the Non-Dominated (ND) model. In the Agg model all the objectives are merged into a single objective function according to an appropriate weighting function. The ND model uses the concept of Pareto dominance to rank the solutions and a new fitness assignment procedure has been designed. Both models use the same set of knowledge based crossover and mutation operators.

The GAs are integrated in a new software application, GenT, in which particular attention has been given to the interaction with the planner. GenT is a friendly, interactive and flexible software tool that enables planners with knowledge and experience to produce and select the most appropriate solution to their problems. GenT is fully compatible with GIST, a decision support system for the operational planning process currently in use by several companies.

The approach has been tested on a collection of real problem instances from three Portuguese companies and the solutions obtained have been compared with the solutions currently implemented in those companies. Both GAs have been able to consistently provide a set of competitive alternative solutions in a rather short period of time.

The new multiobjective approach proposed in this thesis has proved to be valuable and powerful in supporting decision making for short-term (daily) operations, as well as in the simulation of alternative operating scenarios in a medium-term horizon.
Résumé

Cette thèse propose une nouvelle approche pour le problème de la génération d'horaire des chauffeurs d'autobus, en employant des algorithmes génétiques multiobjectif. La génération des horaires des chauffeurs est une étape critique du processus de planification des opérations aux compagnies de transport en commun. Elle consiste à la construction des services quotidiens des chauffeurs couvrant les voyages faits par un certain ensemble de véhicules.

Établir un horaire de chauffeurs est un processus complexe guidé par plusieurs objectifs, souvent conflitueux, qui comprennent les coûts d'opération, la qualité des services et la satisfaction des expectatives des chauffeurs.

Une analyse exhaustive du processus décisionnel et de l'évaluation des horaires de chauffeurs a été effectuée. De cette analyse, on a pu identifier les principaux objectifs et contraintes.

Avec le but de considérer ces objectifs contradictoires, une approche multiobjectif basée sur des algorithmes génétiques a été développée. On propose deux nouveaux algorithmes génétiques, basés sur deux modèles multiobjectifs différents: le modèle Agrégée (Agg) et le modèle Non-Dominé (ND). Dans le modèle Agg tous les objectifs sont groupés dans une seule fonction objectif selon une fonction de pondération appropriée. Le modèle ND emploie le concept de la dominance de Pareto pour ranger les solutions et un nouveau procédé de “fitness assignment” a été conçu. Les deux algorithmes emploient le même ensemble d'opérateurs de croisement et de mutation.

Les algorithmes génétiques sont intégrés dans une nouvelle application de logiciel, GenT, dans laquelle une attention particulière a été donnée à l'interaction avec le planificateur. GenT est un outil amical, interactif et flexible qui permet à des planificateurs expérimentés produire et choisir la solution la plus appropriée à leurs problèmes. GenT est entièrement compatible avec GIST, un système d'aide à la décision pour le processus de planification opérationnel actuellement en service par plusieurs compagnies.

L'approche a été examinée sur un ensemble d'exemples réels de problèmes de trois compagnies portugaises et les solutions obtenues ont été comparées aux solutions actuellement mises en application à ces compagnies. Les deux algorithmes génétiques ont pu fournir un ensemble de solutions concurrentielles dans une période assez courte.

vii
La nouvelle approche multiobjective proposée dans cette thèse s’est avérée très utile, dans l’aide à la décision pour des opérations (quotidiennes) à court terme aussi bien que dans la simulation des scénarios alternatifs de changes de fonctionnement dans l’horizon à moyen terme.
Agradecimentos

Gostaria de agradecer em primeiro lugar aos meus orientadores, o Professor João Falcão e Cunha e o Professor Jorge Pinho de Sousa. A ambos agradeço o apoio incondicional e a confiança que demonstraram ao longo deste trabalho. As conversas que tivemos representaram sem dúvida contribuições valiosas, mas permanecerão para sempre como exemplos de seriedade e competência.

Gostaria também de agradecer ao GEIN, pelo óptimo ambiente de trabalho e pelas condições que me proporcionou para a realização desta tese. De novo ao João Falcão e Cunha, agradeço a confiança que sempre depositou em mim ao longo dos anos, a amizade e o bom senso. Uma palavra especial para o Zé Luís, um grande amigo que por acaso partilha comigo o gabinete, para o Jorge Freire, um amigo para sempre, para o José António Cabral, um cúmplice, e para o Professor Rui Guimarães por permanecer para mim uma referência fundamental. Finalmente, à Soledade, à Isabel e à Carmo, agradeço a competência e a amabilidade com que tomam conta de nós.

Um agradecimento muito especial ao Zé Vasconcelos ao lado de quem, pela sensibilidade e inteligência é um privilégio trabalhar e conviver.

Não posso esquecer todos os elementos da OPT, pela disponibilidade que sempre manifestaram, em particular ao Luís Filipe, pelo excelente trabalho desenvolvido na Bancada de Serviços, uma ferramenta fundamental de apoio ao desenvolvimento desta tese.

Uma palavra de agradecimento às empresas de transporte colectivo que utilizam o GIST, pela disponibilidade sempre demonstrada. Em particular, gostaria de agradecer à Dra. Maria José Matos, da Carris, pelo profissionalismo e competência com que acompanharam este trabalho.

Não posso deixar de agradecer à Câmara Municipal de Vila do Conde, em particular ao Café Concerto do Centro da Juventude e ao Bar das Piscinas Municipais por me terem acolhido tão amavelmente durante o período de escrita da tese.

Agradeço ainda a todos os meus amigos pelo apoio incondicional e também pela paciência com que acompanharam a realização deste trabalho.

Aos meus pais e irmão, agradeço simplesmente por serem como são. Ao João Francisco, por ser uma criança maravilhosa e pelo espelho nos seus olhos. A eles dedico esta tese.
Contents

1 Introduction ................................................................. 1
  1.1 Context of the research .............................................. 1
  1.1.1 The operational planning process .............................. 1
  1.1.2 Decision support systems ..................................... 3
  1.1.3 Portuguese mass transit companies ......................... 3
  1.2 Scope and relevance of the thesis ............................... 5
  1.2.1 The bus driver scheduling problem ......................... 5
  1.2.2 Multiobjective genetic algorithms ......................... 6
  1.3 Objectives of the thesis .......................................... 7
  1.4 Structure of the thesis ............................................ 9

2 The Bus Driver Scheduling Problem (BDSP) ......................... 11
  2.1 Introduction ......................................................... 11
  2.2 The OPP in Portuguese mass transit companies ............... 12
    2.2.1 The operational planning phases ............................. 12
      2.2.1.1 The planning phase ..................................... 12
      2.2.1.2 The rostering phase ..................................... 15
    2.2.2 Operational planning at Portuguese mass transit companies 15
  2.3 Bus driver scheduling ............................................. 20
    2.3.1 Input data from vehicle scheduling ....................... 20
    2.3.2 Rules for driver duties ..................................... 22
    2.3.3 Overall rules .................................................. 28
    2.3.4 The driver scheduling process ............................... 29
      2.3.4.1 Manual scheduling ....................................... 29
      2.3.4.2 Computer scheduling ..................................... 31
2.4 Decision support systems for the operational planning process .......... 32
  2.4.1 General functionality and features ........................................... 32
  2.4.2 The GIST System ................................................................. 33
      2.4.2.1 System structure .......................................................... 34
      2.4.2.2 The driver scheduling module in the GIST System ................. 35
      2.4.2.3 The Duties Framework ................................................... 37
  2.4.3 Other DSS systems for the operational planning process ............... 39
      2.4.3.1 The HASTUS System ...................................................... 39
      2.4.3.2 The HOT II System ...................................................... 40
      2.4.3.3 The BUSMAN II System ................................................ 41
      2.4.3.4 The MTRAM System ...................................................... 41
      2.4.3.5 The MICROBUS System ................................................ 41
      2.4.3.6 The Crews System ...................................................... 42
      2.4.3.7 The TRAPEZE System .................................................. 42
      2.4.3.8 The GoalBus and GoalDriver Systems .................................. 42
  2.5 Summary .................................................................................... 42

3 Approaches to the BDSP 45
  3.1 Introduction ................................................................................ 45
  3.2 Heuristic methods ...................................................................... 46
  3.3 Mathematical programming methods .......................................... 48
      3.3.1 Methods based on set covering and set partitioning formulations 49
      3.3.2 Methods based on network models ....................................... 57
      3.3.3 Integrated approaches ....................................................... 59
          3.3.3.1 Vehicle and driver scheduling .................................... 60
          3.3.3.2 The Crew Planning Problem ....................................... 61
  3.4 Metaheuristics ........................................................................... 63
      3.4.1 Genetic Algorithms ............................................................ 64
          3.4.1.1 Genetic algorithms for the set partitioning problem ........ 65
          3.4.1.2 Genetic algorithms for the set covering problem ............ 66
          3.4.1.3 Genetic Algorithms for the BDSP ................................. 67
          3.4.1.4 Multiobjective genetic algorithms ............................... 68
      3.4.2 Evolutionary Strategies ..................................................... 69
          3.4.2.1 Basic principles .......................................................... 69
      3.4.3 Tabu Search ....................................................................... 71
3.4.3.1 Basic principles ........................................ 71
3.4.3.2 Tabu Search approaches to the BDSP .................. 71
3.5 Summary ....................................................... 73

4 Genetic algorithms for multiobjective optimisation .......... 75
4.1 Introduction ................................................ 75
4.2 Basic concepts of multiobjective optimisation .............. 76
4.3 Some general approaches for multiobjective optimisation .. 77
4.4 Basic concepts of Genetic Algorithms ....................... 80
4.5 Multiobjective Genetic Algorithms ........................ 83
   4.5.1 Weighting based approaches .......................... 84
   4.5.2 Pareto based approaches .............................. 85
4.6 Summary ....................................................... 86

5 A Multiobjective Approach to the BDSP ..................... 89
5.1 Introduction ................................................ 89
5.2 An integrated analysis of the BDSP ........................ 91
   5.2.1 Redefining some fundamental concepts ................ 94
   5.2.2 The complexity of a problem .......................... 95
      5.2.2.1 Groups of pieces-of-work ........................ 95
      5.2.2.2 Duty types ....................................... 96
      5.2.2.3 Operational rules ............................... 97
      5.2.2.4 Problem structure ................................ 97
   5.2.3 Evaluating solutions: a guide to the fundamental issues 98
5.3 Hard constraints ............................................ 101
5.4 Soft constraints involving leftovers ....................... 104
   5.4.1 Number and overall duration of leftovers ............. 105
   5.4.2 Controlling the duration of a leftover ............... 106
      5.4.2.1 Controlling the minimum and/or the maximum duration of the leftovers 107
      5.4.2.2 Controlling the average duration of the leftovers 107
5.5 Soft constraints on duties ................................ 109
   5.5.1 Constraints on the average duration of duties ........ 109
   5.5.2 Constraints on duty types ............................ 110
      5.5.2.1 Average duration for each duty type .............. 110
      5.5.2.2 Minimum and maximum percentage of each duty type 112
5.5.2.3 Assigning preferences to duty types .............................................. 113
5.5.3 Constraints on the number of vehicle changes ................................... 114
5.6 Objective functions .............................................................................. 115
  5.6.1 Evaluating the cost of a duty ......................................................... 115
  5.6.2 Evaluating the cost of a leftover .................................................... 116
  5.6.3 Minimizing the total cost of a solution ........................................... 117
  5.6.4 Minimizing the number of duties ................................................... 118
5.7 Summary ............................................................................................... 118

6 Multiobjective Genetic Algorithms for the BDSP ........................................ 121
  6.1 Introduction ......................................................................................... 121
  6.2 Coding schemes ................................................................................. 122
    6.2.1 Global data structures .............................................................. 124
    6.2.2 First level information: Duties Based coding scheme ................. 127
    6.2.3 Second level information: Pieces-of-Work based coding scheme ... 128
  6.3 Handling hard constraints ................................................................... 131
  6.4 Independent operators ........................................................................ 133
    6.4.1 Defining and selecting eligible duties .......................................... 134
      6.4.1.1 Building the set of eligible duties ....................................... 134
      6.4.1.2 Selecting eligible duties ..................................................... 136
    6.4.2 Generating the initial population ............................................... 137
    6.4.3 Crossover operators ....................................................................... 138
      6.4.3.1 The Union crossover operator ......................................... 139
      6.4.3.2 The Two-Point crossover operator ..................................... 141
      6.4.3.3 The Swap crossover operator ......................................... 144
    6.4.4 Destroying and repairing mutation operators ............................. 146
      6.4.4.1 The Destroy mutation operator ....................................... 147
      6.4.4.2 The Improve mutation operator ...................................... 149
      6.4.4.3 The Reduce _ Empty mutation operator ............................ 152
      6.4.4.4 The Reduce _ Bus _ Changes mutation operator ............... 153
    6.4.5 Population replacement strategies ............................................ 154
  6.5 The Aggregate model ......................................................................... 155
    6.5.1 Evaluation and fitness assignment ............................................. 156
      6.5.1.1 Linear ranking .................................................................. 157
      6.5.1.2 Lexical linear ranking ..................................................... 158
6.5.2 Parent selection .................................................. 159

6.6 The model based on non-dominated solutions ......................... 160

6.6.1 The external set ................................................. 161

6.6.1.1 Building the external set ................................... 164

6.6.2 Evaluation and fitness assignment ................................ 167

6.6.3 Parent selection .................................................. 172

6.7 Diversification strategies ........................................... 172

6.7.1 Boltzmann selection mechanisms .................................. 173

6.7.2 Duplicates .......................................................... 174

6.7.3 Dynamic crossover and mutation rates ............................. 174

6.7.3.1 Dynamic rates based on the entropy of the population ......... 175

6.7.3.2 Crossover dynamic rates based on performance ............... 178

6.7.4 Population reinitialization ....................................... 180

6.7.5 Inserting random chromosomes ................................... 182

6.8 Summary ................................................................... 182

7 Solving Real Problems ................................................ 185

7.1 Introduction ............................................................ 185

7.2 Computational experiments with standard problems ................. 186

7.3 Using the GenT application .......................................... 188

7.4 The methodology used in the experiments ............................. 191

7.4.1 Preprocessing the input data ...................................... 192

7.4.2 Adopted methodology .............................................. 195

7.4.2.1 Criteria ....................................................... 195

7.4.2.2 Parameters .................................................... 195

7.4.2.3 General guidelines ............................................. 196

7.5 Bus driver scheduling at Carris ...................................... 201

7.5.1 Computational results ............................................ 202

7.5.2 Concluding remarks ............................................... 205

7.6 Bus driver scheduling at STCP ....................................... 207

7.6.1 Computational results ............................................. 208

7.6.2 Concluding remarks ............................................... 212

7.7 Bus driver scheduling in SMTUC ..................................... 213

7.7.1 Computational results ............................................. 215

7.7.2 Concluding remarks ............................................... 219
<table>
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.8</td>
<td>Summary</td>
<td>219</td>
</tr>
<tr>
<td>8</td>
<td>Conclusions and Future Work</td>
<td>221</td>
</tr>
<tr>
<td>8.1</td>
<td>Main achievements of this thesis</td>
<td>222</td>
</tr>
<tr>
<td>8.2</td>
<td>Future research</td>
<td>223</td>
</tr>
<tr>
<td>A</td>
<td>Performance of crossover operators</td>
<td>225</td>
</tr>
<tr>
<td>A.1</td>
<td>Measuring the performance of the operators</td>
<td>226</td>
</tr>
<tr>
<td>A.2</td>
<td>Union crossover operator</td>
<td>227</td>
</tr>
<tr>
<td>A.2.1</td>
<td>Experimental design</td>
<td>227</td>
</tr>
<tr>
<td>A.2.2</td>
<td>Influence of problem density</td>
<td>228</td>
</tr>
<tr>
<td>A.2.3</td>
<td>Influence of the number of parents</td>
<td>229</td>
</tr>
<tr>
<td>A.2.4</td>
<td>Conclusions</td>
<td>230</td>
</tr>
<tr>
<td>A.3</td>
<td>Comparing the crossover operators</td>
<td>231</td>
</tr>
<tr>
<td>A.3.1</td>
<td>Experimental design</td>
<td>231</td>
</tr>
<tr>
<td>A.3.2</td>
<td>Influence of the density of the problems</td>
<td>232</td>
</tr>
<tr>
<td>A.3.3</td>
<td>Performance of the crossover operators</td>
<td>232</td>
</tr>
<tr>
<td>A.3.4</td>
<td>Conclusions</td>
<td>234</td>
</tr>
</tbody>
</table>
List of Figures

1-1 Stages of the Operational Planning Process ........................................ 2
1-2 Integrating GenT with GIST ................................................................. 8

2-1 The phases of the operational planning process ................................... 13
2-2 A summary of the main characteristics of nine major Portuguese mass transit companies ................................................................. 17
2-3 Representation diagram of two running boards ..................................... 21
2-4 Some drivers scheduling concepts ....................................................... 24
2-5 Window for visualising and editing relief points ................................... 35
2-6 Windows for visualising and manipulating duty schedules .................... 36
2-7 Window for editing the duty types parameters ...................................... 37
2-8 Window showing the validated properties of the duty .......................... 38

3-1 General procedure to solve set covering and set partitioning problems .... 53

4-1 Example of a convex set and of a non-convex set .................................. 77
4-2 Application of the weighting sum method to a non-convex trade-off surface 79
4-3 Basic structure of a GA ................................................................. 81

5-1 An example of a solution with overcovers .......................................... 93
5-2 Replacing overcovers with leftovers .................................................. 94
5-3 Example of a solution where a constraint imposing a maximum overcover time for each duty has been imposed ................................................. 102
5-4 Example of a solution where constraints that impose a maximum overcover time for each pair of duties were imposed ................................................. 103
5-5 Example of a solution with no additional soft constraints ..................... 115
5-6 Example of a solution where a soft constraint to minimise the number of vehicle changes was added ........................................................ 115
LIST OF FIGURES

6-1 Information Levels .................................................. 124
6-2 Data structure used to store the list of pieces-of-work ............. 125
6-3 Data structure used to store the set of generated duties .......... 126
6-4 Duties based coding scheme ....................................... 128
6-5 Pieces-of-work based coding scheme ................................ 129
6-6 Updating the set of eligible duties ................................ 136
6-7 A solution selected to be a parent for Union Crossover: parent1 .... 140
6-8 A solution selected to be a parent to Union Crossover: parent2 .... 140
6-9 Example of a child resulting from the application of Union crossover to parent1 and parent2 ........................................ 141
6-10 A parent chromosome for to Swap Crossover: parent1 .............. 145
6-11 A parent chromosome for to Swap Crossover: parent2 .............. 145
6-12 A solution resulting from the application of Swap crossover to par1 and par2 .................................................. 146
6-13 Part of a solution selected for application of Destroy mutation .. 148
6-14 Result of the application of Destroy mutation ...................... 148
6-15 Part of a solution selected for application of Improve mutation .. 150
6-16 Result of the application of Improve mutation ...................... 150
6-17 Part of a solution selected for application of Improve mutation with overcovers .............................................. 151
6-18 Result of the application of Improve mutation with overcovers ..... 151
6-19 Part of a solution selected for the application of Reduce_Empty mutation .......................................................... 152
6-20 Result of the application of Reduce_Empty mutation ............... 152
6-21 Part of a solution selected for the application of Reduce_Bus_Changes mutation ..................................................... 153
6-22 Result of the application of Reduce_Bus_Changes mutation ........ 154
6-23 General structure of Agg model ..................................... 156
6-24 Three criteria evaluation plot of a population ...................... 159
6-25 General structure of ND model ...................................... 162
6-26 Example of two different populations for a maximization problem with two objectives .............................................. 168
6-27 External set and population for a maximization problem with two objectives ....................................................... 171
6-28 3-D plot of the function for decreasing the crossover rates ........ 176
6-29 3-D plot of the function for increasing the mutation rates ........ 177
6-30 3-D plot for the function for decreasing the mutation rates ........ 178
6-31 3-D plot of the function for increasing the crossover rate ........ 178
6-32 Summary of the multiobjective GA components ..................... 183

7-1 Main Window of GenT ............................................... 189
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-2</td>
<td>Parameters of the algorithm</td>
<td>190</td>
</tr>
<tr>
<td>7-3</td>
<td>Criteria selection and weight assignment</td>
<td>191</td>
</tr>
<tr>
<td>7-4</td>
<td>Parameters of criteria</td>
<td>192</td>
</tr>
<tr>
<td>7-5</td>
<td>Duty types</td>
<td>193</td>
</tr>
<tr>
<td>7-6</td>
<td>GenT input data process</td>
<td>194</td>
</tr>
<tr>
<td>7-7</td>
<td>A typical run of the Agg model without user interaction</td>
<td>199</td>
</tr>
<tr>
<td>7-8</td>
<td>A typical run of the ND model without user interaction</td>
<td>200</td>
</tr>
<tr>
<td>7-9</td>
<td>Solutions obtained for problem Carris-B</td>
<td>205</td>
</tr>
<tr>
<td>7-10</td>
<td>Number of duty types in ND solutions for problem STCP-A</td>
<td>211</td>
</tr>
<tr>
<td>7-11</td>
<td>Number of duty types in ND solutions for problem STCP-B</td>
<td>212</td>
</tr>
<tr>
<td>7-12</td>
<td>Number of duty types in ND solutions for problem STCP-C</td>
<td>213</td>
</tr>
<tr>
<td>7-13</td>
<td>Distribution of duty types in ND solutions for problem SMTUC-A</td>
<td>216</td>
</tr>
<tr>
<td>7-14</td>
<td>Distribution of duty types in ND solutions for problem SMTUC-B</td>
<td>218</td>
</tr>
<tr>
<td>7-15</td>
<td>Distribution of duty types in ND solutions for problem SMTUC-C</td>
<td>219</td>
</tr>
<tr>
<td>A-1</td>
<td>Case summaries for the performance measures</td>
<td>229</td>
</tr>
<tr>
<td>A-2</td>
<td>Boxplots of the three performance measures for categories of problem densities</td>
<td>229</td>
</tr>
<tr>
<td>A-3</td>
<td>Boxplot and mean plot for the pmd_nparents variable</td>
<td>230</td>
</tr>
<tr>
<td>A-4</td>
<td>Test for the effect of the number of parents and density on the performance (significant under 5%)</td>
<td>230</td>
</tr>
<tr>
<td>A-5</td>
<td>Case summaries for the performance of the operators</td>
<td>232</td>
</tr>
<tr>
<td>A-6</td>
<td>Boxplots for the performance measures and categories of density</td>
<td>233</td>
</tr>
<tr>
<td>A-7</td>
<td>Boxplots for the performance measures and the crossover operators</td>
<td>233</td>
</tr>
<tr>
<td>A-8</td>
<td>Boxplots for each performance measure according to categories of density</td>
<td>234</td>
</tr>
<tr>
<td>A-9</td>
<td>Boxplot and mean plot for the pmd_oper variable</td>
<td>235</td>
</tr>
<tr>
<td>A-10</td>
<td>Test for the effect of the operators and density on performance (significant under 5%)</td>
<td>235</td>
</tr>
</tbody>
</table>
List of Tables

2.1 An example of a set of global rules ........................................... 23
2.2 Duty types rules .................................................................. 26

5.1 Classification of constraints ................................................. 91
5.2 Number/overall duration of leftovers example ....................... 106
5.3 Controlling the duration of leftovers ...................................... 108
5.4 Average duration of duties .................................................. 110
5.5 Average duration for each duty type ...................................... 112
5.6 Duration and percentages of each type of duty for the solutions obtained ......................................................... 112
5.7 Percentages for each type of duty ........................................... 113
5.8 Duration and percentages of each type of duty for the solutions obtained ......................................................... 113
5.9 Assigning preferences to duty types ...................................... 114
5.10 Number of duties for each duty type for the solutions obtained ......................................................... 114
5.11 Number of vehicle changes .................................................. 116

6.1 Example of a DC chromosome ................................................ 127
6.2 Getting the leftovers for a PWC chromosome ......................... 129
6.3 A possible representation of a solution with overcovers .......... 130
6.4 A simplified representation of a solution with overcovers using the PWC scheme ......................................................... 130
6.5 Evaluation of the elements in the population .......................... 158
6.6 Chromosomes and criteria values for 5 non-dominated solutions ......................................................... 166
6.7 Applying different distance functions in the clustering process ......................................................... 166
6.8 criterion priorities and values for 5 non-dominated solutions ......................................................... 167
6.9 Computing the fitness of the population and of the external set ......................................................... 171

7.1 Computational results for a set of SPP benchmarking problems ......................................................... 188
7.2 Available Criteria .............................................. 195
7.3 Default parameter values ..................................... 196
7.4 Default parameter options ................................. 197
7.5 Final solutions in the elite set ......................... 200
7.6 Duty types for Carris ....................................... 202
7.7 Carris problem set ............................................ 203
7.8 Solutions for problem Carris-A ......................... 204
7.9 Solutions for problem Carris-B ......................... 204
7.10 Solutions for problem Carris-C ...................... 205
7.11 Duty types for STCP ....................................... 208
7.12 STCP problem set ........................................... 209
7.13 Solutions for STACP-A problem .................... 210
7.14 Solutions for problem STCP-B ...................... 211
7.15 Solutions for problem STCP-C .................... 212
7.16 Duty types for SMTUC .................................... 214
7.17 SMTUC problem set ....................................... 215
7.18 Solutions for problem SMTUC-A ................ 216
7.19 Solutions for problem SMTUC-B ................ 217
7.20 Solutions for problem SMTUC-C ................ 218

A.1 Correlation between performance measures ............. 228
List of Acronyms

ACSP  Airline Crew Scheduling Problem
Agg  Aggregate Model
ANN  Artificial Neural Networks
AS  Ant Systems
BDSP  Bus Driver Scheduling Problem
COP  Combinatorial Optimisation Problem
CPP  Crew Planning Problem
CSP  Crew Scheduling Problem
DC  Duties based Coding
DSS  Decision Support System
ERP  Enterprise Resource Planning
ES  Evolutionary Strategy
ESS  Exploitation Support System
GA  Genetic Algorithm
ILP  Integer Linear Programming
MOGA  Multiobjective Genetic Algorithm
MOP  Multiobjective Optimisation Problem
SOP  Single Objective Optimisation Problem
ND Non-Dominated

PWC Pieces-of-Work Coding

SA Simulated Annealing

SCP Set Covering Problem

SPEA Strength Pareto Evolutionary Algorithm

SPP Set Partitioning Problem

TS Tabu Search
Chapter 1

Introduction

The research presented in this thesis addresses the application of multiobjective genetic algorithms to the Bus Driver Scheduling Problem (BDSP). The BDSP is a complex Combinatorial Optimisation problem largely studied by the Operational Research community since the 1960's. Despite the multiobjective nature of the BDSP, it has been traditionally handled as a single objective problem based on the minimisation of costs. In practice, the BDSP involves several conflicting objectives and, often, such single objective approaches in real world problems lead to solutions difficult or impossible to implement, since they do not correspond to acceptable trade-offs among the objectives, as expected by the planners.

In the last few years, the application of metaheuristics to multiobjective problems has been an active area of research. In particular, genetic algorithms seem to be particularly well suited to multiobjective problems, since the process that guides the search is able to produce several potentially non-dominated solutions in a single run, as well as to exploit the good features of the solutions for improvement purposes. Furthermore, genetic algorithms have proven to be a robust and powerful search technique. Nevertheless until now, only a few genetic algorithms have been proposed specifically for the BDSP.

In this chapter, Section 1.1 describes the context of this work, while Section 1.2 presents its scope and relevance. A list of the main objectives of this work is outlined in Section 1.3 and the structure of the thesis is presented in Section 1.4.

1.1 Context of the research

1.1.1 The operational planning process

Operational planning is a central process in the activity of a mass transit company. It involves the operational implementation of strategic and tactical policies, according to two main objectives that are
often conflicting: improving the quality of the service provided to the customers and minimising the costs involved in the management of the company resources.

To tackle the complexity of the Operational Planning Process (OPP), it is common to decompose it into several different stages, that are grouped in two main phases: the Planning phase and the Rostering phase (see Figure 1-1). In the Planning phase, the "abstract" daily schedules of the vehicles and drivers are designed for a given planning horizon while, in the Rostering phase these schedules are assigned to actual vehicles and drivers. In this process, the number of stages and the scope of each stage may vary with the company, according to its dimension, complexity and strategy. Naturally, whilst the OPP decomposition is needed due to operational reasons, it is also somehow artificial, since many interactions exist between stages.

![Figure 1-1: Stages of the Operational Planning Process](image)

In particular, the work presented in this thesis is concerned the bus driver scheduling stage, that involves a multiobjective decision making process with several simultaneous and conflicting objectives. Bus driver scheduling consists in the definition of the daily work of the drivers, based on the vehicle schedules defined previously. The daily work of a driver, known as a duty, must satisfy a complex set of rules and operational constraints imposed by the legislation as well as by union and agreements internal to the company.
1.1. CONTEXT OF THE RESEARCH

1.1.2 Decision support systems

The first decision support systems (DSS) for operational planning have been developed in the 70's. The rapid evolution of computer technologies and software engineering paradigms has allowed the development of powerful and sophisticated systems to support the whole operational planning process. These systems usually integrate a global and comprehensive database, advanced user interfaces and a set of different software tools for vehicle and driver scheduling.

Currently, in most systems, computer-aided scheduling is performed using graphical interfaces, in which the planners can view and manipulate the schedules, with a strong level of interaction. The "black box" concept is no longer used and current computer systems include interactive software tools that allow planners to manipulate the solutions produced in an automatic way.

Following this philosophy, one of the objectives of this work was to fully integrate our approach to the bus driver scheduling problem into an existing decision support system, the GIST System.

The GIST System is a commercial DSS developed and maintained by OPT [39], [142]. The system is currently in operation in nine mass transit companies. It is the result of over 20 years of research in university R&D units (INEGI, at the Faculty of Engineering of the University of Porto [143], and ICAT at the Faculty of Sciences of the University of Lisbon [122], [23]) aligned with the activity of a consortium involving these R&D units and five Portuguese mass transit companies (Carris, STCP, Horários do Funchal, Vimeca and Empresa Barraqueiro).

From our point of view, a DSS in a mass transit company does not simply provide software tools together with graphical interfaces that help to quickly produce "good" schedules. As important as this aspect is to take full advantage of the opportunity to introduce innovation in the companies and to undertake a re-engineering process that involves not only the operational planning process, but also other related business areas.

1.1.3 Portuguese mass transit companies

Portuguese major mass transit companies are concentrated in the two largest cities, namely Lisbon and Porto. Some of these companies are public companies, with administrations appointed by the central government, while others are private companies. Spread over the country, there are smaller mass transport companies, which are either private or public, and in the latter case, they are managed by the local municipalities. While the larger companies offer mainly urban and suburban service, smaller companies provide also regional service and renting services.

In this work we consider nine companies (those that use the GIST System). From these, two can be classified as large companies (Carris and STCP), while the other seven can be considered medium size
companies.

The operational planning process in most of these companies follows the generic structure presented in Figure 1-1, the main difference between the companies being related to the size of the problems.

The largest companies partition the network into smaller units, composed by small groups of routes that are handled separately until the rostering phase. Each of these units corresponds to relatively small problems handled independently at each stage, with little interaction between stages. This approach is justified by three main reasons. The first one is that if all routes were handled together, data management would be a very complex task, due to the large amount of data. The second reason is that these companies, operating in large cities, are subject to frequent changes in the operating conditions and the company must adapt to the new conditions as fast as possible. These changes are usually limited in time and restricted to a given area (involving only a few routes), for example due to a sport event or to roadworks. If the network was handled as a whole, a small change in a single route would imply several adjustments in vehicle and driver schedules from other routes. Finally, if the routes were handled together, the drivers should be skilled to drive in any route of the network. In some cases, these routes may have quite different characteristics.

Contrary to this approach, most medium size companies claim that they can obtain important savings by handling the problem as a whole, since the problem is too small to justify its partition into sub-problems. However, when the operational planning is performed by manual processes, these companies face serious difficulties in data management and in providing an updated and consistent service to the public. In fact, medium size companies present the most complex operational planning processes and, particularly for the driver scheduling problem, we often face instances that involve the whole network, with complex rules and constraints related to other stages of the operational planning process (to vehicle scheduling or to crew rostering).

Both approaches are reasonable and both have positive and negative aspects. However, during our work with these large and medium size companies, we have noticed that in general companies need to adjust the size of the problems. In fact, large companies divide the network into too small units, some of them composed by a single route, while medium size companies have too large and complex problems. Supported by the GIST System, many of these companies are currently involved in a re-engineering process in order to improve their current planning process. Large companies are beginning to build schedules for larger groups of lines, trying to save vehicles or drivers, while medium size companies are studying how the whole network can be partitioned into smaller groups of lines, in order to improve flexibility and data management.

In Portugal, the last twenty years have been fundamental for the increasing interest and investment in this area. In fact, the growing competitiveness and the emergence of computer software programs to
help companies in the operational planning process have had an enormous impact in their operational performance.

1.2 Scope and relevance of the thesis

With this work we expect to contribute to the definition of a new perspective in tackling the bus driver scheduling problem, and to provide an innovative multiobjective approach for its resolution based on genetic algorithms. Next, we briefly discuss these two topics of research.

1.2.1 The bus driver scheduling problem

The bus driver scheduling problem has been largely studied since the 60's and several models and techniques have been proposed to solve it. Most of these approaches are based on Set Covering or Set Partitioning formulations, where the objective function consists in the minimisation of the sum of the duty costs, which often include direct costs and penalties. In most cases, set covering and set partitioning models are solved by using integer programming techniques often supported by column generation methods [59]. More recently, several metaheuristics and, in particular, genetic algorithms, have been applied to the set covering and the set partitioning problems with very promising results [16], [104], [17].

All these approaches present several difficulties when applied to real life bus driver scheduling problems:

(i) Assigning costs and penalties to the duties can be a complex task, since it is not always easy to evaluate costs and predict the impact of each penalty in the final solution. The penalties reflect the undesirability of certain particular features of the duty, such as the duty type or the number of vehicle changes.

(ii) Usually, in real life problems, the quality of a solution is not measured taking into account only the individual costs and penalties of the duties. Each solution is also analysed as a whole, considering other features such as the average duration of the duties or the maximum number of duties of a particular type. Although some approaches consider such aspects [105] as additional constraints, these are often relaxed or handled by very simple heuristic procedures, due to their complexity.

(iii) In real life problems, bus driver scheduling solutions often include some driving time that is not assigned to any feasible duty (leftovers). This driving time is usually performed as extra working time or it is assigned to an incomplete (unfeasible) duty. A considerable part of the work of the planners in Portuguese mass transit companies is dedicated to trying to minimise the number (or
the duration) of leftovers. Most approaches in the literature either consider that every solution is completely covered by feasible duties (thus, not including leftovers) or that a leftover is a highly undesirable duty type (with a high penalty).

(iv) The bus driver scheduling problem is usually solved as a single objective optimisation problem, although in fact, several conflicting objectives are always involved. For example, in real life problems, we often must take into account not only costs and quality of service, but also the quality of the duties, as viewed from a drivers' perspective.

This thesis proposes a new approach to the bus driver scheduling process in which all these aspects are considered. We have identified the most relevant issues to be taken into account in the evaluation of a solution, given the multiobjective nature of the problem. We have devoted particular attention to the management of leftovers, which are considered as additional components of a solution, instead of particular duty types.

1.2.2 Multiobjective genetic algorithms

We have designed and implemented our multiobjective approach to the bus driver scheduling problem using genetic algorithms. GAs are currently being successfully applied to several combinatorial optimisation problems, as they are capable of generating high-quality solutions to many problems within reasonable computation times [13], while being relatively easy to understand and implement. Besides, GAs do not impose any particular condition to the objective functions or to the constraints (such as linearity or continuity).

In more recent years, there has been a growing interest in multiobjective genetic algorithms [67], [36] applied to combinatorial optimisation problems, but only a very low number of approaches are related to the bus driver scheduling problem [107], [94], [53].

In this thesis we propose two different multiobjective genetic algorithms for the bus driver scheduling problem, based on two models: the Aggregate (Agg) model and the Non-Dominated (ND) model.

The Agg model follows a traditional approach in which all criteria are merged into a single objective function by using an appropriate weighting function. This model is very simple and computationally efficient, and its major problem is how to determine the appropriate weights. Hence, it is better for companies where the planners know in advance the relative importance of the objectives. It can also be used to generate non-dominated solutions that can be used as initial solutions for the ND model.

The ND model uses the concept of Pareto-optimal solutions and was inspired on the Strength Pareto Evolutionary Algorithm (SPEA) [162], [164], that keeps the non-dominated solutions in an external set and uses Pareto dominance concepts for the fitness assignment process. In this model each criterion is
1.3. OBJECTIVES OF THE THESIS

evaluated separately and a set of alternative non-dominated solutions is maintained. We have introduced some new concepts, namely the density of a population, dominance sharing of a solution and total sharing of a solution and we have used these concepts to design new fitness functions for the external set and for the population.

We have developed several knowledge-based genetic operators that are used by both models. In our experience with real companies, we have identified several different criteria playing an important role in the evaluation of driver scheduling solutions. Tests performed on a set of real problems from different Portuguese companies show that our approach is effective in providing a varied set of good alternative solutions.

Our multiobjective genetic algorithm approach was integrated in an interactive and user-friendly software tool, GenT, also developed in the scope of this thesis. This application is embedded in the GIST System, a decision support system being used in the major Portuguese mass transit companies, although it can also be used as a stand-alone application (see Figure 1-2). GenT is an interactive tool designed to support planners in quickly producing good solutions under different operating scenarios. Within this application planners can, during the execution of the algorithms, choose a different set of criteria, change parameters of the algorithms, save intermediate solutions or introduce previously obtained solutions. In their planning activities, planners are able to evaluate and analyse a wide set of alternative solutions. Furthermore, different scenarios can be tested and evaluated in a rather short time. The impact evaluation of changes in the operating conditions or in the duty rules is highly valuable to support union negotiations or when changes in the tactical policies of the company occur.

1.3 Objectives of the thesis

The main goal of this thesis was to develop and implement a new approach to the bus driver scheduling problem, that takes into account the multiobjective nature of the problem, involving a comprehensive set of constraints and objectives, and that can be effectively daily used to support decision making by planners.

In order to achieve this quite broad goal we have defined the following five more precise objectives:

1. To analyse in detail the structure of the problem in order to identify the components that may influence its complexity.

The bus driver scheduling problem is recognised as a very complex problem by the Operational Research community. However, quite often, this complexity is strictly analysed from the computational or algorithmic point of view. However, in our work of several years with companies we have noticed that several other aspects influence the complexity of the problem. They have to do
with the planning methodologies and concepts and may vary from company to company and even within problem instances from the same company.

2. To identify and to study the set of constraints and objectives with impact on the quality of a solution.

   In mass transit companies, the evaluation of the quality of a solution is a complex process that should take into account a large set of (soft) constraints that are often relaxed by traditional approaches. Furthermore, each solution should be analysed as a whole, by considering the trade-off of several different objectives. In fact, many "optimal" solutions are often very difficult to implement in practice. Planners usually claim that some important (soft) constraint has not been considered, or that although a particular objective was achieved, another important objective is far from the desired value.

3. To design a multiobjective approach for the bus driver scheduling problem.

   This approach should easily handle the set of constraints and objectives found in the previous stage. It should also quickly provide good acceptable solutions. An approach based on multiobjective genetic algorithms was chosen for these purposes.
4. To validate our approach with real problems from different mass transit companies.

Since our work has been developed in deep collaboration with several mass transit companies, our approach should be tested against the solutions currently implemented in these companies. For comparison purposes, we have applied the criteria actually used by the companies.

5. To implement our approach in a real context, by developing a software application that can be effectively used by operational teams in their daily planning work.

This application should be interactive and user-friendly, allowing the planners to select different combinations of objectives and soft constraints, to control certain parameters of the algorithms, and to quickly evaluate the quality of the solutions provided. The application should also be available for R&D teams for research purposes.

1.4 Structure of the thesis

The remainder of this thesis is organised in the following chapters.

Chapter 2 describes the operational planning process in mass transit companies, discussing in detail the case of Portuguese companies. Then, the bus driver scheduling process is formalized and discussed. Several decision support systems available for operational planning are presented, with particular emphasis on the GIST System.

Chapter 3 presents a literature review on the different approaches to the bus driver scheduling problem.

Chapter 4 outlines the main concepts of multiobjective optimisation, discusses some of the traditional approaches to these problems, and presents a brief survey of the some well-known genetic algorithms to multiobjective optimisation.

Chapter 5 introduces our innovative approach to the bus driver scheduling problem, discusses the main aspects that determine the complexity of each problem and the main aspects to be taken into account in the evaluation of solutions. Based on this new approach, we propose and discuss a set of hard and soft constraints to the problem, as well as several objective functions.

Chapter 6 starts with a brief discussion on the importance of an appropriate choice of the coding scheme in genetic algorithms. The chosen coding scheme is presented and discussed. We then present the two multiobjective genetic algorithms we have designed. One is based on the Aggregate model and the other is based on the Non Dominated model. For these algorithms, the several operators and the fitness assignment procedures are described. At the end of the chapter, a brief discussion on diversification strategies is provided.
Chapter 7 presents and discusses the results obtained in a set of benchmarking test problems as well as in a set of real problems from three different mass transit companies (Carris, STCP and SMTUC).

Finally, Chapter 8 summarises the main contributions of this thesis, and presents some topics for future research.
Chapter 2

The Bus Driver Scheduling Problem (BDSP)

2.1 Introduction

The bus driver scheduling process involves a detailed planning of the daily work of the drivers, in order to guarantee that a predetermined set of trips will be performed. It is usually integrated in the more global operational planning process of a transport company.

In Section 2.2 we describe the operational planning process (OPP) which, due to its complexity, is usually divided into several different phases. We will focus on the particular aspects of the operational planning process in Portuguese mass transit companies.

Driver scheduling is known to be a particularly complex stage of the OPP, due to its combinatorial and multiobjective nature. In Section 2.3 we present the fundamental concepts of the problem. We also describe a comprehensive set of rules and constraints that are commonly considered in Portuguese mass transit companies. A generic manual procedure reflecting many common practices is then described.

Finally, in Section 2.4, we present some of the most well known Decision Support Systems (DSS) that are commercially available for operational planning and, in particular, the GIST System, which was used to test and to implement the new approach developed in this work.
2.2 The operational planning process in Portuguese mass transit companies

The operational planning process (OPP) involves the operational implementation of strategic and tactical policies according to two main objectives that are often conflicting: improving the quality of the service provided to the customers and minimising the costs involved in the management of the company resources. This planning process is central to the company business, as it will determine how efficiently the company resources will be used in satisfying the demand.

- The OPP is preceded by a global definition of the service the company is going to offer. The decisions that guide the definition of the company service, usually based on origin/destination inquiries and also on political decisions, are out of the scope of this work.

To tackle the complexity of the OPP it is common to decompose it into several different stages. The number of stages and the scope of each stage may vary with the company, according to its dimension and complexity. Naturally, whilst the OPP decomposition is necessary due to operational reasons, it is also somehow artificial, since many interactions exist between stages. Smaller companies, with comparatively easier problems, tend to separate the OPP into less stages than larger and more complex companies. However, smaller problems are sometimes overcharged with a large set of rules and internal procedures. Hence, we observe an apparent paradox regarding the complexity/dimension binomial of the operational planning process: smaller companies tend to have more complex problems than bigger ones.

2.2.1 The operational planning phases

In this section we describe the decomposition undertaken by most of the medium/large mass transit companies. We identify two major operational planning phases: the planning phase and the rostering phase (see Figure 2-1) which, in larger companies also correspond to different planning teams, sometimes working in different places. In the planning phase, the offered service is defined, the necessary resources are calculated and the corresponding schedules are created. In the rostering phase, the drivers planned work is assigned to real drivers and it can be further corrected according to the daily operating conditions.

The planning phase

In this phase, the planned work of vehicles and drivers is defined for a large planning horizon (a given season, for example), although minor changes can be periodically made. Within the planning phase, we identify four different stages: Basic Information Management, Timetabling, Vehicle Scheduling, and Driver Scheduling. The division into these four stages corresponds to the traditional approach [143], although such division is not adopted in many small companies.
2.2. THE OPP IN PORTUGUESE MASS TRANSIT COMPANIES

Stage 1: Basic information management

This stage corresponds to collecting and structuring all the information that is the basis for decision making in the next stages. Basically, it includes the characterisation of the transport network and the definition of all the paths, routes (or lines) and groups of lines.

The transport network is defined by a set of nodes (or points) and a set of directional segments connecting pairs of nodes. A node corresponds either to a depot or to a bus stop with a particular role in the planning process, such as a terminus, a control point or a relief point. A segment connecting a pair of nodes is used to identify admissible paths and its duration varies with the day period. The number of day periods and the corresponding durations depends on the company but commonly we identify five day periods: two of them correspond to the morning and afternoon rush hours, another two correspond to morning and afternoon off peak periods and the last one corresponds to the night period. Each segment has associated a different duration, according to each day period. There are also different types of segments, depending on types of trips considered. For example, when a segment is used by connection trips (corresponding to operation without passengers that links the depot to a terminus or two different terminuses), its durations along the day will be smaller than when it is used by normal trips. Between two nodes, at least two segments must exist, one for each direction.
A path is a sorted set of segments that are used to define trips. A path is defined between two terminuses or between a depot and a terminus and it is composed by segments of the same type.

A route (or line) is a group of paths that will be used as the basic entity for defining a public timetable. When a set of routes is managed together for scheduling purposes, the company creates a group of routes. This concept is very important, since the company can obtain important savings by reducing the resources needed and, at the same time, improve the quality of the service. Different groups of routes can be created for different management purposes. For example, a group can be formed for vehicle scheduling and another group can be formed for driver scheduling. Essentially, the number of routes that constitute a group depends on the number of trips performed on each route and on the dimension of the company. Once again larger companies tend to form small groups, easier to manage, while smaller companies tend to build larger groups, sometimes involving all the routes of the company.

For defining a group of routes, we need to consider all the paths of each route that will be part of the public timetable and the linking paths, not shown in the public timetable, that will be used to link the depots and the terminuses.

Stage 2: Timetabling

A timetable is defined for each route separately, or for a group of routes that share part of their paths. The process for creating a timetable consists on the definition of the trips for each path, according to a predefined frequency. The information gathered in the previous stage is used to calculate the duration of the trips, based on the durations of the segments on each day period. Due to the strong interactions between timetables and vehicle schedules, these two stages are often merged into a single one.

Stage 3: Vehicle scheduling

In the vehicle scheduling process the trips previously defined are grouped to form the individual vehicle schedules (the running boards). A running board corresponds to the set of trips that a single vehicle must perform and includes the trips offered to the public and the connection trips, which can involve dead running time. The vehicle schedule is the set of all running boards for an individual route or for a group of routes. The main objectives of the vehicle scheduling process are the minimisation of the number of vehicle needed to perform all the trips and the minimisation the running costs. It can be performed manually or using automatic procedures which are usually based on mathematical programming models [124]. In this last case, the resulting solution does, in principle, guarantee that the trips are performed by the minimum number of vehicles. Additional constraints can be added to the problem, namely the number of vehicles that are available [115] or the number of depots [125]. Penalties for some schedule features can be included such as a penalty for dead running time.
Stage 4: Driver scheduling

The driver scheduling process builds the daily duties for the drivers on a route or a set of routes. This stage is based on the vehicle schedule defined in the previous stage. Each driver duty is subject to a set of rules and constraints defined by the government legislation, union agreements and some internal rules of the company. For many companies, the main objective is to minimise the number of drivers required to cover all the trips, although several other objectives can be defined, namely the minimisation of the total costs involved. This problem is very complex and several approaches have been proposed to tackle it (as described in Chapter 3). Since it corresponds to the central problem tackled in this work, it will be described in detail in Section 2.3.

The rostering phase

Once generic driver duties are defined, the next step is to assign these duties to real drivers. In the rostering process, changes are made more often (daily or weekly). The complexity of the rostering phase in a particular transport company depends on the number and nature of the constraints and operational rules existing in the company [63], [21]. Although rostering varies significantly from company to company, we have considered here the general approach proposed by [64], which divides the task into three different stages according to the planning horizon: Long Term Rostering, Regular Rostering and Post Rostering.

Stage 1: Long term rostering

Long term assignments are defined first, ensuring that all the personal constraints and skills of the drivers are respected, such as days-off, holidays and individual skills for driving certain types of vehicles.

Stage 2: Regular rostering

Then, changes on the availability of drivers and on the duties are inserted in a regular short term basis (daily or weekly).

Stage 3: Post rostering

Last hour changes on the duties and on the actual availability of the drivers are tackled in this stage of the rostering phase. The information gathered in this stage will be the basis for the calculation of the wages of the drivers and it can give important operational indicators of the deviations between the planned and the actually performed work.

The OPP approach presented in this section is adopted by most large and medium mass transit Portuguese companies. The decomposition of the process in phases and stages is necessary due to its
complexity and to the amount of resources involved (vehicles and drivers). However, such decomposition implies a lower quality of the resources management, since e.g. a minor change in a bus schedule might lead to the reduction of a driver or less extra hours costs.

In many small mass transit companies the Planning phase is not divided into stages: the timetabling of each line is defined at the same time of the bus and driver scheduling processes. This methodology tends to be impracticable when the company grows, although some recent attempts have been made to solve vehicle and driver scheduling jointly. Some of these approaches will be discussed in Section 3.3.3 of Chapter 3.

2.2.2 Operational planning at Portuguese mass transit companies

Portuguese major mass transit companies are concentrated in the two largest cities, namely Lisbon and Porto. Some of these companies are public companies, with administrations appointed by central government, while others are private companies. Spread over the country, there are smaller mass transport companies, which are either private or public, in the latter case, managed by the municipalities. While the larger companies offer mainly urban and suburban service, smaller companies provide also regional service and renting services.

Usually, the operational planning is a central process in a mass transit company. It is responsible for the management of the main resources of the company (vehicles and drivers), while guaranteeing the quality of the service to the public. It is a very complex task, that involves a team with a deep knowledge of the company and requires different skills and competencies from the people involved.

In Portugal, the last twenty years have been fundamental for the increasing interest and investment in this area. In fact, the growing competitiveness and the emergence of computer software programs to help companies in the OPP have had an enormous impact.

Since this work is based on a long term collaboration with some of the major Portuguese mass transit companies, we do in this section the characterisation of the operational planning process in such companies. We identify the main problems and difficulties in the OPP, as well as the main differences between the companies. We have grouped the companies according to their dimension into three levels: large, medium and small companies.

In Figure 2-2 we present a summary of information characterising nine major mass transit Portuguese companies.

Large companies

The two largest Portuguese companies are public companies under central government administration, operating in Lisbon and Porto and providing essentially urban service. In these companies, the OPP is
2.2. THE OPP IN PORTUGUESE MASS TRANSIT COMPANIES

<table>
<thead>
<tr>
<th>Company Name</th>
<th>City</th>
<th>Property</th>
<th>Km</th>
<th>Pass./Year</th>
<th>N.Buses</th>
<th>N.Drivers</th>
<th>N.Depots</th>
</tr>
</thead>
<tbody>
<tr>
<td>Companhia Carris de Ferro de Lisboa, SA</td>
<td>Lisbon</td>
<td>Public</td>
<td>672</td>
<td>276,700,000</td>
<td>882</td>
<td>1776</td>
<td>4</td>
</tr>
<tr>
<td>Horários do Funchal, Transportes Públicos, SA</td>
<td>Funchal</td>
<td>Public</td>
<td>1,297</td>
<td>74,858,000</td>
<td>378</td>
<td>569</td>
<td>5</td>
</tr>
<tr>
<td>Rodoviária de Lisboa, SA</td>
<td>Lisbon</td>
<td>Private (Grupo Barraqueiro)</td>
<td>1,297</td>
<td>29,900,000</td>
<td>150</td>
<td>267</td>
<td>1</td>
</tr>
<tr>
<td>STUC-Serviços Municipalizados de Transportes Urbanos de Coimbra</td>
<td>Coimbra</td>
<td>Public (Municipality)</td>
<td>502</td>
<td>27,028,000</td>
<td>130</td>
<td>255</td>
<td>1</td>
</tr>
<tr>
<td>STCP-Sociedade de Transportes Collectivos do Porto, SA</td>
<td>Porto</td>
<td>Public</td>
<td>497</td>
<td>222,000,000</td>
<td>549</td>
<td>1214</td>
<td>2</td>
</tr>
<tr>
<td>TST-Transportes Sul do Tejo, SA</td>
<td>Lisbon</td>
<td>Private (Grupo Barraqueiro)</td>
<td>3,209</td>
<td>91,500,000</td>
<td>643</td>
<td>946</td>
<td>8</td>
</tr>
<tr>
<td>Transportes Urbanos de Aveiro</td>
<td>Aveiro</td>
<td>Public (Municipality)</td>
<td>220</td>
<td>2,062,000</td>
<td>45</td>
<td>106</td>
<td>1</td>
</tr>
<tr>
<td>TUB - Transportes Urbanos de Braga-EM</td>
<td>Braga</td>
<td>Public (Municipality)</td>
<td>224</td>
<td>12,040,999</td>
<td>116</td>
<td>196</td>
<td>1</td>
</tr>
<tr>
<td>S. M. Transportes Colectivos do Barreiro</td>
<td>Barreiro</td>
<td>Public (Municipality)</td>
<td>149</td>
<td>21,628,874</td>
<td>79</td>
<td>141</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 2-2: A summary of the main characteristics of nine major Portuguese mass transit companies

very similar to the description given in Section 2.2. The Planning phase and the Long Term Rostering stage are carried out by a central team and the last two stages of the Rostering phase are performed by a team on each depot. Due to the large size of the network, these companies have divided it into smaller units, composed by small groups of routes that are handled separately in the stages until the rostering phase. Each of these units corresponds to relatively small problems handled independently at each stage, with small interaction between stages. At the rostering phase, the procedures differ greatly between the companies, mainly due to the historical background of the union agreements.

Until the end of the 80’s, the operational planning process in these large companies was performed manually. However, for the vehicle and driver scheduling problems, some of the larger companies have cooperated in R&D projects in collaboration with university teams from Porto and Lisbon [39], [142], in order to develop automated systems to help in the process.

Medium companies

Portuguese medium sized companies operate in large cities and are either public (managed by local authorities) or private companies. In most of such companies, the operational planning is done manually, based solely on the accumulated experience of the planners. The information about the network is often out-of-date and the planning information is saved in manual reports. However, in the last few years, some of them have adopted a decision support system (see Section 2.4), that clearly helps them in their planning activities.

Many of these companies perform both urban and suburban services. Each route is a very complex
unit with many variants and paths. This situation results from the demand increase along the years. Vehicle and driver scheduling is often performed at the same time and the network is handled as a whole. Using this holistic approach the companies can obtain marginal gains, while maintaining a complex system of rules and procedures. On the other hand, the information tends to become progressively more difficult to manage and maintain.

Medium sized companies present the most complex operational planning processes, since the problem is too big to be handled manually but it is too small to justify its division into sub-problems. Particularly for the driver scheduling problem we often face instances that involve the whole network, with complex rules and constraints related to other stages of the operational planning process (to vehicle scheduling or to crew rostering).

Hence, the biggest challenge in medium sized companies is the reengineering process that has to occur in order to simplify and rationalise the whole operational planning process, thus increasing its flexibility.

Small companies

Portuguese small sized companies are mainly private and provide both urban and interurban service. These companies have an easy and small operational planning process, that can be tackled manually, by a small planning team. However, the development of these planning processes in small size companies should be carefully planned in order to avoid some of the mistakes that many medium companies have made.

Finally, we conclude this section by providing an analysis of two particular aspects that help to fully understand the present situation of OPP in Portuguese transport companies: the historical background of the companies and the composition of the planning teams.

1. Historical background of the company

The operational planning process results from the summation of many procedures, rules and decisions that have probably been introduced in quite different moments. The final result is in general a very complex process, with many inconsistent, subjective and obsolete rules. In larger companies, many of these rules have progressively been ignored, mainly because it was impossible to manage them all in practice. In smaller companies, a high number of these inconsistent and subjective rules still remain, since the dimension of the problem is still manageable and the information is often centralised in a single person.

We have also observed that, in many companies, the OPP is not perceived as a crucial process by the managers or even by the administration board. In fact, we should emphasise that the planning
process is the basis for introducing and performing new and better services. Certainly, there is no precise ways of measuring if this process could be improved or if the resources could be optimised. The process is too technical and too complex for the managers fully understand it. Moreover, being an internal process, it has no direct impact on the services provided and investments tend to be assigned to other areas such as the acquisition of new vehicles, an Enterprise Resource Planning (ERP) or an Exploitation Support System (ESS).

It should also be noted that investments in this area usually do not lead to an immediate return. Reengineering of the operational planning process as well as investments in software or in the qualification of the planning team surely can give positive gains, but in most cases in a medium to long term basis.

All these aspects affect the way the OPP is structured within the complex organization of the company. In the last few years, large and some medium companies began recognizing the relevance of the operational planning process. This change of attitude necessarily involves restructuring the whole process, including training of new highly qualified planners, enhancing the motivation of the older planners and investing in a decision support system. For smaller companies, this reengineering process can be long, complex and involve relatively high costs, but it corresponds to a safe and consistent approach for gaining competitiveness.

2. The planning team

Traditionally, the operational planning teams were composed by ex-drivers or other people with low academic qualifications, but with a deep knowledge on the particular features of the network, the drivers and the operating rules of the company. In general, such teams were small and overwhelmed with the daily changes on the operating conditions. Hence, there was few time available for what-if analysis and careful medium term planning.

In the last few years this situation has changed in large companies. There was an investment in high qualified personnel and in the acquisition of decision support systems for operational planning. These measures are being adopted by some of the medium companies, which are restructuring the planning team and the whole operational planning process. However, in the majority of medium and small companies, OPP is still centralised in a single person that gathers all relevant information. The procedures and rules are based on the experience accumulated in many years and are not formally written or expressed. Replacement of such persons and training of new planners are very difficult and time expensive tasks.

A consistent investment in personnel qualification and in forming a sound, skilled planning team, as well as in adopting the support of new technologies, are therefore key issues for the development
and competitiveness of these companies.

2.3 Bus driver scheduling

The bus driver scheduling process presented in this section corresponds to the fourth stage of the OPP, as described in Section 2.2.1. This process consists in the construction of the driver duties for a predetermined vehicle schedule.

A driver duty corresponds to the daily work to be performed by each driver. It is composed by a set of trips, already assigned to vehicles, and it is subject to a set of rules and constraints defined by the legislation, by the union agreements and by internal procedures. The work of each driver is mainly composed by driving time, but it may also contain some non-driving time (corresponding, for example, to the walking time from one relief point to another).

Scheduling drivers has been acknowledged to be a complex task, even for experienced planners working with the support of sophisticated software tools. It has been described by many authors (see e.g. [154], [131], [42], [51], [40], [152], [149]). In what follows, we present the fundamental concepts of this problem as well as the main rules and constraints that exist in the Portuguese transport companies that we have been working with.

However, we would like to note that the general approach present in this work is not the only one for the bus driver scheduling process. Some companies tackle the vehicle scheduling and the bus driver scheduling problems simultaneously [73], [70] while some other companies, namely those with extra-urban service, assign the vehicle to the trips after solving the bus driver scheduling problem [127], [146]. Some technical aspects of these approaches will be described in Chapter 3.

2.3.1 Input data from vehicle scheduling

The vehicle scheduling process provides some essential input data for driver scheduling, namely the relief points and the relief opportunities. The additional data necessary for the driver scheduling problem is related to legislation, union agreements or internal procedures of the company.

Relief points and relief opportunities

A relief point is a predetermined location point at which a driver can be replaced by another. Usually, the depots and the main central bus stops are defined as relief points. The number of relief points is of major importance for driver scheduling and varies significantly from company to company. While most of the companies define a set of relief points for each route or group of routes, other companies define a single relief point for all routes (the depot) or as many relief points as bus stops.
Each vehicle, according to its schedule (running board), can pass by a relief point several times during the day. Hence, associated to each relief point we define the passing times for each bus working day. A pair relief point / passing time is known as a *relief opportunity*.

**Pieces-of-work**

The time interval between two contiguous relief opportunities is defined as a *piece-of-work* and it corresponds to the minimum working unit for a driver. The total number of pieces-of-work is a major factor in determining the complexity of a particular bus driver scheduling problem. Clearly, this number is affected by the variety of relief points and also by the number of relief opportunities.

Generally speaking, routes that are mainly in urban service tend to have several relief points physically close to each other, and hence the relief opportunities are also very close. Consequently, the resulting pieces-of-work are very short (sometimes with durations of 10 or 15 minutes). On the contrary, routes that provide suburban service have longer pieces-of-work, with durations of 1 or 2 hours.

In Figure 2-3 we present a diagram of two running boards. The first bus leaves the depot ("Estação") at 7:18 and passes at the first relief point ("Av Lib CTT") at 9:12. The time interval between the two relief opportunities defines a piece-of-work that will be assigned to a single driver. Throughout its day this bus will pass once at relief point "Av Central" and several times by "Av Lib CTT". This running board ends at the depot at 19:43. The second running board is composed by three different fragments and involves an additional relief point ("Gualt U.M"). Each fragment starts and terminates at the depot. The time a bus is stopped at the depot is considered non-driving time and is not assigned to a driver, even if the bus is operated inside the depot. Usually, a fragmented running board, like the second one, reflects the need of increasing the number of vehicles in operation at certain periods of the day. These periods correspond to peaks of demand that are mainly determined by the rush hours.

![Figure 2-3: Representation diagram of two running boards](image)

The particular configuration of a bus schedule is another important factor in the complexity of the problem. Although this factor cannot be directly measured, as was the case of the number of pieces-of-work, it affects directly the driver scheduling process. A bus schedule with many fragmented running boards is usually more difficult to tackle than one with single journey running boards, particularly
when the fragments are short. In fact, the time the bus is at the depot (thus not needing a driver) is fundamental in the construction of efficient duties.

Once again, the configuration of a bus schedule for urban service often differs from one for suburban service. Urban routes are characterised by having many running boards with a single, long journey and a few fragmented running boards at some periods of the day (usually, the morning and the afternoon peaks). Routes that are not exclusively urban may have a lot of vehicles with several short fragments, that do not correspond to any particular day period.

Finally, it is also important for driver scheduling to know the travel time between each pair of relief points. Such information is important in order to put together two pieces-of-work involving different relief points. This issue will be further detailed in the next section.

2.3.2 Rules for driver duties

In order to construct drivers duties, the planner has to satisfy a set of rules, some being periodically negotiated and agreed with the unions while others are general rules imposed by the labour legislation. Some particular rules are determined by the historical background of the company. A duty that satisfies all these rules is called a feasible duty.

In this section we give a comprehensive description of the set of rules that we have identified as the most important for the generality of the companies. Such set of rules, or a subset of it, applies to all Portuguese mass transit companies we have worked with, and we believe (based on some preliminary international experience) that it can be easily adjusted to any mass transit company. Naturally, the particular values for each rule differ from company to company. Some particular additional rules, used only by a single company and which are not likely to be adopted by any other company were not included in this description.

The rules governing the construction of the duties can be divided into two types: the global rules and those involving the duty types.

Global rules

Global rules are applied to all duties, independently of their type. Some of these rules are hard in the sense that they have to be strictly respected, while other rules are soft and can be slightly relaxed.

In Table 2.1, we present a summary of some global rules. The Start and Finish values define the time interval for the corresponding period. From our experience, we believe that there is a single hard global rule which is the Night Period (Rule N. 3). In fact, since the night work involves extra payment to a driver, it is strictly respected by the company. On the contrary, the other global rules are more flexible, in the sense that most companies do even allow some violation if a better solution can be obtained. The
2.3. BUS DRIVER SCHEDULING

<table>
<thead>
<tr>
<th>N.</th>
<th>Rule abbreviation</th>
<th>Rule description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LunchPer</td>
<td>Lunch Period</td>
<td>Start  Finish</td>
</tr>
<tr>
<td>2</td>
<td>DinnerPer</td>
<td>Dinner Period</td>
<td>Start  Finish</td>
</tr>
<tr>
<td>3</td>
<td>NightPer</td>
<td>Night Period</td>
<td>Start  Finish</td>
</tr>
<tr>
<td>4</td>
<td>DayXtraWrk</td>
<td>Extra work at Day Period</td>
<td>Max</td>
</tr>
<tr>
<td>5</td>
<td>NghtXtraWrk</td>
<td>Extra work at Night Period</td>
<td>Max</td>
</tr>
</tbody>
</table>

Table 2.1: An example of a set of global rules

extent of violation depends on each particular company. For example, some companies are very rigid concerning the lunch period, while the dinner period is flexible. Moreover, some companies allow larger violations, if they are limited to a small number of duties. For example, they can allow a 30 minutes violation on the lunch period if it applies to a single duty.

Next, we present some concepts to be used in the forthcoming description of the rules involving the duty types.

A **spell** of work is a set of contiguous pieces-of-work in the same vehicle.

A **stretch** or a **shift** is a series of spells without a break.

A **break** corresponds to a period of time assigned to a driver for him to rest between two stretches. Usually, this time is not considered as working time, thus it is not paid by the company. When a break occurs in the lunch or dinner period it is called a mealbreak.

The **walking time** between relief points is the time necessary to walk from one relief point to another.

A **duty interruption** is the time interval between two spells of work belonging to the same stretch. Usually these intervals are considered as working time (but not as driving time) of the duty and they are paid by the company. The walking time must be respected and included in the duty interruption if the relief points involved in the two spells are different.

In Figure 2-4 we present a graphical representation of a driver schedule in order to illustrate these concepts. In this figure, duty 4 is composed of two stretches separated by a meal break. The first stretch starts at 7:15 and finishes at 11:20 and includes two spells, separated by a duty interruption. The second stretch starts at 13:00 and finishes at 16:05. Duty 5 is composed of three stretches, each one with a single spell. Both duties 1 and 2 are composed of two stretches (each one with a single spell). Duty 3 has a single stretch (with a single spell).

**Rules involving the duty types**

Most transport companies classify the drivers duties in different types, according to their particular features. The main feature that distinguishes duty types is the number of stretches (or shifts).
A straight duty is composed by a single stretch that may contain several spells. Since a straight duty does not have any mealbreak (or any break), the meals must occur before the driver starts his work or after he finishes the duty. Most companies validate if the driver can have his meals (lunch and dinner) during the meal periods, even if they occur outside a break. Duty 3 in Figure 2-4 is a straight duty with a single spell.

A split duty is composed by two or more stretches and each of them can contain several spells. The most common are the two-split duties, which have two stretches, usually separated by a mealbreak. If a duty break is not contained in any meal period, companies try to ensure that the driver can have his meals beyond his working period. For example, duties 4 and 3 in Figure 2-4 are two-split duties, while duty 5 is composed of three stretches (a three-split duty).

Split duties with three or more stretches are not very common in large companies, although they are frequent in medium and small companies.

Another common duty type is the long split duty, which is a duty with two stretches and a long break between the stretches. Long split duties are used to cover the peak periods when more buses are in operation. In Portugal, only a few companies allow this duty type, although it is very common abroad, namely in the U.K. [69], since it is very useful to build efficient schedules. However, long split duties are
very unpopular with the drivers, since their total spreadover (the duration between the beginning and end of the duty) is usually very large. Hence, even when they are allowed, companies try to reduce their number to the minimum.

Most companies define night duties, which are straight duties performed during the night period.

Next, we describe the most important rules that govern the duty construction and in Table 2.2, we present a summary of the parameters involved in those rules. Some rules are multidimensional in the sense that they are composed of several entries. The second column indicates the size of the multidimensional rules. The two rightmost columns represent the parameter values. The units of these values are given in brackets in the rule description column.

The most important rules concerning the duty types are those that control the duration of the duty, its stretches and its breaks.

The stretch length is defined as the duration between the beginning and the end of a stretch. It includes the driving time of each spell of work and the non driving time involved in each duty interruption occurring in the stretch. Companies may impose different minimum and maximum durations for each stretch, thus in Table 2.2 we have considered a multidimensional rule (Rule 2) of size $n_1$, where $n_1$ is the number of stretches.

The break length is the interval between two stretches. When a stretch of a duty terminates at a given relief point and the next stretch begins at a different point, the estimated travelling time between the two relief points must be taken into consideration. In most companies, if this time is longer than the break length, the two stretches cannot be joined, otherwise it is included in the break length. However, in some companies, the walking time is included in one of the stretches, instead of in the break length. In this case, it must be added to the stretch length and the minimum and maximum durations of the break will not include the travelling time. Hence, it may be necessary to adjust the value of this parameter to each particular company, since it may be computed in several different ways. Rule 3 in Table 2.2 defines the minimum and maximum length of each break.

The total length of the breaks is the sum of the lengths for each break. However, the minimum (maximum) value of this parameter (Rule 4 of Table 2.2) is not necessarily the sum of the minimum (maximum) values for each break length. Consider for example, a duty type with three stretches (thus, two breaks). We can define a minimum length of 1 hour for each break and impose that the total length of the breaks should not be less than 3 hours.

It is common for drivers to have a meal during one break and the meal duration is limited to a minimum value (see Rule 5 in Table 2.2) that is strictly respected, in most companies. Moreover, the meal must occur within one of the meal periods defined above as global parameters.

The normal duty length is defined as the working time in normal period or, in other words, it does
not include extra working time. It is subject to a minimum and to a maximum value (see Rule 6 of Table 2.2) and, if the sum of the duration of each stretch is larger than the maximum value that was defined, the exceeding time is considered as extra work. The maximum value of this parameter is based on an average value that is estimated according to rostering procedures.

<table>
<thead>
<tr>
<th>N.</th>
<th>Dim</th>
<th>Rule abbreviation</th>
<th>Rule parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NStrch</td>
<td>Number of stretches ( (n_1) )</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>n_1</td>
<td>StrchLgth</td>
<td>Stretch length ( (hh:mm) )</td>
<td>Min Max</td>
</tr>
<tr>
<td>3</td>
<td>n_1-1</td>
<td>BrkLgth</td>
<td>Break length ( (hh:mm) )</td>
<td>Min Max</td>
</tr>
<tr>
<td>4</td>
<td>TBrkLgth</td>
<td>Total length of breaks ( (hh:mm) )</td>
<td>Min Max</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>MealDur</td>
<td>Meal duration ( (hh:mm) )</td>
<td>Min</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>DtyLgth</td>
<td>Normal duty length ( (hh:mm) )</td>
<td>Min Max</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>DrvTime</td>
<td>Driving time ( (hh:mm) )</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>XWLgth</td>
<td>Extra work length ( (hh:mm) )</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>WkTime</td>
<td>Total working time ( (hh:mm) )</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>TSprd</td>
<td>Total normal spread over ( (hh:mm) )</td>
<td>Min Max</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>WkSprd</td>
<td>Total spread over ( (hh:mm) )</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>NDtyInt</td>
<td>Number of duty interruptions ( (n) )</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>NDtyIntStrch</td>
<td>Number of duty interruptions in a stretch ( (n) )</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>LgthDtyInt</td>
<td>Length of duty interruptions ( (hh:mm) )</td>
<td>Min Max</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LgthBefChgVhcl</td>
<td>Length before changing vehicle ( (hh:mm) )</td>
<td>Min</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>LgthAftChgVhcl</td>
<td>Length after changing vehicle ( (hh:mm) )</td>
<td>Min</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>NVhcChg</td>
<td>Number of vehicle changes ( (n) )</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>NVhcChgStrch</td>
<td>Number of vehicle changes in a stretch ( (n) )</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>n_2</td>
<td>NStrDPer</td>
<td>Non-start duties periods ( (hh:mm) )</td>
<td>Start Finish</td>
</tr>
<tr>
<td>20</td>
<td>n_3</td>
<td>NFshDPer</td>
<td>Non-finish duties periods ( (hh:mm) )</td>
<td>Start Finish</td>
</tr>
<tr>
<td>21</td>
<td>StDpt</td>
<td>Start at depot (Yes/No)</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>FshDpt</td>
<td>Finish at depot (Yes/No)</td>
<td>Value</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2: Duty types rules

The driving time is usually considered as the time for a driver to be in a bus without any interruption (continuous driving). However, it is computed differently by the companies. In fact, some companies consider that the time the driver is working inside the bus (even if the bus is stopped at a terminus) is driving time. Other companies consider that idle time in a terminus is an interruption, and hence it is not driving time. This parameter is limited to a maximum value (see Rule 7 of Table 2.2), which is considered very restrictive by the companies (currently, 5 hours in most Portuguese companies). In fact, if the idle time at a terminus is considered driving time, the maximum driving time could constrain the maximum length of a stretch. The second interpretation of the driving time concept allows some companies to extend this limit.

The extra working time of a duty is calculated as the working time that exceeds the maximum normal
2.3. BUS DRIVER SCHEDULING

duty length. It is usually bound to a maximum value, which is presently dictated by the legislation. Usually, the same value is applied to all duty types (see Rules 4 and 5 of Table 2.1), but we have also considered a specific value for each duty type, since a few companies define different values according to the duty type (see Rule 8 of Table 2.2). Some companies assign extra work to the duties at the driver scheduling process while others only take into account the extra hours needed at the rostering stage (namely, at the Long Term Rostering).

The total working time is the sum of the durations of all stretches of the duty, and it is usually bounded by a maximum value (see Rule 9 of Table 2.2). It may include extra working time, if some is present in the duty, for those companies that assign extra work at the driver scheduling process. For the others, this parameter is equal to the maximum duty length.

The total normal spreadover is the duration between the beginning and the end of a duty in normal period. It is calculated by adding the break lengths to the normal duty length, thus it does not include any extra work time. Usually, the total normal spreadover must lie between a minimum and a maximum value (see Rule 10 of Table 2.2).

The total work spreadover is the total duration between the beginning and the end of the duty and it is bound by a maximum value (see Rule 11 of Table 2.2). It is calculated adding the lengths of all stretches and all breaks of the duty and it includes the extra work that was assigned to the duty. For those companies that do not include extra work at this stage, this parameter is equivalent to the maximum total normal spreadover, presented above.

The rules presented so far are among those that most directly affect the duty construction, since they are generally imposed by the central government and union agreements. The purpose of the following rules is to facilitate the work of the driver scheduling planners, while trying to avoid or minimise some undesirable features of the duties. Moreover, the following rules are particularly interesting when an automatic procedure for duties generation is used in the driver scheduling process (see Section 2.3.4).

The number of duty interruptions (see Rule 12 of Table 2.2) may be constrained to a maximum value for each duty type. A duty interruption corresponds to unproductive time that is considered as working time and, hence, it is paid as normal work. Naturally, companies wish to minimise the number of duty interruptions in duties. Moreover, they may want to limit the number of duty interruptions in a stretch (see Rule 13 of Table 2.2). The length of each duty interruption may also be controlled through Rule 14 of Table 2.2.

It also seems reasonable to force a driver to perform a minimum amount of work in a vehicle before and after changing of vehicle. These parameters are defined in Rules 15 and 16 of Table 2.2.

The number of vehicle changes by drivers is also a very important parameter for some companies (specially for the smaller ones), since it is consensual that the smaller the number of vehicle changes, the
smaller the maintenance costs of the vehicles involved. The maximum number of vehicle changes can be determined by Rule 17 of Table 2.2 and the maximum number of vehicle changes in a stretch can be defined in Rule 18 of Table 2.2.

Moreover, some companies define periods in which some duty types must not begin or finish. For example, a day duty should not finish after 22:00, or a straight morning duty should not start after 11:00. These day periods are defined in Rules 19 and 20 of Table 2.2.

Additionally, certain duty types must start or finish at the depot (see Rules 21 and 22 of Table 2.2). For example, night duties usually finish at a depot but some companies also impose this rule to long split duties.

2.3.3 Overall rules

A driver schedule must also satisfy a set of overall rules that concern to the whole schedule and not only to each particular duty. These rules are based on internal procedures that aim at homogenising driver duties and at improving the quality of the global solution. The set of overall rules differs greatly between companies. Typically the rules are subjective and flexible and in most situations represent a set of goals to attain, rather than a set of constraints. In Chapter 5 we describe in detail some of the most important overall rules, since they play an important role in our approach to the bus driver scheduling problem. In this section we just list and briefly explain some of them.

In the driver scheduling process, duties are abstract units of daily work that are not yet assigned to any real driver. The assignment of drivers to duties is performed in the rostering phase of the OPP and the workload of the drivers is then computed for a specific number of weeks or months. Typically, a driver performs a different duty each week or each day, according to a rotation group in a way that is specific of each company [63], [21]. The legislation imposes that a driver should not work more than a maximum number of hours per week. On a daily basis, such constraint should be reflected by a daily average workload. Hence, the particular length of a duty is not very important to a driver, as long as it is within the feasible limits, but the week workload surely is. In fact, the maximum duty length parameter, described in the previous section tends to be larger than the day workload imposed by the legislation. Companies must ensure that all the drivers have similar workloads and as close as possible to the maximum allowed, in order to minimise the number of drivers.

Controlling the average duration of the duties in a driver schedule is a common process used by transport companies to balance the drivers workload. If the average duration of the duties in the whole schedule is close to the desired value, it is easier for the rostering planners afterwards to prepare balanced rotation groups. Other companies prefer to impose an average duration to each type of duty, since their rotation groups are composed of duties of the same type. Hence, instead of a single parameter for a
driver schedule, we have a parameter for each duty type.

It is also recognised that some duty types are very unpopular among drivers. This is the case, for example, of the long split duties, which have a long spreadover. Hence, some companies try to limit the occurrence of these duty types, imposing a maximum percentage for them in the driver schedule.

On the contrary, certain groups of drivers are used to be assigned to some particular duty types. In order to take this situation into account, companies try to impose a lower bound to the number of duties of these types.

Certainly, most of the overall rules cannot be completely satisfied, but the planners need to keep them in their minds, and the driver scheduling process should be guided by such rules. In some companies they are perceived as flexible guidelines but, for other companies, this set of rules is very rigid, particularly those concerning the duration of the duties.

### 2.3.4 The driver scheduling process

Once duty types have been defined and all information from vehicle scheduling has been gathered, the driver scheduling process consists in constructing a set of duties, according to the predefined types, in such a way that they all together cover the trips defined for the selected vehicle schedule. Roughly speaking, one can say that the main objective of the planners is to minimise the costs related to the drivers, although this is not that straightforward, as we will see in Chapters 5. The process requires deep knowledge of both the transport network and the rules to be applied. It is therefore usually performed by an experienced planner.

The *driver scheduling process* can be entirely performed by hand, or it can be supported by software tools that guide and help the manual procedure. The available software tools, of different complexity, can in some cases lead to an almost fully automated driver scheduling process. A description of the most widely used computer approaches will be presented in Chapter 3.

**Manual scheduling**

When performed manually, the driver scheduling process consists in an iterative process, in which the planner generates feasible duties that fit together to cover the vehicle schedule. He will probably start by creating duties that cover the first and the last trips of each running board, and in a subsequent step he will attempt to cover the trips of the middle of the day, by trying different combinations of duties. In such process, similar to a puzzle game, duties are adjusted and changed several times in order to completely cover the bus schedule.

However, most of the times it is not possible to fully cover the bus schedule with feasible duties. The
final schedule includes some pieces-of-work that remain uncovered. Such pieces-of-work, called leftovers are one of the main problems for many mass transit companies, since they represent work that has to be done, probably with additional costs. In practice, leftovers are either assigned as extra work or as incomplete duties to the drivers. Both solutions involve extra costs to the company that should be minimised.

In order to solve the problem of leftovers, the planners iteratively try one or more of the following options, not necessarily by this order:

1. **Violate some of the predefined rules.** Most of the companies prefer a small violation of one or more rules to the payment of additional costs. The extent of the violations depends on the particular company and it may involve a considerable number of duties. The resulting schedule usually becomes composed of several infeasible duties. This fact is implicitly accepted by both the company and the unions as long as it does not seriously violate the social rights of the drivers.

2. **Try to combine leftovers in different driver schedules.** The planner gathers all the leftovers resulting from several driver schedules and tries to create feasible duties by putting them together. This solution is very frequent when the driver scheduling is performed line by line, which usually occurs in large companies.

3. **Leave leftovers unchanged and let the rostering planner assign them to drivers.** This option postpones the problem to the next stage but, in fact, the rostering planners have more detailed information about the workload of each driver and on their predisposition to accept extra work. Moreover, some companies impose that extra work cannot be assigned on a regular basis, but may be offered to all drivers who must might then apply to perform it.

4. **Assign the uncovered pieces-of work as extra work.** Some companies add each leftover (or a part of it) to an existing feasible duty, as extra work. This option is frequent in small companies, without enough drivers to perform all the planned service. Often, drivers are accustomed to perform some daily extra work and consider it as a way to get a salary complement in a regular basis. These situations are very complex and difficult to change, and often require a deep but careful reengineering process.

5. **Build incomplete duties.** This option is only possible if the company has drivers available. It involves additional costs, since the drivers would be inactive and unproductive part of their time. However, some companies consider that those idle drivers can be useful to replace missing drivers (that are in holidays or ill).
2.3. BUS DRIVER SCHEDULING

Anyway, leftovers are not a minor problem, at least in Portuguese companies. For many of these companies, the management of leftovers is one of the most important issues of the driver scheduling process.

In manual scheduling, a planner usually tries to obtain the minimum number of drivers needed while guaranteeing the satisfaction of the overall rules. When the planner finds a satisfactory solution, he will probably only change it when alterations on the bus schedule occur. This means that almost never will he start to build a new schedule from scratch.

Finally, it should be noted that, several years ago, the manual scheduling process as described was made directly on paper. Nowadays, manual scheduling can be supported by sophisticated computer graphic interfaces, like those described in Section 2.4. However, in this work, we do still consider these approaches as manual processes, given that the used tools do not involve automatic generation of duties.

**Computer scheduling**

Today, there is a great diversity of computer based techniques that automatically produce driver schedules. These techniques can vary from heuristics that simply simulate the manual scheduling processes to sophisticated mathematical programming models. Many techniques make use of a combination of mathematical programming techniques and heuristics. In Chapter 3, a selection of these approaches will be presented in detail.

However, from our experience with Portuguese companies, the driver scheduling process always benefits from human interaction. Even when highly sophisticated and advanced computer techniques are applied, there are often small adjustments and improvements that require a direct human interaction. Moreover, the skills and acquired knowledge of a experienced planner are not easily translated to rules that can be included in a software application.

Modelling a complex system is always a very difficult task in which a trade-off between efficiency and effectiveness must be considered. On the one hand, the most complete models, which are as close as possible to the real system, are often inefficient if not intractable. Hence, they are not effective, since they are not able to solve the real problem. On the other hand, less complex models, which can be solved efficiently in a reasonable period of time, do sometimes introduce too many simplifications and neglect important aspects of the problem. Such models are often very far from the real problem and, in this sense, they are not effective since they solve an easier but different problem.

Nevertheless, computer techniques for driver scheduling are presently widespread in mass transit companies, since they represent a powerful and fundamental tool for the planning process. With their support, planners can produce better schedules, evaluate the consequences of rule changes and test different alternative scenarios.
2.4 Decision support systems for the operational planning process

2.4.1 General functionality and features

The first decision support systems (DSS) for operational planning have been developed in the 70's. From those days to now there was a huge evolution in computer technology and the most advanced systems integrate a lot of sophisticated features. In this section we describe the main features of a decision support system for the OPP and we briefly present some of the most used commercial software packages. We describe in more detail the GIST System, which is the system running our metaheuristic approach to the bus driver scheduling problem, and is currently in use by the major Portuguese mass transit companies.

The rapid evolution of computer technologies and software engineering paradigms has allowed the development of powerful and sophisticated decision support systems to help the whole operational planning process. These systems usually integrate a global and comprehensive database, advanced graphic interfaces and a set of different software tools for vehicle and driver scheduling. The production of statistics and customised reports are also important aspects that must not be neglected. Finally, importing and exporting information from and to other computer systems existing in the company is another key issue for the success of such a decision support system.

- The information system

The operational planning process deals with a lot of information, from the transportation network to the personal data about the drivers. Part of a decision support system in this area can be viewed as an information system that guarantees that all this information is integrated and consistent. Information should be easily updated or changed and aggregated information should be easily obtained. A comprehensive database must provide detailed information about each driver duty, including information about each trip and the times the bus passes at specific points. A report should be produced and provided to the driver that will perform the duty.

- Computer-aided scheduling

Despite the emphasis on information management, the scope of a decision support system is obviously much broader. It must also provide software tools to support and guide the decisions of the planners. Currently, in most computer systems, computer-aided scheduling is performed using graphic interfaces, in which the planners can view and manipulate the schedules using a diagram with which they can interact (an example of a diagram for driver scheduling is given in Figure
2.4. DECISION SUPPORT SYSTEMS FOR THE OPERATIONAL PLANNING PROCESS

Some of these systems provide simple heuristics or computation tools that help manual scheduling, while others include sophisticated optimisation algorithms that automatically produce the "optimal" solution for the adopted model. The "black box" concept is no longer used and current computer systems include interactive software tools that allow planners to manipulate the parameters and tune the solutions produced in an automatic way.

- Consolidating information

Operational planning is not an isolated process within the transport company. On the contrary, it is integrated in a complex and dynamic environment which involves many people with different tasks. Furthermore, it concentrates an important part of the company activity: planning a regular and high quality transportation service to the public, with an optimised utilisation of resources (basically vehicles and drivers). Hence, many advanced decision support systems also include interfaces with other relevant legacy systems, such as GIS (Geographic Information Systems), Exploitation Support System (ESS) and Enterprise Resource Planning (ERP).

Finally, we would like to emphasise another important aspect concerning decision support systems for the operational planning process. Each transport company has its own set of procedures, rules and culture. A DSS should be flexible and easily adaptable to different operating realities and policies. In fact, a mass transit company starts the implementation of a DSS because one of the following two situations has occurred: the company does not have any system of this type and experiences its need, or it already has one that does not work properly. Both situations involve high implementation failure risks. On the one hand, the DSS will have to be able to fulfil the high expectations of the planners, when it is the first one to be used by the company. On the other hand, it will have to compete with an existing well-known system. Negative attitudes to an emergent change in the process often occur, but they are not always reported to the managers or to the Administration Board.

In conclusion, the implementation of a DSS for the operational planning process may take a long time to be successful, even if success here is very difficult to assess and measure. The high investments involved in purchasing and maintaining these system are clearly an obstacle to their introduction or replacement. Nevertheless, a general DSS evaluation model was proposed by [55] and applied to the GIST System.

2.4.2 The GIST System

The GIST System is a Portuguese commercial DSS developed and maintained by OPT, SA [39], [142]. It is in operation in the largest Portuguese mass transit companies (Carris, STCP, Horários do Funchal, Rodoviária de Lisboa, Transportes Sul do Tejo, Transportes Urbanos de Braga, Serviços Municipaliza-
dos de Transportes Urbanos de Coimbra, Transportes Urbanos de Aveiro and Transportes Urbanos do Barreiro). It is the result of over 20 years of research in university R&D units (INEGI, at the Faculty of Engineering of the University of Porto [143], and ICAT at the Faculty of Sciences of the University of Lisbon [122], [23]) aligned with the activity of a consortium involving these R&D units and five Portuguese mass transit companies (Carris, STCP, Horários do Funchal, Vimeca and Empresa Barraqueiro).

System structure

GIST is a fully modular system, based on a global integrated database system. Its current version comprises the following modules:

- **Basic Information Management Modules**: these modules allow the definition and maintenance of the data for the whole network, including depots, paths, routes, relief points, distances and travel times.

- **Timetabling and Vehicle Scheduling Module**: this module integrates the production of public timetables and the vehicle scheduling processes, since these processes are closely related. Vehicle scheduling is supported by a set of algorithms based on mathematical programming models [23], [124]. These algorithms include single and multi-depot problems [115] and also constraints imposing a fixed number of vehicles [125]. The system also provides a graphical representation of the schedules, which can be used for the definition of trips and also to modify or even to manually construct the vehicle scheduling solutions.

- **Driver Scheduling Module**: this module, concerning the assignment of drivers to the vehicle schedules defined in the previous module, integrates a graphical representation of the schedules and duties and several heuristics to support the driver scheduling process [56], [129], [107], [53], including the innovative approach presented in this thesis. The graphic representation can be easily manipulated by the planners to build the duties, or to change the solutions presented by the algorithms. This module will be described in more detail below.

- **Crew Rostering Module**: this module handles the actual assignment of driver duties to specific crews or drivers, taking into account personnel days-off [128], holidays and driver's requests. It includes a heuristic based on the application of a sequential user-defined set of rules that, in some companies, produces the actual assignment of the totality of the drivers. This module has proved to be robust, flexible and adaptable to a wide range of different situations.

The GIST System also provides custom reports and operational statistics and custom interfaces to other legacy information systems. Further information about the GIST System can be found in [142],
The driver scheduling module in the GIST System

The objective of this module is to assign driver duties to the trips defined in the Timetabling and Vehicle Scheduling module. It has an advanced graphical user-interface component to help the representation and the manipulation of duties, and to support the evaluation of solutions.

Individual vehicle schedules (running boards) are presented in such a way that the relief points and depots are highlighted. A user can visualise and manipulate the relief opportunities directly on a window, as the one presented in Figure 2-5 or on a graphic as the presented in Figure 2-6. Each relief opportunity is either in an active or in an inactive state. When a user deactivates a relief opportunity, it is no longer considered for duty construction. Since in an urban context some relief opportunities are very close, most users start the process by deactivating those relief opportunities that most unlikely would be used in a driver changeover. At any stage of this process, users can also change the time associated to a selected relief opportunity. This is often used to adjust the final solution or to change a duty that violates some constraint, making it feasible.

![Relief Opportunities](image)

Figure 2-5: Window for visualising and editing relief points

Figure 2-6 presents two windows with a graphical representation of a duties schedule. The top window
presents a schedule for which the duties have already been defined and each duty is represented by a different color. The lower section of the window presents an "empty schedule", in which duties have not been created yet.

Figure 2-6: Windows for visualising and manipulating duty schedules

Duties must respect labour legislation rules as well as the additional rules defined by the company. For each duty type, the definition of the rules to be satisfied is done in accordance with an extensive set of parameters (see Figure 2-7).

If a user wishes to manually manipulate a duty, he can do it directly on the graphic, using the mouse or the keyboard, while visualising the most important features of the duty (see the small window at the center of Figure 2-6).

Moreover, the user can visualise, in the window presented in Figure 2-8, the complete set of validated features for the selected duty. For flexibility purposes, the GIST System lets the user build infeasible duties, although it provides alerts regarding the rules that are being violated.
The Driver Scheduling module of the GIST System allows the user to manually create the duties or, in alternative, to use one of the algorithms supplied. The algorithms are based in metaheuristics [56], [129], [107], [53] (see Section 3.4) and produce several solutions that can be manually adjusted by the user. Furthermore, the system offers the possibility to manually construct some of the duties (a partial schedule) and use the algorithms to complete the solution. One of these algorithms, a multiobjective genetic algorithm is the object of this thesis and will be described in detail in Chapters 5 and 6.

This module also allows the exportation of a driver schedule to a specific file type that can be used in an autonomous module, named Duties Framework which works autonomously from the GIST database. This will be presented in the next section.

The Duties Framework

The Duties Framework is a stand-alone application that imports and exports data from and to the GIST System, converting the database objects into a specific file format. The development of such an autonomous application is justified by the need to have a separate tool for supporting research on algorithms, and was motivated by the following:
Figure 2-8: Window showing the validated properties of the duty

- The GIST System, yet modular and very flexible, is a global software package for operational planning. In particular, the information used and manipulated by the Driver Scheduling module is integrated with the GIST global database. A new driver schedule cannot be created from scratch, without introducing into the system all the information needed by the previous modules.

- The development, tuning and testing of new algorithms for driver scheduling often needs large data sets, that would hugely increase the size of the GIST database.

- The GIST System is an advanced commercial software package, which is regularly updated, with the addition of new developments and tools. The R&D and the software development teams must be in constant dialogue, in order to guarantee that both teams are working with the most recent version of the system.

The Duties Framework has therefore been designed as a light Driver Scheduling module that is not directly integrated in the GIST System. It can be used as a stand-alone application in any personal computer. Initially, it was designed to help the R&D team, providing them with a tool to quickly test
and tune new algorithms.

Users can directly introduce in the application all the information needed by the driver scheduling process, namely the vehicle schedules, the relief opportunities and the set of rules. The information of each particular problem is saved into independent files. In this way, the users and, in particular, the R&D team, can create their own problems. In addition, real problems can be easily imported from the GIST database.

The Duties Framework was developed based on a object-oriented methodology and, hence, the integration of a new algorithm is a very simple process. Furthermore, it provides a graphical user interface similar to that of the GIST System.

These were the reasons that motivated the development of Duties Framework application. However, it has proved to be a very useful tool for most companies, which in fact are currently using it in many daily operations, namely (i) for testing new solutions, without overcrowding the database with irrelevant data, (ii) for quickly testing new scenarios, involving the definition of new rules or changing the values of some of the existing rules (this situation is very frequent when negotiations with the unions occur), and (iii) in formal presentations to managers or to the Administration Board, showing the impact of a new rule or the results of the new planning process (watching how a driver schedule is produced and manipulated, managers understand better the problem and the constraints involved).

Finally, the Duties Framework can also be used by small companies, as a simple but powerful software application for the driver scheduling process. Often, these companies are too small to invest in a system such as GIST.

In conclusion, the Duties Framework application has proved to be an interesting and powerful software tool both for mass transit companies and for R&D teams. In fact, it has strongly supported the development of the new approach presented in this thesis.

2.4.3 Other DSS systems for the operational planning process

The systems presented in this section are commercial software packages in operation in several important European cities. Most of them have started as the work of small R&D university teams and evolved to successful packages available from private companies. For most of them only commercial information is available, namely in their websites.

The HASTUS System

The HASTUS System is probably the most widely used DSS for operational planning. It is the result of over 25 years of ongoing research and development at GIRO (http://www.giro.ca/), in collaboration
with the Center for Research on Transportation of the University of Montreal [132], [20], [88]. Constant improvements is both optimisation methods and computer techniques are made and it is installed in more than 150 bus and rail companies, in more than 20 countries.

HASTUS includes an integrated database and several application modules for different phases of the OPP:

- **HASTUS-Vehicle** generates vehicle timetables and vehicle schedules and it includes a mouse-driven graphical scheduler.

- **HASTUS-Crew** produces legal and efficient crew schedules, using optimisation algorithms that can be fine-tuned interactively (the models and the solution methods of HASTUS System will be described in Chapter 3).

- **HASTUS-Roster** automatically generates multi-day crew assignments.

- **Geo Module** provides an integrated geographical information system and several other options for run time analysis, customer information and payroll calculations.

The HOT II System

The HOT System [90], [41], [148] was developed by HanseCom, an associate company of the Hamburger Hochbahn AG and it has been widely used in Germany since the 70's. A new version was developed at the middle of the 80's, the HOT II system, which is a modular and very interactive system. HOT II is composed by the following modules:

- **Basic Data** includes all the information about the network structure and the trips to be scheduled.

- **Sensitivity Analysis** enables the timetable specification to be examined before the vehicle schedule is produced. The goal is to reduce the number of vehicles required, by performing small changes in the departure times of the trips.

- **Vehicle Scheduling** produces the vehicle timetables, using optimisation algorithms that try to minimise the maximum number of vehicles to be used and also the non-productive time. Graphical user interfaces are provided for manual adjustment of the solutions.

- **Driver Scheduling** produces a complete driver schedule in an automatic multistage process with the aid of optimisation techniques. Manual adjustments can be carried out by the user. The HOT II approach for bus driver scheduling will be further detailed in Chapter 3.

- **Driver Rostering** automatically produces the driver rosters for each depot, although they can be adapted interactively to particular requirements.
2.4. DECISION SUPPORT SYSTEMS FOR THE OPERATIONAL PLANNING PROCESS

The BUSMAN II System

The BUSMAN II System [31] is an updated version of the BUSMAN System [155] and was developed in collaboration with the University of Leeds. Presently, BUSMAN II is being used by many operators, mainly in the UK. It integrates several components:

- ROUTEPLAN is a module for service and route planning analysis.
- BUSPLAN is a package for vehicle scheduling and timetables. It integrates heuristic-based algorithms, known as VAMPIRES and TASC [139]. Recently, an object oriented approach, known as BOOST [100], was developed and integrated in the BUSPLAN package.
- CREWPLAN is a package for driver scheduling which integrates interactive heuristics like the COMPACS system [31] and algorithms based on optimisation techniques, such the IMPACS System [138], [140], [157]. Simultaneously, the University of Leeds maintains an updated version of IMPACS, the TRACS II System. TRACS II (Techniques for Running Automatic Crew Schedules) [68], [102] is being developed at the University of Leeds since 1994 and it is being used on several bus and train companies, including First Group, the largest bus company in the UK. In the next chapter we will describe some of the approaches exploited in TRACS II.
- TRAVELGUIDE is a module for passenger information and publicity.

The MTRAM System

The MTRAM System (Mass Transit Management) is an Italian DSS for operational planning, distributed and marketed by MAIOR (Lucca) in collaboration with the University of Pisa. It is currently installed in several mass transit companies in Italy, namely in Florence, Bologna and Milan.

MTRAM includes a relational data base, sophisticated graphic interfaces and advanced models and algorithms for the optimisation processes.

It is composed of modules for information to passengers, vehicle scheduling and driver scheduling [27] as well as sensitivity analysis tools. These tools allow a quick evaluation of the costs induced by variations of the service level, an analysis and evaluation of the costs induced by locations of depots, their capacity and assigned routes, and an evaluation of the costs due to variations in union regulations.

The approach adopted by MTRAM for the bus driver scheduling problem will be presented in Chapter 3.

The MICROBUS System

The MICROBUS System [19] is a comprehensive DSS developed by IVU Berlin and is installed in more than 80 companies in Germany. It integrates a relational database and is composed of modules
for timetabling, vehicle scheduling, driver scheduling and rostering, along with modules for the evaluation of passenger demand surveys, the actual assignment of vehicles to vehicle schedules and passenger information.

IVU Berlin has also developed BERTA, a DSS for the operational planning specifically developed for the Berlin transport services (BVG).

The Crews System

The Crews System [119] is a crew scheduling system developed by SISCOG, which is currently in use in the Portuguese Railways, CP, and in the Dutch Railways, NS. This system heavily relies on the use of Artificial Intelligence (AI) techniques and has been built as an interactive system, in which the planner can propose alternatives, querying decisions and adapt the behavior of the system to changing circumstances. Scheduling can be done in automatic, semiautomatic, or manual mode. It includes advanced graphical user interfaces which are used in combination with the AI techniques to produce driver schedules, and to analyse and evaluate alternative solutions.

The TRAPEZE System

The TRAPEZE System was developed by UMA Engineering Ltd, in Canada. It includes an integrated database and provides help in timetables definition, vehicle scheduling, driver scheduling and driver rostering. Since UMA does not publish the approaches and techniques adopted in the TRAPEZE System, this system will not be further considered in this work.

The GoalBus and GoalDriver Systems

The GoalBus and the GoalDriver systems are developed and marketed by GOAL Systems, an associate company of Simrom Group in Spain. These systems are implemented in several Spanish mass transit companies. GoalBus is a computer system for vehicle and crew scheduling, while GoalDriver is a software tool for rostering.

There are no publications available on the methods used by the system, and only commercial information is available on these products. Hence, these systems will not be further considered in this work.

2.5 Summary

In this chapter we have described the operational planning process and discussed the particular case of Portuguese mass transit companies. Next, we have identified the most relevant information required for
the bus driver scheduling process. A summary of the most important duty type and overall rules was given and the manual scheduling process was explained.

Finally, we have presented some commercial decision support systems for operational planning. We have described in more detail the GIST System, since it will be used for testing the approach proposed in this thesis.
CHAPTER 2. THE BUS DRIVER SCHEDULING PROBLEM (BDSP)
Chapter 3

Approaches to the BDSP

3.1 Introduction

Driver scheduling has always received considerable attention from the Operational Research community [57]. Since the 60's, several models have been proposed and different solution techniques have been developed and experimented. Significant advances in computer technology allowed the resolution of larger and larger instances, in reasonable time. However, research in this field is far from being finished, and new approaches and methods are always under development.

In this chapter we present a comprehensive literature review about the approaches, models and techniques for the Bus Driver Scheduling Problem (BDSP). The chapter is divided into three major sections that, in a certain sense, correspond to a chronological sequence. In fact, the first attempts to automatically solve the BDSP were based on heuristic methods, that are presented in Section 3.2.

The development of computers allowed the application of Mathematical Programming methods, which are described in Section 3.3. We have divided the methods based on mathematical programming into two different classes. In Section 3.3.1 we present some methods based on formulations for the Set Covering Problem (SCP) or the Set Partitioning Problem (SPP), while in Section 3.3.2 we present some methods based on network models. Even if recently there has not been significant progress in terms of models, new solution methods are being developed and can now solve very large instances in reasonable time. Integrated approaches that simultaneously model and solve the vehicle and driver scheduling problems or the driver scheduling and rostering problems are currently being explored. Some of them are described in Section 3.3.3.

Finally, in Section 3.4 we describe a new class of approaches to the BDSP, based on metaheuristics. We discuss some works on Genetic Algorithm (GA) (Section 3.4.1), Evolutionary Strategies ES
(Section 3.4.2) and Tabu Search (TS) (Section 3.4.3). These approaches do not guarantee an optimal solution although they can quickly produce very good solutions.

References to real world applications of any of these methods are made in the text. Most commercial software systems appeared during the late 80's and the early 90's. At that time, their innovative solution techniques and their implementation results with real world instances have been extensively reported (see, for instance, the proceedings of the international workshop conferences in Computer Aided Scheduling of Public Transport [131], [42], [51], [40]). More recently, however, companies strongly publish the enhancements of their approaches but do not provide information on the real quality of the results. (see, for example, the last proceedings of the international workshop conferences [152], [149]). This is probably due to the increasing competition between the different systems in a area where investments may be extremely high.

3.2 Heuristic methods

The first computerised procedures for bus driver scheduling appeared in the 60's [57] and were based on heuristics. At that time, the existing computer technology was not able to solve mathematical programming models for real world problems and the procedures were not very interactive. Most of this early work was summarised by Wren and Rousseau [156]. Since then, many heuristic methods have been enhanced with interactive visual tools and integrated in decision support systems, such as RUCUS II [7], [108], HOT II [41], [148], COMPACS [31], OPTIBUS [30], PinGuim [143], and WinBus [38]. Even if most of these systems are no longer in use, they were successfully applied to mass transit companies, often yielding significative savings. With the help of interactive procedures and graphical user interfaces, they were capable of producing good solutions very quickly. However, they usually required a strong "customisation" and software changes whenever the operation conditions changed.

In this section we present a summary of the most significative heuristic approaches. Most of them are based on a rationalization of the manual procedures adopted by manual schedulers. The basic idea is to build an initial schedule following a sequence of rules and then attempting to improve it with small alterations.

The RUCUS II (RUn CUtting and Scheduling) system [7], [108], developed in the late 60's, was one of the first computerised systems for bus driver scheduling and it was in widespread use for several years in North America. The heuristic method builds an initial solution by forming the straight duties first and then the two-split duties. The uncovered pieces-of-work are considered as overtime work. Then the heuristic tries to improve the initial solution exchanging, for example, some pieces-of-work between duties. The final schedule may need some manual adjustments.
3.2. HEURISTIC METHODS

The HOT II (Hamburg Optimisation Techniques) system is a DSS for the whole operational transport planning process [90], [41], [148]. The driver scheduling module of this system is mainly based on heuristic procedures, although an optimisation technique is used in the process. The heuristic proceeds as follows: first, the early duties are built, using a selection procedure that rejects the duties that are not potentially good (e.g., too short). The remaining pieces-of-work are used to form partial duties. These partial duties are then placed into two sets. Set A contains early partial duties and set B contains the short partial duties. Then, an unbalanced assignment problem is formulated and solved using a variant of the Hungarian method [98]. Finally, the late duties are formed for each bus using a similar process but in a reverse way. The optimisation process also allows breaking and re-assigning less efficient duties previously formed.

TRACS (Techniques for Running Automatic Crew Scheduling) has been under continuous development at the University of Leeds since 1967 [126]. The system starts to build a good initial schedule that can be further altered by heuristic improvements. This initial schedule is constructed in the following steps:

1. Build early straight duties for each vehicle, leaving sufficient work for the first halves of split duties to cover the morning peak.

2. Build late straight duties and late split duties for each vehicle, leaving work enough for the second halves of split duties to cover the afternoon peak.

3. Build split duties by matching the first and second halves.

4. When possible, attach the remaining work to the existing duties.

The initial schedule is improved by two sets of procedures. The first set attempts to reduce the number of duties and the number of uncovered pieces-of-work. Each duty is inspected to determine whether the work in it can be contained in other duties. The other set of procedures tries to reduce the cost of the solution. This could be achieved by several different procedures, namely by swapping or moving stretches of duties, re-matching first and second stretches of duties, switching changeovers of a stretch to another relief time, and relocating small pieces-of-work around the middle of the day.

The COMPACS system (COMPuter Assisted Crew Scheduling) was an interactive system developed in the early 1980's [159], which was later incorporated into the BUSMAN package [31]. It combines the heuristics of TRACS and the interactive features of an early interactive system known as TRICS. One useful feature of COMPACS is that it can give an estimate on the possible number of duties of each type. The estimate is useful since it can guide the schedulers to build up the schedule interactively.
CHAPTER 3. APPROACHES TO THE BDSP

The PinGuim system [143], [141], was the forerunner of the present GIST System and it included an interactive heuristic procedure for the driver scheduling in STCP, the mass transit company operating in Porto. This heuristic was derived from the one that was implemented in TRACS by Parker and Smith [126], but it included enhanced interactive features. The user can choose between a fully automatic procedure or an interactive one. The fully automatic procedure is structured in two stages: first, it builds the early and late straight duties for each bus, and second, it tries to bring together the remaining pieces-of-work to form good duties. These duties can be either straight or split duties. The interactive procedure lets the user choose which buses are going to be used to build the early and late straight duties, in order to allow more opportunities to form split duties. This heuristic was particularly well adjusted to STCP, since this company tried to build as many straight duties as possible. Unfortunately, this heuristic was not suited to the other Portuguese mass transit companies as they imposed upper limits to the number of straight duties. Some years later, Couto [38] has developed an enhanced version of this heuristic, which included a wider variety of user options and also some fuzzy parameters that allowed the schedule to slightly violate some of the rules. This new heuristic has proved to adequately tackle a larger number of situations.

The use of pure heuristic methods has been progressively replaced by mathematical programming methods aided by heuristic procedures. In some particular situations, the application of heuristic methods can still be appropriate, for example when providing an upper bound to more sophisticated techniques.

3.3 Mathematical programming methods

The BDSP can be formulated as an integer linear programming model, even if some complex constraints may render models quite difficult to tackle. In this section we explore the most common formulations used to model the BDSP and the usual resolution methods applied. The set covering / partitioning models are the most common and successful formulations for the BDSP. Some of the approaches based on these models are presented in Section 3.3.1. Some other approaches based on network flow models are described in Section 3.3.2. Network flow formulations have been recently used to model an integrated approach for vehicle and driver scheduling. These integrated approaches are presented in Section 3.3.3. In this section we also describe some of the approaches that have been applied to the Crew Planning Problem, that is closely related to the BDSP.

The BDSP is usually referred to in the literature as a particular case of the crew scheduling problem (CSP), which also includes train crew scheduling (TCS) and airline crew scheduling (ACS). Since many of the approaches described are common to all these problems, some references are made to these specific
situations.

According to Wren and Rousseau [156], the BDSP can be expressed as follows:

Given a set of \( m \) pieces of bus work each requiring a driver, together with a set of \( n \) possible driver duties, it is required to find a subset of the \( n \) duties which together cover all the pieces-of-work and which minimises either the total cost of duties chosen or the number of possible duties (or some combination thereof).

### 3.3.1 Methods based on set covering and set partitioning formulations

The BDSP (3.1) can be formulated as a general integer linear programming (ILP) model as follows:

\[
\text{minimise } f(x_j) = \sum_{j=1}^{n} c_j x_j \\
\text{subject to } \sum_{j=1}^{n} a_{ij} x_j \geq 1 , i = 1, \ldots, m \\
x_j \in \{0, 1\} , j = 1, \ldots, n \\
x_j \in X , j = 1, \ldots, n
\]

where \( n \) is the number of previously generated duties, \( m \) is the number of pieces-of-work and

\[
x_j = \begin{cases} 
1 & \text{if duty } j \text{ is in the solution} \\
0 & \text{otherwise}
\end{cases}
\]

\[
a_{ij} = \begin{cases} 
1 & \text{if duty } j \text{ covers piece-of-work } i \\
0 & \text{otherwise}
\end{cases}
\]

\( c_j \) is the cost associated to duty \( j \) and \( X \) is a set of side (global) constraints.

The side constraints in \( X \) correspond to features that affect the whole schedule and the number and complexity of these constraints depend on each particular company. Without these constraints, this model would become the well-known Set Covering Problem (SCP). Here we give some examples of the most common global constraints [118]:

- **Minimum and maximum number of duties of each type**

Let there be \( p \) different duty types and \( D_1, D_2, \ldots, D_p \) are the sets of indices corresponding to these duty types:

\[
D_1 = \{1, 2, \ldots, n_1\}, \\
D_2 = \{n_1 + 1, \ldots, n_2\}, \ldots, \\
D_p = \{n_{p-1} + 1, \ldots, n_p\}, \text{where } n_1 < n_2 < \ldots < n_p = n.
\]
Let $L_k$ and $U_k$ be the minimum and the maximum number of duties of type $k$ respectively. Hence these constraints can be expressed as follows:

$$L_k \leq \sum_{j \in D_k} x_j \leq U_k, \ k = 1, 2, \ldots p$$

(3.2)

- **Average time constraint for each duty type**

Let $t_j$ be the time worked by duty $j$ and $av_k$ be the limit on the average time worked by duty type $k$. This constraint can be expressed as follows:

$$\sum_{j \in D_k} \frac{t_j x_j}{x_j} \leq av_k \Leftrightarrow \sum_{j \in D_k} (t_j - av_k) x_j \leq 0, \ k = 1, 2, \ldots p$$

(3.3)

- **Maximum total number of duties**

Let $N$ be the maximum number of duties allowed in the solution. This constraint can be expressed as follows:

$$\sum_{j=1}^{n} x_j \leq N$$

(3.4)

Other constraints can be expressed in a similar way, namely those that impose a limit to the extra work assigned to each duty, or an average number of changeovers for each duty type.

Problem (3.1) is a rather difficult one, because of the large number of integer variables and constraints, even for small size transportation companies. Some authors, namely [118] and [29] have proposed to solve it by integer linear programming methods. However, the most common approaches are based on a simplification of the model. The solutions obtained are obviously suboptimal. The most common technique corresponds to removing side constraints (3.1d) and thus transforming (3.1) into a SCP, which can be formulated as follows:

$$\begin{align*}
\text{minimise} \quad f(x_j) & = \sum_{j=1}^{n} c_j x_j \\
\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j & \geq 1 \quad i = 1, \ldots, m \\
& x_j \in \{0, 1\} \quad j = 1, \ldots, n
\end{align*}$$

(3.5a, 3.5b, 3.5c)

The SPP is a particular case of the SCP (3.5) in which constraints 3.5b are equality constraints. Both set covering and set partitioning formulations have been widely used to model the BDSP, although the solution methods differ.
Set covering and set partitioning problems have been largely studied and both belong to the class of NP-hard problems [74]. For this class of problems no algorithm is known that can find the optimal solution in polynomial time and hence, the time needed to find the optimal solution increases exponentially with the size of the input data.

Traditionally, SCP/SPP models are solved by first generating a large number of feasible duties and by applying then a mathematical programming method that selects a set of these duties (columns) that cover all rows with minimum cost.

Formulated as an ILP model, the problem has been naturally tackled quite often by implicit enumeration or branch-and-bound approaches. In these methods partial solutions are generated by taking the columns one at a time and exploring logical implications of their assignment. Garfinkel and Nemhauser [75] and Marsten [112] have developed implicit enumeration methods for the SPP. Marsten and Shepardson [113] concluded that, for small instances of the SPP, the continuous primal solution is often integer. Beasley [10] and some years later, Beasley and Jörnsten [18] have exactly solved the SCP for problems with hundreds of rows and several thousands of columns. In practice, however, there can be billions of such feasible duties. Paixão [122] and Ryan and Falkner [134] have explored the polyhedral structure of the problem and designed preprocessing routines that can reduce the number of rows and columns of the problem. Hoffman and Padberg [89] proposed an exact method based on the use of branch-and-cut and reported optimal solutions for a large set of real-world SPP problems.

Graph theory has provided some interesting tools to improve the techniques for the SCP/SPP and a comprehensive discussion on these issues is presented by Borndorfer [22]. However, in most cases these preprocessing routines are not able to reduce a very large problem to a manageable size. Computational efficiency of exact methods rapidly decays with the problem size and, in fact, they are rarely used in real-world applications for the BDSP. Most common approaches to the SCP/SPP are based on approximate algorithms using mathematical programming techniques just to provide lower bounds to the optimal solution. On the other hand, greedy heuristics techniques are often used to obtain upper bounds and integer solutions that expectably are as close as possible to the lower bounds.

In Figure 3-1 we present a general procedure to solve a SCP/SPP. This procedure is structured in four major steps and, for each step, several different methods have been proposed. In this work we will present some of the most representative methods, focusing on real world applications.

In the first step, a large number of feasible duties is generated. In fact, theoretically, to achieve optimality, SCP/SPP approaches require that all feasible duties are generated. In order to guarantee the existence of a feasible solution for the SPP, the set of pieces-of-work are also included as feasible duties (known as tripsters).
However, even for small instances, the total number of feasible duties may be extremely large (obviously growing in an exponential way), and it is therefore only possible to consider a (small) subset of these duties. One possible technique is to eliminate potential duties with particular long breaks, with breaks at unsuitable times of the day, or with little work content before or after a meal break [126]. Other heuristic procedures select potentially interesting duties and eliminate less efficient ones [157], [61].

Instead of trying to reduce the number of columns (duties), one can also try to reduce the number of rows (pieces-of-work). Examining all pieces-of-work, we can determine whether two consecutive ones can be combined together. Reducing the number of relief opportunities leads naturally to the generation of a lower number of feasible duties. Such reduction techniques can be found in several commercial systems, such as HASTUS [132], [20], TRACS II [68], [102] or GIST [39], [142]. Naturally, they increase the algorithm efficiency but decrease the probability of finding better solutions. Sometimes, the existence of some worse duties will allow an improvement of the overall solution. Therefore, special care in the design of these heuristics must be taken in order to avoid discarding interesting possibilities.

The set of generated feasible duties is used in the second step as input to a relaxed SCP/SPP, which is solved using some optimisation technique. Currently, the approaches based on SCP/SPP models can be divided into two main groups. In the first group (option 2.a in Figure 3-1), the strategy is to solve a single relaxed problem. This strategy is also called a generate-and-select approach. The second group (option 2.b) uses a column generation approach. In fact, column generation methods are becoming more and more popular and methods of the first group are only being used for small / medium size problems, for which it is possible to handle the entire set of generated feasible duties. Even with heuristic reduction procedures, this set is often too large to be tackled by standard mathematical programming software. Column generation methods allow the creation of new columns as required, so as to implicitly consider all feasible duties without the need to previously generate them.

**Generate-and-select approaches**

In many of these approaches a linear relaxation of the problem is solved (relaxing the integrality constraint 3.5c), thus providing a lower bound to the optimal solution of the original problem. An initial feasible solution can be obtained using greedy constructive heuristics, trying to obtain good upper bounds for the value of the optimal solution [122], [10]. The linear relaxation can be solved either by primal or dual simplex approaches, the dual approaches being in general much more efficient, given the degeneracy of solutions. Then, a branch-and-bound algorithm is used to get an integer solution. The gap between the lower and the upper bounds can be tightened by the application of primal and dual heuristics.

TRACS II (based on IMPACS) formulates the problem as a set covering problem with side constraints. Slack and surplus variables are used to transform the formulation into one analogous to that of Mitra
and Darby-Dowman [117] in their CRU-SCHED system. The TRACS II approach is interesting because it addresses the multiobjective nature of the BDSP [68]. It considers different and conflicting objectives to which different degrees of importance are assigned. These objectives include (by decreasing order of importance) the minimisation of the number of uncovered pieces-of-work (leftovers), the number of duties in the solution, the number of duties with undesirable features, the wage costs and the total duration of overcovered pieces-of-work. This set of objectives is condensed in two main objectives that are solved using a sequential approach. A simplified model, which is currently integrated in TRACS II, was proposed by Willers [151]. This model combines the objectives of minimising the number of duties and the total duty costs, while retaining their preference ordering, using Sherali [137] approach to transform multi-objective models into single-objective models. The resulting model is then solved using a branch-and-bound procedure leading to solutions that can be further improved by heuristics. In
TRACS II the branch-and-bound method was developed with an emphasis on finding a good integer solution quickly. A branch-and-bound tree is developed and the lower bound on the objective cost is given by the optimal continuous solution. Although TRACS II is in general quite successful, there are still computational difficulties in its branch-and-bound algorithm [69], leading the University of Leeds R&D team to develop new approaches to the BDSP, as we will see later, in Sections 3.4.1 and 3.4.3.

Falkner and Ryan [61] developed the EXPRESS system specially for Christchurch Transport in New Zealand. It is based on a set partitioning model. During the search process, the strictness of the model is diminished by the addition of slack variables. It uses heuristics to reduce the size of the problem and a 3-stage approach is used to solve the problem. In the first stage, a mathematical programming technique is applied to select middle and late duties. The pieces-of-work covered by these duties are removed from the problem. In the second stage, the early duties are selected also using a mathematical programming model and the pieces-of-work covered by these early duties are removed. Finally, in the third stage, the half duties and the extra early duties are chosen to cover the remaining pieces-of-work, by a set partitioning model. An assignment algorithm is then used to combine the half duties. EXPRESS uses a version of the original Zero-One Integer Programming package (ZIP) similar to those being used in IMPACS and TRACS II.

The HASTUS [132] system includes a module for driver scheduling that uses a slightly different approach. This module is divided into two parts, HASTUS-micro and HASTUS-macro. HASTUS-macro constructs an initial schedule and HASTUS-micro produces the final schedule. HASTUS-macro uses linear programming to generate a initial schedule that provides an estimate of the number of drivers needed. The initial schedule is built by idealised duties, which are generated using simplified relief opportunities by simply cutting the day into pieces specified by the user. Then HASTUS-micro uses this initial schedule to create a final schedule, by producing real duties as close as possible to the idealised ones. This is accomplished by using a network flow formulation to find the desired number of pieces-of-work and then by using a matching algorithm to build the duties. Finally, several heuristics are applied to improve the solution. This method has proved to be very effective and was implemented in more than sixty cities around the world. Since 1990, HASTUS integrates a component, Crew-Opt, that uses a column generation approach. The Crew-Opt approach will be described later in this chapter.

Paixão [123] formulated the SCP as a dynamic programming problem and he solved it using state space relaxation (SSR). Paixas and Paixão [121] showed that SSR efficiently provides a good lower bound for the original SCP. They were able to build feasible solutions upon the SSR solution with an acceptable gap with the lower bound.
A different class of approaches consists in using lagrangean relaxation instead of linear programming to solve the SCP. These approaches were initially proposed by Balas and Ho [6], and were further explored by [11], [150] and [5].

In fact, Caprara et al., [24], [26] claim that the most effective heuristic approaches for the SCP are those based on lagrangean relaxation. The authors report an interesting competition organised by an Italian railway company and the Italian Operational Research Society, intended to promote the development of algorithms for large instances of the SCP. Their algorithm, that is also used by the CARMEN system for airline crew scheduling, won the competition. They adopted a lagrangean relaxation approach that is structured in three stages. In the first stage, one aims at quickly finding a near-optimal lagrangean multiplier vector. The second stage starts from the best lagrangean vector found in the previous stage and uses a greedy heuristic to improve it. Finally, in the third stage a subset of duties with a high probability of being in the optimal solution is selected and the corresponding variables are set to 1. Hence, a smaller instance of the SCP is obtained and after a refining procedure, the three-stage process restarts. For very large instances, the authors work on a small subset of columns, defining the core problem, which is dynamically updated using the dual information associated to the current lagrangean multiplier vector. This strategy is similar to column generation applied to lagrangean relaxation.

Kroon and Fischetti [97] also report a similar approach that have been successfully applied to a Dutch railway operator.

Column generation approaches

The major disadvantage of the generate-and-select approaches is that the search for the optimal solution is limited to a set of previously generated duties that, most of the times, has been heuristically reduced. Column generation is an approach used to solve mathematical programming problems involving a large number of variables (columns) without needing to explicitly consider them all at the same time. The method solves a linear relaxation of the problem over a subset of variables. Then, the process uses the dual variables to generate new duties with negative reduced costs which will potentially improve the current solution. These duties are added to the subset and a new problem is solved. This process is repeated until no feasible duty with negative reduced cost can be found. Then, a branch and bound procedure is started in order to obtain an integer solution.

Several column generation approaches for the BDSP have been proposed regarding the following aspects:

1. how to choose the initial set of feasible duties;

2. how to generate new columns (duties);
3. how to obtain a final integer solution.

HASTUS contains a module, Crew-Opt [52], [50], that applies column generation with encouraging results when compared with the HASTUS-macro and the HASTUS-micro approaches [133]. Crew-Opt generates an initial set of duties, which is composed by feasible duties and by trippers. Each tripper corresponds to a short (and probably unfeasible) duty corresponding to a single piece-of-work. An initial solution is constructed with a set of trippers that cover all the pieces-of-work. This initial solution is likely to be very expensive since each tripper is assigned a very high cost. However, these duties will probably be removed from the solution as new columns are added. The linear programming relaxation of the SPP (known as the master problem) is solved for the set of initial duties and a procedure to find new feasible duties is then initiated.

The problem of generating new feasible duties with negative reduced cost is known as the subproblem, and Crew-Opt formulates it as a constrained shortest path problem. The nodes on this time-space oriented network correspond to relief opportunities and the arcs correspond to pieces-of-work and breaks. The duties correspond to paths through this network and resource constraints are defined to ensure that only feasible duties will be considered. Costs are assigned to the arcs on the network, so that the shortest path corresponds to the duty with the most negative reduced cost. If the cost of this duty is positive, then the current solution is optimal, otherwise, the duty must be added to the subset and the new problem re-optimised. The shortest path problem is solved using a dynamic programming algorithm and, instead of finding a single path, the algorithm builds a shortest path tree, representing several duties. All these duties are added to the subset in order to accelerate the convergence.

Desaulniers and Desrosiers [49] have presented a different network structure, namely a compatibility oriented network, in which the nodes correspond to the tasks (pieces-of-work) while the arcs connect compatible tasks. This structure has the advantage that the feasibility constraints concerning the individual pieces-of-work are satisfied by default and do not need to be checked during column generation. However, the authors compared a time-space oriented network with a compatibility oriented network for a particular airline crew scheduling problem and concluded that the first one was the most appropriate for that specific context.

Once the optimal solution for the linear relaxation is found, a specialised branch-and-bound procedure is initiated. At each node of the branch-and-bound tree, column generation is also used to solve the linear programming relaxation, but the network is restricted to represent only the subset of duties relevant to that current node.

Fores [69], [68] proposed another implementation of the column generation approach, which is currently included in the TRACS II software. In this approach, the whole set of feasible duties is generated.
initially and it is therefore available to the column generation process. This strategy has the disadvantage of needing to maintain this large set at the main memory, which is computationally expensive, but the main reason to adopt it was that the duty costs include penalty factors that are complicated to model using a shortest path technique. Based on this large set of duties, a subset sufficiently large and varied is selected and an initial solution is generated. The relaxed linear programming problem is solved over the current subset of duties. The reduced costs of the duties in the large set of duties that are not currently selected are computed. The duties with favorable reduced costs are added to the subset and the linear programming problem is re-optimised. If no duties with favorable reduced costs can be found, then the current solution is optimal. A branch-and-bound procedure similar to the one used in the generate-and-select approach is applied. Fores states that this approach has allowed TRACS II to tackle considerably larger problems than previously.

Recently, new approaches, based on constraint programming, have been proposed to solve the sub-problem (in column generation approaches). Constraint programming techniques are adequate to highly constrained problems, since they naturally provide feasible solutions and complex constraints can be easily expressed. Pure constraint programming techniques have been tried [161], [86], but they faced difficulties when handling large scale instances. Hence, hybrid approaches that combine column generation with constraint programming have been tested with success [160]. The basic idea of these hybrid approaches can be described as follows: the linear programming relaxation is solved for the master problem. The values of the dual variables are then used by the constraint programming algorithm in order to generate duties with negative reduced costs. These duties are added to the subset, the master problem is solved again and the process continues until no further duties can be generated by the constraint solver. The feasibility constraints are expressed in a declarative way and in order to ensure that the generated duties have negative reduced costs, it is only necessary to add another constraint. This constraint programming procedure replaces the shortest path algorithm used by other column generation approaches.

3.3.2 Methods based on network models

There are several different network based approaches for the BDSP and most of them can be easily transformed into set covering / partitioning models. For most of these approaches, the problem can be formulated as the problem of covering the vertices of an acyclic graph by a number of minimum cost node disjoint paths that satisfy a set of constraints [116]. Here we describe the network model proposed by Beasley and Cao [14].

Consider a set of \( N \) tasks (the pieces-of-work) which correspond to the nodes in the network. Each
task \( i \) has a cost \( d_i \), a fixed starting time \( s_i \), together with a fixed finish time \( f_i(> s_i) \). For any two tasks \( i \) and \( j(j > i) \) there is a transition arc of cost \( c_{ij} \) if task \( j \) can be performed after task \( i \) by the same crew. A path is the set of tasks that will be performed by the same crew. The problem therefore is to find \( K \) paths of minimum total cost such that each task is performed exactly once and the total working time involved in each path does not exceed the maximum working time \( T \). The problem can be formulated as follows:

\[
\begin{align*}
\text{minimise} & \quad \sum_{i,j} c_{ij}x_{ij} \\
\text{subject to} & \quad \sum_{k} x_{jk} = \sum_{i} x_{ij} \quad j = 1, \ldots, N \\
& \quad \sum_{j} x_{ij} = 1 \quad i = 1, \ldots, N \\
& \quad \sum_{j} x_{0j} = K \\
& \quad \text{time limit constraints} \\
& \quad x_{ij} \in \{0, 1\} \quad \forall i, j
\end{align*}
\]

where \( x_{ij} = \begin{cases} 
1 & \text{if transition arc } (i, j) \text{ is used in the solution} \\
0 & \text{otherwise}
\end{cases} \)

Constraints (3.6b) impose that the number of arcs leaving each task \( j \) is equal to the number of arcs entering the task. Constraints (3.6c) impose that each task is on a single path. Constraint (3.6d) states that \( K \) tasks are used to be the first task of some path, which implies that we will need \( K \) crews. Constraints (3.6e) represent the requirement that the working time in each path does not exceed \( T \) and, finally, constraints (3.6f) specify that each arc can be used at most once. Although the authors state that this is a formulation of the generic crew scheduling problem, it is a rather simplified model, since it does not include any side constraints and fixes the number of drivers needed \( (K) \), which in most cases is unknown.

The authors used lagrangean relaxation to obtain a lower bound, applying the subgradient optimisation method. Then, they have designed a particular tree search algorithm for the branch and bound stage. Results have been given for random generated test problems. In 1998, the same authors [15] have used a similar approach, but they have applied a dynamic programing algorithm for the initial lower bound.

Mingoazzi et al [116] used a similar network based formulation and also a set partitioning formulation with a side constraint. They created relaxed problems of the proposed network model whose feasible
solutions are also feasible to the original problem. The optimal solutions of these relaxed problems
are then viewed as heuristic solutions to the original problem. They also propose a heuristic procedure
based on the set partitioning formulation. This procedure uses the dual of the linear relaxation of the set
partitioning model to find a lower bound for the initial problem. This lower bound is then used to reduce
the number of variables of the set partitioning problem, which can then be solved by branch-and-bound.
The authors refer that the model can deal with additional side constraints and results on the same data
sets of Beasley and Cao [14] are reported.

Carraresi et al [29],[28], [27] propose a general network model for the BDSP that is embedded in
the MTRAM system. This model can also be applied to a larger class of problems, namely those
with more than one crew domicile, such as airline crew scheduling problems. They propose an algorithm
belonging to the class of subgradient bundle methods to solve the lagrangean dual of the proposed model.
They have implemented a column generation approach and a row deletion heuristic procedure based on
constrained shortest path algorithms and k-shortest paths enumeration. Results for real instances of
driver scheduling and airline crew scheduling problems are given.

Banihashemi and Haghani [8] propose a new model for the bus driver scheduling inspired by the
multi-commodity formulation of the multi-depot vehicle scheduling problem (MDVS). This model is
more complete than those described above, since it considers constraints involving different duty types.
They solve a relaxed problem and build an iterative procedure in which they add variables and constraints
to the relaxed model. However, this model is still exploratory and computational results have not been
provided.

Network based approaches do not need to explicitly generate the set of all feasible duties, since each
duty corresponds to a path in the network. However, when the feasibility constraints are overwhelming,
such as when the duties must satisfy a large set of complex union agreements, set covering / partitioning
models are preferred, since it is often impossible to incorporate some of those constraints in the network
structure.

3.3.3 Integrated approaches

As we have referred to in Section 2.2, there are strong interactions between the different phases of the
operational planning process. Probably, the planning process could be more efficient if an integrated
approach was adopted.

Some years ago, integrated approaches were computationally intractable and thus each stage is usually
handled separately and sequentially. However, the increasing development of computer technology has
allowed to handle real size problems and, currently, integrated approaches are an active and promising area of research. Recently, some integrated approaches have been proposed, basically in two situations: for vehicle and driver scheduling, and for driver scheduling and rostering. The first situation mainly occurs in mass transit companies where vehicle schedules impose certain constraints on the driver schedules. The second situation is very frequent in railways and airline companies and is also known as the Crew Planning Problem (CPP).

Vehicle and driver scheduling

The vehicle and driver scheduling problem (VDSP) can be defined as follows [70]: given a set of service requirements or trips within a fixed planning horizon, find minimum cost schedules for the vehicles and the drivers, such that both the vehicle and the driver schedules are feasible and mutually compatible.

Few proposals have been made for the solution of the VCSP (for an overview, see the work of Freling et al [70]). Since in most cases, the costs related to drivers are much larger than those related to vehicles, it seems reasonable to tackle the joint problem, particularly when an efficient vehicle schedule may lead to a poor driver schedule.

Freling et al [70] also describe some other situations when an integrated approach could be beneficial, namely a restricted number of changeovers (number of vehicle changes for each driver), minimum duration of a piece-of-work, or when a driver can only be relieved at the depot. These authors propose a formulation for the VCSP that is a combination of the quasi-assignment formulation for the single depot vehicle scheduling and a set partitioning formulation for driver scheduling. They adopt a lagrangian relaxation to compute a lower bound and, at each iteration, a feasible vehicle schedule is obtained and used to get feasible final vehicle and driver schedules. This approach was tested with two real problems from a public transport company in the Netherlands [71]. Results were compared with a sequential approach and have proved that vehicle and / or drivers can be saved by using the integrated approach.

Friberg and Haase [72] propose an integrated approach for the VCSP with a single depot. The problem structure is exploited through two different graphs, on which the vehicle schedules and the duties are defined as paths. Linear relaxation and a column generation approach are applied. In order to increase the values of the lower bounds, column generation was combined with polyhedral cuts such as those suggested by Hoffman and Padberg [89]. However, this strategy has exponentially increased the computing time.

Galfi and Nonato [73] have proposed an integrated model for the extra-urban case with several depots, which present some particular and complex constraints linking vehicles and crews. They have adopted
a heuristic based on lagrangean relaxation and column generation. Computational tests were carried out with data sets from several Italian municipal mass transit companies. The results obtained with the integrated approach have outperformed those obtained with the sequential approach of the MTRAM system.

Patrikalakis and Xerocostas [127] have proposed a different approach for the scheduling problem in urban transport companies. They suggest that duties should be created first and then vehicle schedules built. First, driver schedules are created using a set covering formulation. Then, the vehicle scheduling problem is formulated as a network flow problem on a graph formed by the duties previously chosen. The driver scheduling problem is solved again in order to obtain a driver schedule compatible with the vehicle schedule. They have concluded that the proposed approach can be more efficient than the sequential one when vehicle dependent constraints, such as a maximum number of changeovers, are not very important. Results are presented for small problems.

The Crew Planning Problem

The Crew Planning Problem (CPP) is a common problem that arises in large transit systems, such as railway and airline companies. It involves an integrated view of crew scheduling and rostering. Due to the size of the instances and to the type of rules and constraints, it is usually decomposed into two separate phases that are solved sequentially.

The Crew Planning Problem can be stated as follows: Given a planned timetable and a set of depots (home bases) where the crews are located, assign work schedules (or crew rosters) to crews located at each depot for a given planning horizon, such that each trip (or flight) of the timetable is served once, the operational constraints are satisfied and the total costs are minimised [114], [26].

In this context, a duty is a sequence of trips (or flights) without rests that must satisfy a set of rules. These rules can be very different from those of the usual bus driver scheduling problem. Then, daily duties are used to build pairings. A pairing (or round-trip, in railways terminology) is a sequence of duties for several days (1 or 2 for railways and over up to 5 for airlines) that must start and end at the same home base. Pairing generation is constrained to a set of rules (for example, a minimum for the rest time between two consecutive duties). Finally, the work schedule (or crew roster) for each crew member is defined as a sequence of pairings for the given planning horizon. The weekly rests for a crew member have to be considered, and all pairings of the crew roster must start and end at the same depot.

In the airline industry crew planning has attracted considerable attention, since the 60's (see, for example, [4], [60]). Air crews are amongst the most valuable of airline resources and the efficient utilization of crews is obviously an important issue in airline operations.
In railways, however, crew planning has received much less attention until recently. But in the last few years, a large variety of approaches to the crew planning problem applied to railways and several software systems have been implemented. This can be explained by two factors:

- Since the early 90's, the European rail industry has been under a re-organization process, imposed by European Union directives concerning the separation of the railway system into separate companies: those that control the infrastructure and those that control the train operations. While the former could stay under government control, train operations should be divided into several independent companies. This new scenario has increased the competitiveness in rail services and has introduced a new management style in order to improve their business results. In many European countries, this process has already happened, namely in Sweden, in the United Kingdom or in Italy. Hence this sector is showing increasing interest in producing better quality train driver schedules. Presently, several train driver scheduling applications have been implemented in many European railway companies [25], [102].

- Both hardware and software capabilities have hugely evolved since the first computational experiments. Problems that were intractable due to their size and complexity can now be solved in reasonable time. Moreover, some new approaches, such as column generation, can be applied to reduce the size of the problems (or subproblems) to be solved.

The CPP is considered to be one of the hardest practical problems in the domain, due to its high combinatorial complexity. In fact, constraints are very complex and can impose limits to the daily, weekly and monthly working times, length of rest periods or number of night duties. Usually it is decomposed into two different phases, which are solved sequentially:

1. Crew Pairing: build a set of legal pairings for crews, such that each trip is covered exactly by one pairing. The crew pairing process (also known as crew scheduling) is very similar to the bus driver scheduling, except for the set of rules that govern the construction of legal duties and pairings, and thus similar approaches are used to solve both problems.

2. Crew Rostering: construct crew work schedules, by chaining the pairings into crew rosters for a given planning period. The Crew Rostering problem consists of finding a feasible set of crew rosters covering all the pairings with a minimum cost. Such costs may reflect the workload balance or the global number of crews needed to perform all the pairings for the planning horizon.

This decomposition is also motivated by operational reasons [26]. Each crew is located at a given depot, that corresponds to the starting and ending point of its work segments. Each crew must return to its home depot in few days, leading to the concept of pairing as the short-term working unit.
Moreover, the set of rules and operational constraints that affect pairings are different from those related to the overall crew rosters. Finally, decomposition is the approach adopted by most airline and railway companies.

Caprara et al. [25] have suggested a sequential approach based on lagrangean relaxation and several heuristics, and this approach has been applied to the Italian Railways. In 2001, Caprara, Monaci and Toth [26] have proposed a modification to that approach that integrates the Crew Pairing and Crew Rostering phases. The authors keep both the Crew Pairing phase and the Crew Rostering phase, with the difference that the selection of the pairings in the former is driven by the objective function of the latter. Using an iterative procedure, each time a new candidate set of pairings is found by the Crew Pairing phase, the Crew Rostering phase is called to check whether this pairing set leads to a set of rosters better than the current one. The integrated approach has achieved considerable improvements when compared with the previous sequential approach.

Mellouli [114] proposed a network flow approach to the CPP. The author has integrated this approach in a DSS that was adopted by an European airline company. The approach is based on an analogy between the CPP and the Vehicle Scheduling Problem with maintenance routing. While vehicles in this problem are routed as to pass through a maintenance base every 3-4 days for inspection, crews in the CPP have to pass through their home bases every at most five days, in order to accomplish a weekly 2-days rest. The network model was designed to take into account multiple depots, crew requests, as well as time-dependent and day-dependent crew availability at different depots. Results for a small set of real world problems have been published.

3.4 Metaheuristics

In recent years, much attention has been devoted to general heuristics for solving Combinatorial Optimisation Problems (COP). Among these techniques are the so-called metaheuristics, including Genetic Algorithms (GA), Evolutionary Strategies (ES), Tabu Search (TS) and Simulated Annealing (SA). Many metaheuristics are inspired on optimisation procedures occurring in natural systems and can be considered as local search strategies, meaning that local search is performed moving from a solution to another one in the neighborhood, according to some precise rules. The application of a metaheuristic strategy does not guarantee optimality but hopefully a satisfactory solution will be reached within a reasonable computation time.

The increasing interest in the application of metaheuristics to Combinatorial Optimisation is motivated by the difficulties that often arise when traditional mathematical programming methods are
applied. First, most mathematical programming methods, such as those described in Section 3.3 are, in fact, based on heuristic procedures, in the sense that the optimal solution is no longer attainable due to the relaxations that were introduced in the model. Second, some real problems include complex constraints and objective functions that cannot be effectively tackled by traditional approaches.

These considerations suggest that metaheuristics can become an alternative approach to complex problems and in this section we discuss the application of some of these approaches to crew scheduling and to the related problems, the SCP and the SPP. Although metaheuristics are not attached to any particular formulation we give here the underlying model which helps to describe the constraints and objective functions considered. The most performant GA for these problems are presented in Section 3.4.1. Some interesting ES and TS approaches are described in Sections 3.4.2 and 3.4.3, respectively.

3.4.1 Genetic Algorithms

Genetic Algorithms (GA), [80], [45] are local search heuristics that mimic the evolutionary process of living organisms. GAs work on a set of solutions and combine these solutions together in some way, to generate new solutions. The underlying theory and the main components of a GA will be described in more detail in Chapter 4.

GAs have been applied to crew scheduling problems since the middle 90's. These are in general based on SCP and SPP formulations with applications to airline crew scheduling problems. Only a few are particularly designed for the BDSP. Most of the approaches combine genetic algorithms with other heuristics or metaheuristics procedures, and incorporate specific problem knowledge in the search process. These GA are often called hybrid genetic algorithms or population heuristics [13]. Although these approaches have significative differences relatively to the standard GA, we will refer to all of them as GA. In this section we focus on the most significative GA approaches to the BDSP.

Several motivations exist for applying GA to the BDSP. First, a GA works directly with representations of solutions, which means that there is no need to solve LP relaxations. Second, GA are very flexible in handling complex constraints such as the average working time or limits imposed to duty types. Traditional methods have difficulties in accommodating the addition of new constraints. Moreover, an important feature of a GA is that at each generation, a set of possible solutions is provided. As noted by Arabeyre et al [4], "the knowledge of a family of good solutions is far more important than obtaining an isolated optimum".

Nevertheless, the BDSP is a very difficult problem for a standard GA, since it is highly constrained and has a multimodal search space. Hence, most of the GA approaches to the BDSP incorporate some specific problem knowledge in order to obtain feasible solutions. In fact, one of the main differences between such approaches are the specific strategies that were designed in order to maintain or restore
the feasibility of the solutions.

For many of the GA approaches to the BDSP, the set of feasible duties is previously generated and the GA is applied in order to select a subset of duties that covers the vehicle's work with the minimum cost. In fact, in this domain, GA are usually used as a generate-and-select approach (see Section 3.3).

Genetic algorithms for the set partitioning problem

Levine [103] designed a parallel genetic algorithm for the SPP. This hybrid GA is based on an island model with multiple independent subpopulations (each run on a different processor) and highly fit individuals occasionally migrate between the subpopulations. He uses the "traditional" binary representation of the solutions, in which a gene in the j-position has the value 1 if column j is in the solution and the value 0 otherwise. Binary tournament selection is used to choose chromosomes for reproduction. At each generation, the hybrid GA applies a two-point crossover operator or standard mutation to a pair of parents and a very simple local search heuristic is applied to each child. The infeasible solutions (i.e. solutions with uncovered or overcovered rows) are penalised by means of a linear penalty term that assigns a penalty to each constraint violation. Hence, the fitness function is made of two different components ($fit(x) = c(x) + p(x)$), where $c(x)$ is the objective function and $p(x)$ is the penalty function. The main disadvantage of this approach is that choosing an appropriate penalty function is a difficult task. If infeasible solutions are too penalised, those solutions potentially carrying useful information might be discarded from the population. On the other hand, if the penalty is not strong enough, the GA may search only among infeasible solutions. The parallel GA was applied to a subset of the benchmark problem data sets provided by Hoffman and Padberg [89]. Although the algorithm was capable of finding optimal solutions for some problems with a few thousand columns, it had difficulty in finding feasible solutions for problems with many rows.

Chu and Beasley [33] and [32] have proposed a different GA approach for the SPP. Instead of using a penalty function to penalise constraint violation, they suggest a separation of the fitness function into two different functions: one calculates the objective function value of a solution, and the other, called unfitness, measures the amount of infeasibility of that solution. The authors used a binary coding scheme, a uniform crossover operator, a static mutation operator and also a dynamic mutation operator, that mutates only certain genes at each iteration according to given rules. Most of the times, these crossover and mutation operators do not produce feasible solutions and hence a heuristic repairing operator is used, trying to reduce constraint violations. This operator starts by removing columns until all rows are covered by at most one column. Then columns are added in order to cover as many rows as possible without causing any overcover. Note that this heuristic operator does not guarantee to produce a feasible solution, since it allows uncovered rows (but not overcovered rows). The authors have designed
a specialised parent selection method that takes into account the total number of rows covered by both parents and those which are common to both of them. The population replacement scheme is based on the fitness and unfitness scores of the solutions. The GA was applied to the whole set of problems provided by Hoffman and Padberg[89] and the results have shown that the GA were able to find the optimal solution to almost all the problems.

Genetic algorithms for the set covering problem

Beasley and Chu [16] have proposed a GA for the SCP, based on a binary coding scheme, on binary tournament for parent selection and on the steady state replacement method. They have designed a specialised crossover operator, the fusion operator, which produces a single child. The choice of whose gene values from the parents should be passed to the child is based on the relative fitness of the parents, and hence the inheritance of a particular gene from a fitter parent is more likely to occur than that from a less fit parent. A mutation operator that inverts bits in the solution with a small probability is applied. Since the solutions generated by these crossover and mutation operators may violate the problems constraints (some rows may not be covered), a repairing operator was designed in order to restore feasibility. This heuristic operator adds columns until all rows are covered and, at the same time, removes the redundant columns from the solution. A redundant column is one such that by removing it from the solution, the solution still remains feasible. The algorithm was tested on randomly generated SCP instances from the OR-Library [12] and the results indicated that the GA was able to produce high-quality solutions for large problems and optimal solutions for small problems.

Eremeev et al [58] have designed a hybrid GA for the SCP with a non-binary representation. In this representation, each gene corresponds to a row of the problem and contains a column index that is assigned to cover that row and it always represents a feasible solution (no rows are left uncovered). However, it requires a repairing operator to eliminate the redundant columns from the solution after crossover and mutation operators are applied. A crossover operator, named LP-crossover, was specially designed for the SCP, trying to find the best possible combination of parents. In order to achieve this goal, a reduced version of the original SCP is constructed by considering only those columns that were selected for the solution in the parents. After a reduction procedure that further removes some columns and rows from the subproblem, the Dual Simplex method is used to solve the linear relaxation of the reduced subproblem. The LP-crossover returns the solution obtained by Simplex method when it is integer. Otherwise, the crossover operator returns one of the unchanged parents. Computational tests were made on the same OR-Library benchmark problem sets used by Beasley and Chu [16] and provided analogous or even better results.
3.4. METAHEURISTICS

Genetic Algorithms for the BDSP

Wren and Wren [158] have made an exploratory work on applying GA to the BDSP based on a SCP formulation. Clement and Wren [34] furthered the investigation, proposing a GA approach with a binary coding and greedy crossover operators. Instead of producing a random child (solution) from the parents, greedy crossover uses a heuristic procedure that tries to produce a "good" offspring, based on a defined evaluation function. The basic process can be described as follows: duties from both parents are grouped in a set named fertilised cover. Then, a heuristic is applied to this set in order to find a solution that minimises a given criterion. The authors have applied several such heuristics, namely a greedy heuristic that iteratively chooses the duty that covers the maximum number of uncovered pieces-of-work, and an ILP-based procedure that uses a mathematical programming software to solve the associated SCP restricted to the duties in the fertilised cover. The approach was tested on small real world problems, but the results were not very satisfactory in terms of quality of the solutions, since an integer linear programming approach was used for comparison purposes and it always provided better solutions.

Kwan et al [101] and [99] developed a hybrid GA that provides a small set of duties to seed greedy heuristics, instead of producing complete schedules. This approach is based on the concept of combinatorial traits, which are certain patterns or characteristics of a good solution, such as some combinations of pieces-of-work and relief opportunities. First, the problem is formulated as a SCP and the associated relaxed LP is solved. The values of the variables in the resulting solution lie between 0 and 1. The potential seeding duty set is composed by all the duties with a value in the relaxed LP solution greater than a given parameter (e.g. 0.3). A chromosome is represented by a standard binary string, in which each gene indicates whether a particular potential seeding duty is selected or not. There is other associated information stored along with each chromosome, namely the inherited combinatorial traits and the associated solution constructed by greedy heuristics. The authors adopted the single point crossover operator while guaranteeing that all offspring inherits the combinatorial traits from one of the parents, chosen randomly. Since a chromosome does not correspond to a complete solution, a greedy heuristic is used in order to construct a driver schedule. The heuristic first uses the seeding duties of the chromosome and the combinatorial traits to select the duties in the solution. The other duties are selected taking into account the number of uncovered pieces-of-work they can cover. This approach was tested on several real-life problems from the public transport industry and the results have shown that this GA can achieve good solutions, comparable with those obtained by the traditional ILP method used in TRACS II.
Multiobjective genetic algorithms

In recent years there has been a growing interest in Multiobjective Genetic Algorithms (MOGA) [67], [162], [36], which have been applied to many hard multiobjective problems. However, only a few are devoted to the BDSP or to any of the associated SCP or SPP models. For the purpose of this section it is sufficient to say that MOGA try to generate an approximation of the non-dominated set of solutions. A Multiobjective Optimisation Problem (MOP) can be "transformed" into a Single Objective Optimisation Problem (SOP) by a scalarising function that assigns appropriate weights to each objective function [162]. Many of the GA approaches to MOP use the scalarising function in the fitness assignment procedure, while in other approaches the fitness assignment is based on a ranking procedure induced by the dominance relation. All the approaches discussed in this section use a scalarising function. A more complete review of multiobjective genetic algorithms will be presented in Chapter 4.

Particularly concerning the SCP, Jaszkiewicz [94] has described a comparative study of several multiobjective metaheuristics for the bi-objective SCP. The tests were performed for ten different metaheuristics based on GA, hybrid GA, multiple start local search and simulated annealing. The results have shown that any of the metaheuristics was able to outperform all other methods on all instances. Furthermore, the results indicated that they differ radically even if the methods are based on the same single objective algorithm and use the same operators. He concluded that the elements of the methods that are specific to the multiobjective case have substantial influence on their performance.

A GA approach for the BDSP involving several objective functions was proposed by Dias [56]. The algorithm was embedded in the GIST system and it was able to produce solutions that were closer to the planners expectations than the current solutions (obtained either manually or through an LP-based method [10]). Furthermore, that work has inspired and motivated further developments also described in this thesis [54], [53]. The approach was based on a generalised SPF in which the uncovered pieces-of-work (leftovers) were penalised in the objective function. Besides the duties cost, several other objectives can be considered, such as the number of duties, the total duration of the leftovers, the average duration of the duties and the percentage of duties of a certain type. The weight values assigned to each objective function are defined interactively by the user that analyses the solutions obtained after a certain number of generations and chooses the appropriate combination of weights. This approach uses a non-binary coding scheme similar to that adopted by Eremin et al [58] for the SCP, and specialised crossover and mutation operators developed to easily preserve feasibility.

Lourenço et al [107] designed a GA for the BDSP that is also embedded in the GIST System. However, in this case, the BDSP is formulated as a SCP. The authors also consider several objectives, namely the
minimisation of the total costs, the minimisation of the uncovered and overcovered rows, the minimisation of the number of duties, the minimisation of the number of duties with a single piece-of-work and the minimisation of the number of vehicle changes. This GA is based on the work of Beasley and Chu [16], but considering the multiobjective aspects of the problem. Hence, parent selection population replacement strategies have been adapted. A binary tournament strategy was used for parent selection, by randomly choosing one of the objective functions to compare the solutions. The population replacement scheme is based on dominance criteria. Each new generated chromosome is sequentially compared with the existing solutions until one of the following situations occur: the child is dominated by one of the solutions in the population and is discarded; the child dominates an existing solution and replaces it in the population; if any of the previous situations happens, the new chromosome is simply added to the population. A binary coding scheme was adopted and the two-point crossover and the standard mutation operators were used. A specialised crossover operator, similar to that proposed by Eremeev [58], was also designed. The GA was tested on real data sets provided by several Portuguese mass transit companies and the results were compared with a TS approach (briefly presented in Section 3.4.3) and with a LP-based algorithm for the SCP that was also available in the system. Although the authors have concluded that TS outperformed the other methods, they state that the GA approach considering several objectives compares favorably with the LP-based solution.

3.4.2 Evolutionary Strategies

Basic principles

An Evolutionary Strategy (ES) is a metaheuristic similar to a GAs in several aspects. However, while a GA maintains a population of solutions during each generation, a ES operates on a single solution, generating a child from one parent during each generation. Furthermore, while a GA probabilistically selects a set of solutions from the current population, a ES selects the worse components of a solution to discard.

The basic algorithm iteratively performs a sequence of three steps (Evaluation, Selection and Construction) on a single solution. In the Evaluation step, a quality measure of each component of the current solution is computed. The Selection step uses this measure to select some components of the solution, through rules that are partially based on randomness. The resulting partial solution is then fed to the Construction step to complete that partial solution. Throughout these iterations, the best solution is kept and finally returned as the final solution.

Marchiori and Steenbeek [111] have proposed an ES for the SCP and applied it to airline crew scheduling problems. The approach was compared with the GA proposed by Beasley and Chu [16]
and Eremeev [58], and also with the lagrangean relaxation method adopted by Caprara et al [24], with satisfactory results. The ES extracts an initial subset of columns (the core) from the set of all generated columns. In an initialization procedure, an approximated solution is constructed using a greedy heuristic, which is further improved using a local search optimisation procedure. Then the algorithm consists on an iterated procedure, following the standard structure of a ES. The solution is evaluated and compared with the best solution found up to now. If the new solution has a lower cost, it replaces the current best solution. Next, some columns in the best solution are selected for the initial partial solution of the next iteration. Then, a new approximated solution is constructed using the greedy heuristic and the local search optimisation algorithm. These steps are repeated until no further improvements can be achieved or a given number of iterations is reached. The size of the core is determined by an adaptive parameter and the selection of a column is specified by a suitable merit criterion. During the execution, the score of the columns (contribution for the solution value) is modified as well as the size parameter, and the core is dynamically updated.

Another interesting ES was suggested by Li and Kwan [105]. The authors have designed and implemented an ES for the BDSP based on a SCP formulation that combines the features of an evolutionary strategy with a "fuzzy" evaluation of the components. The basic structure of the ES follows the traditional steps but some concepts based on fuzzy set theory are introduced in the Evaluation and Construction steps. An initial seed solution is generated by a greedy heuristic that sequentially adds the column with the highest coverage ratio (number of rows covered / cost) to the solution until all rows are covered. In the Evaluation step, the quality of an individual duty (column) in a complete solution is computed based on a combination of two factors: an overcover penalty and a structural coefficient. The computation of the structural coefficient of a duty is based on fuzzy concepts, since among the large set of generated duties, it is difficult to decide which has a more effective structure. The main factors considered for the structural coefficient of a duty have been the total worked time, the spreadover, the ratio of total worked time to spreadover and the number of spells. The evaluation of each duty according to these factors is based on fuzzy transformations that measure the quality of the individual duty within the whole set of generated duties. When all the duties in the solution are evaluated, a selection procedure selects with higher probability those duties with higher evaluation. Finally, the Construction step takes the partial solution composed of the selected duties and builds a new complete solution. In this procedure the choice of the duties that will complete the solution is also based on the fuzzy evaluation function defined. The ES was tested on actual problems from bus and train companies and the results were very promising.
3.4. METAHEURISTICS

3.4.3 Tabu Search

Basic principles

Tabu Search (TS) [76], [77] is a class of local search based metaheuristics that impose some particular restrictions to guide the search process. These restrictions operate in several forms, both by direct exclusion of alternatives, classified as tabu, and by modifying evaluations and probabilities of selection of such alternatives [37]. The search process operates by performing a set of modifications (moves) on a feasible solution which lead to other feasible solutions (thus defining some kind of neighbourhood structure).

The fundamental element underlying TS is the use of a flexible memory, that creates and exploits structures for taking advantage of the history of the search process. Taking into account the previous moves (the history), the memory structures modify the neighborhood of the current solution and the cost associated to each element in the neighbourhood. At each iteration, the best evaluated solution in the modified neighborhood is selected.

A subset of the moves in the neighbourhood of the current solution is classified as tabu and is recorded in a tabu list. The tabu list stores the attributes of moves that might lead back to the old, already visited solutions, aiming at preventing cycling and guiding the search towards promising and unexplored regions of the search space. At each iteration the tabu list memorises a new set of attributes and discards the old ones. A key issue in applying TS is to determine a good value for the length of the tabu list. Tabu restrictions can be violated under certain circumstances. For example, when a tabu move would result in a solution better than any visited so far, its tabu classification can be overridden.

Tabu Search approaches to the BDSP

Lourenço et al [107] proposed a TS algorithm for the BDSP. They used a formulation analogous to that used for the GA approach, described in Section 3.4.1.4. The initial solution is obtained either applying a random procedure or a greedy heuristic. Three different neighbourhoods were designed, namely exchange, insert and remove neighbourhoods. In the exchange neighbourhood, a column is removed and a new column, covering at least one uncovered row, is added. The insert neighbourhood contains all the solutions that can be obtained from the current one by adding one column, while the remove neighbourhood considers the solutions that can be obtained from the current one by removing one column. Since the size of the insert neighbourhood can be very large, a candidate list strategy is adopted. The candidate list only considers columns that cover an uncovered or single covered row and whose cost is smaller or equal to the average cost of the solution. Each of the three neighbourhoods is applied a given number of times, following a specified order. Two tabu lists with different sizes were
designed, each one storing the columns that have been recently added and removed from the current solution. A tabu move can be performed whenever it leads to a solution with an objective function value smaller than the best found so far. The TS procedure was applied to a set of different objective functions already described in Section 3.4.1.4. The procedure first applies TS to each single objective function and afterwards a weighted sum function is considered. All non-dominated solutions found are stored and presented to the decision-maker. This approach also suggests an intensification strategy that consists in applying a Greedy Randomised Adaptative Search Procedure [62] (GRASP) or a LP based method to a subset of columns. This subset of columns is found by the application of the TS procedure for several iterations using only the insert neighbourhood. The resulting problem is smaller than the original and it does not contains the most expensive columns, which have been inserted in the tabu list. TS combined with intensification strategies has been applied with success to several scheduling problems [106].

Shen and Kwan [136] suggest an alternative TS approach for the BDSP. In this approach the set of feasible candidate duties is not previously generated. Instead, duties that may be infeasible are built within the TS procedure. The approach relies on a refining process with three main components: minimising infeasibility, minimising total costs and adding duties. The initial solution is the starting point of local search, and most TS approaches put a lot of effort in building a good initial feasible solution [107]. In contrast, Shen and Kwan use a rather simple and quick method to construct the initial solution, trying to cover all pieces-of-work with the minimum number of duties. However, the initial schedule can contain infeasible duties. TS is then applied in two phases: minimising infeasibility and minimising cost. The authors have defined four multi-neighbourhood structures, based on the following move operations: swapping two links, swapping two spells, inserting one spell and recutting block. The best move in the neighbourhood of the current solution is calculated differently according to the phase of the algorithm. In the first phase, the best move is the one that most reduces the infeasibility while in the second phase the best move is the one that most reduces the cost of the current solution. Different memory structures were designed to implement the tabu lists, in order to store the forbidden (most recent) moves. The length of the tabu lists, which decides the number of iterations for which the current selected move must remain tabu, is based on values suggested by Glover and Laguna [78]. If, at the end of the TS procedure, the solution is infeasible, an heuristic is applied in order to restore feasibility. This heuristic consists in adding duties until all rows are covered. Then, the TS procedure restarts, trying to improve the new solution. Experiments with real world problems have been made and comparisons with results provided by the ILP-based generate-and-select TRACS II approach have shown that TS can produce competitive solutions considerably quicker than TRACS II.
3.5 Summary

In this chapter we have reviewed the most widely used approaches to the BDSP. While pure heuristic methods are being abandoned and replaced by hybrid approaches that combine mathematical programming techniques and heuristics, the number of approaches based on metaheuristics is consistently increasing. Within mathematical programming methods, column generation is the most promising approach, specially when combined with constraint programming techniques. Integrated approaches for vehicle and driver scheduling are also an interesting and active current field of research.

A lot of mathematical programming and metaheuristic approaches have been proposed for the SCP, while only a few are devoted to the SPP. The BDSP is commonly solved using one of these two formulations and there is no evidence whether one formulation is more suitable than the other (probably because such an assessment depends on the particular features of the problem instance). When a particular technique is applied to the SCP, many authors agree that solutions with overcoves have to be manually adjusted in order to remove the overcovered pieces-of-work, which they claim to be a minor problem. Other authors add constraints or penalties in order to reduce the amount of overcover in the solutions. On the other hand, constraints in the SPP are very restrictive and it is often extremely difficult to find a feasible solution, with any overcovered or uncovered row.

BDSP instances are often very large, involving several thousands or millions of generated duties. Although memory and speed of computers are always increasing, the computational time required by optimisation techniques is still a major practical problem.

Moreover, when attempting to implement these techniques to real mass transit companies, their particular features are seldom considered by standard formulations. Some complex contraints and objective functions must be added to the model in order to fulfil the requirements of the driver scheduling process in a transport company. The resulting models would become too complex to be tackled as a whole, and hence some of the constraints have to be relaxed in the core problem and used in greedy heuristics or in refinement procedures to improve the current solution.

Metaheuristics have given an important contribution in handling complex problems such the BDSP, since they are not constrained by any particular formulation and can easily consider the particular features of the real problem. However, metaheuristics are not being able to outperform mathematical programming methods when pure formulations are considered although, in practical applications, they have proved to be able to quickly produce good solutions, close to the real problem requirements and easy to implement.

Some of these considerations will be further explored in Chapter 6.
Chapter 4

Genetic algorithms for multiobjective optimisation

4.1 Introduction

Multiobjective Optimisation may in practice be extremely complex. Most of the traditional approaches try to reduce the search space into a more manageable size, by aggregating objectives or by linearizing and relaxing objectives and constraints. In the last few years, the application of metaheuristics to multiobjective problems became an active area of research [67], [36], [163]. In particular, genetic algorithms seem to be particularly well suited to tackle multiobjective problems, since the process that guides the search is able to produce several potentially non-dominated solutions in a single run, as well as to improve solution by exploiting their good features. Furthermore, genetic algorithms have proven to be a robust and powerful search technique.

In this chapter we present an overview of genetic algorithms (GA) with a particular emphasis on their application to multiobjective optimisation problems.

In Section 4.2 the main concepts of multiobjective optimisation are outlined and, in Section 4.3, we present a summary of the most common traditional approaches to multiobjective optimisation problems.

In Section 4.4 we present the fundamental concepts and describe the main components of a genetic algorithm. Finally, in Section 4.5 some of the most advanced genetic algorithms for multiobjective optimisation are briefly described.
4.2 Basic concepts of multiobjective optimisation

Consider a problem with \( n \) decision variables, \( m \) constraints and \( k \) objective functions. A multiobjective optimisation problem consists in finding

\[
\begin{align*}
\min \quad & y = F(x) = (f_1(x), f_2(x), \ldots, f_k(x)) \\
\text{s.t.} \quad & g_i(x) \geq 0, \ i = 1, \ldots, m
\end{align*}
\]  \hspace{1cm} (4.1)

where a solution (decision vector) \( x = (x_1, x_2, \ldots, x_n) \in X \), with \( X \) being the decision space. An objective vector is \( y = (y_1, y_2, \ldots, y_k) \in Y \), where \( Y \) defines the objective space.

The set \( X_F = \{ x \in X : g_i(x) \geq 0, i = 1, \ldots, m \} \) is known as the feasible set.

Notation 4.1 For any two objective vectors \( F(a), F(b) \in Y \)

\[
\begin{align*}
F(a) = F(b) & \iff f_i(a) = f_i(b), \forall i \in \{1, \ldots, k\} \\
F(a) \leq F(b) & \iff f_i(a) \leq f_i(b), \forall i \in \{1, \ldots, k\} \\
F(a) < F(b) & \iff F(a) \leq F(b) \wedge F(a) \neq F(b)
\end{align*}
\]

Definition 4.1 (Pareto dominance) For any two solutions \( a, b \in X_F \), we say that

\( a \) weakly dominates \( b \) (\( b \preceq a \)) iff \( F(a) \leq F(b) \)

\( a \) dominates \( b \) (\( b < a \)) iff \( F(a) < F(b) \)

\( a \) is indifferent to \( b \) (\( a \sim b \)) iff \( a \not< b \wedge b \not< a \)

Definition 4.2 (Nondominated solution) A solution \( x \in X_F \) is non-dominated regarding \( A \subseteq X_F \)

iff

\[ \nexists a \in A : x \prec a. \]

The set \( P_A = \{ a \in A : a \) is non-dominated regarding \( A \} \) is the non-dominated set regarding \( A \).

Definition 4.3 (Pareto optimality) A solution \( x \in X_f \) is said to be Pareto optimal if it is non-dominated regarding \( X_f \).

The set of all Pareto optimal solutions is called \( \text{Pareto optimal set} \). The set of the corresponding objective vectors is called the Pareto optimal front or the trade-off surface.
4.3. SOME GENERAL APPROACHES FOR MULTIOBJECTIVE OPTIMISATION

Definition 4.4 (Convex set) A set $A$ is convex if $\forall a, b \in A$ and $\forall \alpha \in [0, 1]$, 

$$F(\alpha \times a + (1 - \alpha) \times b) \leq \alpha \times F(a) + (1 - \alpha) \times F(b)$$

A set $A$ is convex if, for any two points in the set, the line segment joining these points is also in the set. As depicted in Figure 4-1, the set on the left (A) is convex, while the set on the right (B) is non-convex.

![Figure 4-1: Example of a convex set (A) and of a non-convex set (B)](image)

4.3 Some general approaches for multiobjective optimisation

In a multiobjective optimisation problem the concept of optimal solution is usually meaningless. Instead, there is a large set of alternative solutions (the Pareto optimal set), representing different trade-offs of the objectives. Decision making plays here an important role since there is no unquestionable right or wrong decisions and different decision makers may choose different solutions, based on their own set of preferences. Hence, in a multiobjective problem we have two different processes: the search process itself, that tries to find a set of alternative solutions, and the decision making process, that consists in the choice of one or more of the non-dominated solutions, by a human decision maker.

The decision making process can occur before the search, by allowing the decision maker to assign preferences (or priorities) to each criterion. This method is appropriate when the decision maker knows *a priori* which criteria are more important and how they interact with each other. Alternatively, the decision making process can occur after the search, by allowing the decision maker to choose one or more solutions from a set of non-dominated solutions that have been previously found by the search process.
Here, the search is performed without any given preference information. However, if the Pareto optimal set is too large, it may be difficult for the decision maker to choose a solution.

Another approach consists in an interactive combination of decision making and search. In this method, the decision maker assigns preferences during the search process. Regularly (after each step, for example), a set of alternative solutions is presented to the decision maker who chooses those that represent (from his point of view) the best trade-off. These partial decisions are then used to guide the search.

Concerning the search process, several methods have been proposed over the years. Since most of the problems are far too complex to allow enumeration and evaluation of all possible solutions, many of these techniques try to reduce the search space to a more manageable size, by aggregating objectives or by linearizing and relaxing objectives and constraints. Here we will only present a brief description of some of the most common approaches, namely the Weighted Sum method, the Goal Programming method, the Constraint method and the Sequential Optimisation method.

The Weighted Sum method is one of the most popular approaches to multiobjective problems and it consists in forming a linear combination of the objectives. This approach is simple and intuitive and, for certain problems, it can also be very efficient. The original multiobjective problem 4.1 is transformed into a single optimisation problem of the form

\[
\begin{align*}
\min & \quad y = \sum_{i=1}^{k} w_i \cdot f_i(x) \\
\text{s.t.} & \quad g_i(x) \geq 0, \quad i = 1, \ldots, m
\end{align*}
\]  

(4.2)

where \( \sum_{i=1}^{k} w_i = 1 \) and \( w_i \geq 0, \forall i = 1, \ldots, k \). The weighting coefficients \( w_i \) represent the relative importance of the objectives. When little information is available on how to choose these coefficients, as it is usually the case, one needs to solve problem 4.2 for several different weighting combination values thus yielding a set of non-dominated solutions. In this case, any non-dominated solution can be derived from the weighting coefficients used.

The main disadvantage of this method is that it cannot generate Pareto optimal solutions that lie on non-convex regions of the trade-off surface. To illustrate the idea, consider Figure 4-2, for a maximisation problem with two objectives and a non-convex trade-off surface. As depicted in the figure, the weighted sum approach applied to this problem for this particular weights values \( w_1 \) and \( w_2 \) would not produce solution \( C \) that, in fact, belongs to the Pareto optimal front.

The Goal Programming method is based on minimizing deviations from target values. In this technique, the decision maker has to assign target values or goals that he wishes to achieve for each objective.
4.3. SOME GENERAL APPROACHES FOR MULTIOBJECTIVE OPTIMISATION

![Diagram showing an application of the weighting sum method to a non-convex trade-off surface.](image)

Figure 4.2: Application of the weighting sum method to a non-convex trade-off surface

In this method the multiobjective optimisation problem is transformed into a single (scalar) optimisation problem that can be formulated as follows:

\[
\begin{align*}
\min & \quad y = \sum_{i=1}^{k} |f_i(x) - T_i| \\
\text{s.t.} & \quad g_i(x) \geq 0, \quad i = 1, \ldots, m
\end{align*}
\] (4.3)

This method may be very efficient [36], although it has the disadvantage of needing prior information about the desired target values, which is not always available.

The Constraint method consists in minimizing one of the objectives while considering the others as constraints, yielding a single optimisation problem that can be formulated as follows:

\[
\begin{align*}
\min & \quad y = f_r(x) \\
\text{s.t.} & \quad f_i(x) \leq \epsilon_i, \quad \forall i = 1, \ldots, k \text{ and } i \neq r \\
& \quad g_i(x) \geq 0, \quad i = 1, \ldots, m
\end{align*}
\] (4.4)

Problem 4.4 is solved several times for different values of the upper bounds \( \epsilon_i \), so that several non-dominated solutions are obtained. Usually, the objective chosen to be minimised (the \( r \)-th objective) is the most preferred one, but it may be also necessary to repeat the procedure for different values of \( r \). Obviously, choosing appropriate values for \( \epsilon_i \) is a fundamental decision in this technique, since if these
values are not appropriately chosen, the resulting feasible set may be empty. Besides, this technique may become very inefficient, particularly when several single optimisation problems must be solved.

Finally, the *Sequential Optimisation* method ranks the objectives according to preferences established by the decision maker and performs optimisation procedures in lexicographic order. A single optimisation problem is solved for each objective, by order of importance. The optimal solution of each problem is viewed as a goal and used as a constraint to the next. Hence the i-th problem can be formulated as follows:

\[
\begin{align*}
\min & \quad y = f_i(x) \\
\text{s.t.} & \quad f_j(x) = f_j^*, \forall j = 1, \ldots, i - 1 \\
& \quad g_i(x) \geq 0, \quad i = 1, \ldots, m
\end{align*}
\] (4.5)

where \( j \) reflects the order of importance of the objectives and \( f_j^* \) are the optimal solutions of the problems solved before problem \( i \).

Naturally, this problem relies on the capability of the decision maker to establish an order of importance for each objective. When this is not possible, the method is applied several times according to different preference vectors, arbitrarily generated.

### 4.4 Basic concepts of Genetic Algorithms

Genetic Algorithms (GAs) were introduced by Holland in 1975 [91] and have been applied to a wide variety of problems, from machine learning and pattern recognition to many difficult Combinatorial Optimisation problems. In this section we briefly present and discuss the main structure and components of a GA.

GAs are inspired on the evolutionary process of biological organisms in nature [44]. In this process, natural populations evolve through generations according to the principles of natural selection and "survival of the fittest". Individuals that are more successful in adapting to the environment will have a higher chance of surviving and reproducing, while less fit individuals will be eliminated. Hence, the genes of highly fit individuals will "produce" a large number of individuals in the next generations. The underlying idea is that combining good characteristics from highly fit ancestors may produce even more fit offspring.

GAs use a direct analogy of this natural process. They use a *population* of individuals, each representing a possible solution to the problem under consideration. Each individual in the population is encoded into a string or *chromosome* and is assigned a fitness score that measures its quality. The *fitness* of an individual is evaluated according to a given objective function. Highly fit individuals are
given the opportunity to reproduce by exchanging parts of their genetic information with other highly fit individuals, through a crossover operator. This process produces new individuals as offspring that share some features of the parents. The least fit individuals of the population are less likely to get selected for reproduction and tend to disappear. Occasionally, mutation is applied to an individual after crossover, by changing some values in its chromosome. The offspring will replace some of the members in the current population and the whole evaluation-selection-reproduction-replacement cycle is repeated until a satisfactory solution is found or some other termination criterion is met. The main steps of a GA are illustrated in Figure 4-3.

![Diagram of GA process]

**Figure 4-3: Basic structure of a GA**

In the *Initialization* step, an initial population composed of a fixed number of elements is created. It can be randomly generated or obtained by applying heuristic procedures that try to produce initial good individuals.

In the *Evaluation* step, all the individuals in the population are evaluated through the objective function. The evaluation process does not rank the individuals in the populations: it only assigns the objective function value to each individual.

The *Fitness Assignment* procedure ranks the individuals in the population according to that evaluation. In general, the fitness function results from the composition of two functions: one is the objective function and the other one transforms the value of the objective function into a non-negative number. The individuals in the population are ranked in such a way that the best individuals are assigned higher
fitness values.

The Selection step consists in selecting the individuals that will be involved in the reproduction procedures (crossover and mutation), according to their fitness values.

Then, Crossover and Mutation operators are applied to the individuals that have been selected for reproduction, according to given probability rates that can be dynamically updated throughout the execution of the algorithm.

The Population Replacement scheme is responsible for the replacement of the older individuals by the new ones. The Termination Criterion decides when the algorithm should stop and typically this happens when a given number of generations is reached or when the best solution obtained so far is satisfactory. If the termination criterion is not met, a new cycle begins.

Although every GA is composed by these steps (see Figure 4-3), for each particular step several different methods have been proposed. For the canonical (standard) GA proposed by Holland [91], several crossover and mutation operators have been suggested, tested and compared for different classes of problems. During the last years, alternative strategies for fitness assignment, parent selection and population replacement have been proposed and a wide variety of both theoretical and empirical results have been discussed. A complete review of these methods and the corresponding results is beyond the scope of this work and it would not be so useful in this context, since the basic principles of a GA are currently quite well known.

Besides, the majority of the methods proposed are based on the canonical GA, for which many theoretical results have been obtained. However, it is currently recognised by most practitioners and theoreticians that the canonical GA presents strong limitations in certain classes of problems [83], [82] and the extension of those results to non-standard GAs is not clear. Moreover, even for the canonical GA, the conclusions of which is the best method are not consensual and some contradictory results can be found. In fact, according to the No Free Lunch Theorem [153], [96], two different techniques applied to a particular problem instance can produce completely different results but, when applied to the set of all instances, the average performance of both techniques may be similar.

When GAs were introduced and first applied, they have been presented as a general local search technique which could be applied to a wide variety of problems with only slight changes in the parameters. Basically, one had only to choose the appropriate coding, the fitness function and tune the different parameters involved (for example, the crossover and mutation rates or the rate of individuals to replace at each generation). This approach had the advantage of being quite generic, but it turned out to be very ineffective in many real world applications.

In the last few years, standard GAs have been less used and hybrid approaches have gained popularity. Hybrid approaches can incorporate problem specific knowledge, use non-binary codings or apply
4.5. MULTIOBJECTIVE GENETIC ALGORITHMS

hybrid operators that include other optimisation techniques. They are clearly less generic, but can be very effective for the particular problem they intend to solve. Particularly when applied to real world problems, we can find several different GA approaches, designed and tuned for each particular situation. At the same time, some people have expressed their opinion that over-specialised GA are less elegant and not easy to understand. These arguments can make sense, but we can also cite Ackoff [2], by saying that instead of adapting the problem to the technique, we should do the opposite. In fact, hybrid GAs are clearly more in line with this perspective.

4.5 Multiobjective Genetic Algorithms

The basic structure of a genetic algorithm for multiobjective optimisation is analogous to that of a standard single objective GA, as was shown in Figure 4-3. However, the particular techniques used in some of the stages may be considerably different. While in a single objective GA we are searching for a single solution, in a multiobjective genetic algorithm we are trying to find a set of non-dominated solutions. Hence, the design of a GA for a multiobjective problem should take into account the following issues: how to cover most of the search space, thus finding a large number of good solutions, how to obtain a set of non-dominated solutions as close as possible to the Pareto optimal set and, finally, how to ensure a good spread of the trade-off surface, thus providing a diversified set of alternatives.

Due to stochastic errors and to selection pressure, among other factors, GAs tend to converge to a single local optimum solution [109], [48]. This problem, known as genetic drift [46], has serious implications in multiobjective optimisation, for which population diversity is crucial. In order to overcome this convergence problem, several methods have been proposed. It is beyond the scope of this thesis to analyse such methods. However, we briefly describe fitness sharing, one of the most frequently used techniques, namely by most of the multiobjective genetic algorithms described in the forthcoming sections.

The basic idea of fitness sharing [85] is to decrease the fitness, fit, of those solutions that are in the neighbourhood of many other solutions. Here, the neighbourhood is computed based on a given distance measure \( d(i, j) \) and on an input parameter, the niche radius \( \sigma_{sh} \), that defines the maximum distance allowed.

The distance \( d(i, j) \) between two individuals can be measured in the objective space or in the decision space. The distance in the objective space, also called phenotypic sharing is usually calculated by the euclidean distance between the objective vectors of chromosomes \( i \) and \( j \). The distance in the decision space, also called genotypic sharing is often calculated by the Hamming distance between the strings of \( i \) and \( j \). Several experiments performed by [48] indicated that phenotypic sharing was in general more useful than genotypic sharing. Then, a sharing function \( \phi(d(i, j)) \) is defined as follows:
\[
\phi(d(i,j)) = \begin{cases} 
1 - \left( \frac{d(i,j)}{d_{\text{share}}} \right)^\alpha, & d(i,j) < \sigma_{\text{sh}} \\
0, & \text{otherwise}
\end{cases}
\]

where usually \( \alpha = 1 \), and the final fitness of an individual \( i \), \( \text{fit}'(i) \), is calculated as:

\[
\text{fit}'(i) = \frac{\text{fit}(i)}{\sum_{j \in P_i} \phi(d(i,j))}
\]

Another important issue when trying to obtain a good representation of the Pareto optimal set is the population replacement scheme. When a GA is applied to a single objective optimisation problem it is very common to apply a strategy known as elitism, that consists in keeping the best solution of a generation in the next generation. In a multiobjective optimisation problem, we have several "best" solutions (the non-dominated solutions), hence we have to adapt elitism to this context. Two alternative elitist strategies have been proposed for multiobjective optimisation problems. The first simply copies the non-dominated solutions from one generation to the next, while the second maintains an external set, also known as archive, composed by all non-dominated solutions found so far. In both approaches we need to decide how many solutions to keep, for how long and whether they are going to be used to produce new solutions or only in the selection process.

Next, we present a brief overview of the most popular GA approaches for multiobjective optimisation. We have classified these approaches into two major groups: weighting based approaches that, more or less explicitly, assign weights to the objectives, and Pareto based approaches, that directly use the concept of Pareto dominance.

### 4.5.1 Weighting based approaches

One of the first GA approaches to multiobjective optimisation was proposed by Schaffer [135] who presented the Vector Evaluated Genetic Algorithm (VEGA) that handles equal portions of the population according to distinct objectives. In this algorithm, a fitness proportionate selection operator is performed for each objective separately, producing a number of sub-populations equal to the number of objectives. These sub-populations are merged together to produce a new population. Then, crossover and mutation operators can be applied in the usual way to the whole population. VEGA is very simple and reported some success, although some serious drawbacks have also been reported. In fact, it has been proved [130], [66] that this method was equivalent to a linear combination of the objectives and it would miss solutions in concave regions of the search space. Moreover, Schaffer himself stated that this method could lead to solutions that do not have the best value for any objective function, but only moderately
4.5. **MULTIOBJECTIVE GENETIC ALGORITHMS**

good values for all of them.

A GA based on the aggregating approach was proposed by Hajela and Lin [87] (here referred to as HL-WGA) who have associated different combinations of weights of each objective to each chromosome. The idea was to produce a set of non-dominated solutions corresponding to different weighting functions in each run of the algorithm. HL-WGA uses fitness sharing in order to promote the diversity of the weight combinations and restricted mating in order to speed convergence. Coello [35] has reported that, although premature convergence can be avoided by fitness sharing, this method can create a very high selection pressure if certain combinations of weights are produced at early stages of the search. Furthermore, this method has the disadvantages of any aggregating approach, biasing the solutions towards convex regions of the Pareto front.

### 4.5.2 Pareto based approaches

Fonseca and Fleming [65] have proposed the *Multiobjective Genetic Algorithm* (here referred as FF-MOGA) based on a Pareto ranking procedure, that ranks each solution in the current population according to the number of solutions that dominate it. Consider, without loss of generality, that the fitness function is a minimisation function (the best individuals have the lowest fitness values). In FF-MOGA, all non-dominated solutions are assigned rank 1, while the rank of a dominated solution is computed by adding 1 to the number of solutions that dominate it. The individuals in population are sorted according to this rank and linear ranking is used to compute initial raw fitness values. Those solutions with identical ranks (that are dominated by the same number of individuals in the current population) are assigned the same average fitness value, calculated by sharing and averaging the corresponding initial raw fitness values. Fitness sharing is performed in the objective space, and therefore two solutions with the same values for all objective functions cannot coexist in the same population.

The *Niched Pareto Genetic Algorithm* (NPGA) was introduced by Horn and Nafpliotis [92], [93]. Here, a Binary Tournament selection operator based on Pareto dominance is proposed. The competition between two candidate solutions is supported by a comparison set composed by random solutions selected from the population. If one (and only one) of the candidate solutions is non-dominated regarding the comparison set, that solution is selected for reproduction. When both candidates are non-dominated or dominated by some solution in the comparison set, fitness sharing is applied in order to choose the winner. According to this procedure, the winner is the candidate solution with the least number of elements in its niche (the set the solutions with distances in the objective space smaller than the niche radius $\sigma_{nk}$).
Another Pareto based approach, the Non-dominated Sorting Genetic Algorithm (NSGA) was proposed by Srinivas and Deb [144]. In this technique, at each generation, and before the selection operator is applied, all non-dominated solutions are selected and assigned a dummy fitness value (proportional to the population size). Fitness sharing (in the decision space) is used within this set of non-dominated solutions, in order to maintain the population diversity. These solutions are ignored in the next classification process and a new set of non-dominated solutions, regarding the remaining population, is formed. In the fitness assignment process, the dummy fitness values are decreased in order to guarantee that all solutions in the new set have lower fitness than those in the previous one. This process goes on until the resulting population is empty. An enhanced and faster elitist version of this method, NSGA-II, was recently proposed by [47].

Finally, the Strength Pareto Evolutionary Algorithm (SPEA), introduced by Zitzler [162], has emerged as one of the leading evolutionary algorithms for multiobjective optimisation [163]. An improved version, SPEA 2 has already been presented by [164]. We have selected this technique as the basis for one of our multiobjective genetic algorithm approaches, the ND model. It will therefore be described in more detail in Chapter 6.

At each generation, SPEA maintains two different populations: an internal population $P_t$ and an external population $P'_t$ (also known by external set). The external set contains all non-dominated solutions found so far, thus representing an estimate of the Pareto optimal set. When the size of the external set is larger than a predetermined limit, it is reduced by a clustering procedure. A binary tournament selection operator, involving solutions from both the internal and the external populations is applied, in order to select solutions for reproduction. A central concept of the algorithm is the strength of a solution that is proportional to the number of solutions it dominates. The fitness of an element of the external set is given by its strength, while the fitness of a dominated solution (in the internal population) is calculated by summing up the strengths of all non-dominated solutions that dominate it.

SPEA 2 proposes an improved fitness assignment procedure that differentiates solutions dominated by the same subset of the external set. Furthermore, a density estimation technique based on the distances in the objective space is incorporated. Finally, the reduction of the size of the external set is now based on a truncation method.

4.6 Summary

In this chapter we have presented the main concepts in multiobjective optimisation and we have discussed how some general methods tackle this class of problems. Next we have outlined the basic concepts
of genetic algorithms and we have briefly described some of the most used genetic algorithms for multiobjective optimisation problems.

This brief introduction sets up the ground for the choice of the detailed approaches to be used along this work, in particular in Chapter 6.
Chapter 5

A Multiobjective Approach to the BDSP

"If OR is to survive it must maintain a strong problem orientation, not a technique orientation. It must expand its methods and techniques to fit the problems and not contract the problems to fit available methods and techniques."

Russell Ackoff, 1961, quoted by Kirby [95]

5.1 Introduction

In this chapter, we start by discussing the reasons that justify our approach to the BDSP. This approach has naturally evolved from years of work with the planners of Portuguese urban transport companies. We believe that their concerns and problems are similar to those that can be found in many small and medium urban transport companies in other countries and cultures. We present the most relevant issues for defining the complexity of the bus driver scheduling problem and we describe some of the main difficulties that arise in the evaluation of a BDSP solution. While most of the traditional techniques are based on the minimisation of costs, we found out that most companies evaluate and compare solutions, simultaneously using rather different criteria. This fact explains the difficulty in the implementation of fully automated procedures in real companies, since algorithms often find the optimal solution of a problem that does not correspond to the real one. In our work, we have identified the main criteria used by planners to evaluate solutions in practice.

Moreover, some fundamental concepts are viewed in a quite different way by the research community and transport companies. Although every company claims that established rules must be accomplished,
our experience with Portuguese companies shows that, often, they are not respected. The concept of feasible duty is too constrained and, when we analyse an implemented solution, we often find out that most of the duties are, in fact, infeasible ones. Naturally, these duties have just slight violations of the rules, that can be ignored or accepted by the planners (and by the drivers) if they do not occur systematically.

Finally, the need to build a tractable mathematical approach for a complex system often leads to a model that is very far from the real problem. In this chapter, we intended to provide some insights into "how sometimes we are correctly solving the wrong problem".

All these issues are discussed in Section 5.2. Some fundamental concepts are redefined in Section 5.2.1. In Section 5.2.2 we discuss the issues that determine the complexity of a BDSP instance. Finally, in Section 5.2.3, we present some guidelines to evaluate solutions from the planner's point of view.

In the remaining sections, we describe the constraints and objective functions involved in the BDSP, and we try to explore some of their features that may improve the quality of the solutions. Constraints can be divided into two groups: those imposed to the construction of each separate duty, and those involved in the generation of a schedule (a feasible BDSP solution). In our approach, the generation of the feasible duties is performed before the application of the genetic algorithm, and the constraints involved in this process have already been presented and discussed (see Chapter 2).

The constraints discussed in this chapter are those involved in the construction of a feasible BDSP solution and can be divided into hard and soft constraints. Hard constraints cannot be violated and basically define the type of problem under consideration. In our approach we consider two groups of hard constraints: hard partitioning and hard covering constraints. In a BDSP with hard partitioning constraints, a feasible solution cannot contain overcovers, but undercovers are allowed. In a BDSP with hard covering constraints some overcover is allowed (and also undercover), but the amount of overcover can be limited. Hard constraints are discussed in Section 5.3.

Soft constraints can be violated although their violation is in general not desirable, and some of them correspond to the side constraints presented in Chapter 2 and in Chapter 3. Each violation of a soft constraint is penalised according to its importance and to the extent of that violation. Therefore, minimizing the violation of each soft constraint can be viewed as an additional objective function of the BDSP. In fact, some companies already consider certain soft constraints as actual objective functions. For example, some companies consider that the most important objective is to obtain solutions with a given average duration for each type of duty. We identified two groups of soft constraints, those concerning leftovers and those concerning duties. They are discussed in Sections 5.4 and 5.5.

This classification of hard and soft constraints is summarised in table 5.1.

Although soft constraints are handled and often perceived as objective functions, here we view them
5.2. AN INTEGRATED ANALYSIS OF THE BDSP

<table>
<thead>
<tr>
<th>Hard Constraints</th>
<th>Soft Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covering constraints (Section 5.3)</td>
<td>Overcover time for each duty</td>
</tr>
<tr>
<td></td>
<td>Total overcover time</td>
</tr>
<tr>
<td></td>
<td>Number of overcovers</td>
</tr>
<tr>
<td>Concerning leftovers (Section 5.4)</td>
<td>Number of leftovers</td>
</tr>
<tr>
<td></td>
<td>Overall duration of leftovers</td>
</tr>
<tr>
<td></td>
<td>Minimum/maximum duration of leftovers</td>
</tr>
<tr>
<td></td>
<td>Average duration of leftovers</td>
</tr>
<tr>
<td>Concerning duties (Section 5.5)</td>
<td>Average duration of duties</td>
</tr>
<tr>
<td></td>
<td>Average duration of each duty type</td>
</tr>
<tr>
<td></td>
<td>Minimum/maximum percentage of each duty type</td>
</tr>
<tr>
<td></td>
<td>Number of vehicle changes</td>
</tr>
</tbody>
</table>

Table 5.1: Classification of constraints

as different entities (objective functions are described in Section 5.6). Roughly speaking, one of the main objectives of a public transport company in the driver scheduling process is the minimisation of the cost of the solution. However, it is not always clear what is the meaning of this cost. It obviously involves the costs of the duties and the costs of the leftovers (whenever they exist).

Some undesirable features of duties and leftovers can be penalised. However, these penalisations do not necessarily involve direct costs. In particular, concerning leftovers, it is impossible to assign them a cost in advance, because they are going to be handled in a posterior phase, after the basic driver scheduling process, and their direct cost depends in fact on the final solution of several schedules. For example, sometimes we can build a duty merging several leftovers and, in this case, we do not have any additional cost besides the direct cost of the duty. Some other times, we have to consider leftovers as a driver extra time, which is considerably more expensive.

5.2 An integrated analysis of the BDSP

As we have seen in Chapter 3, the majority of the approaches for the BDSP consider that the variables of the problem are directly related with the feasible duties, that constraints are imposed to the set of feasible duties and that the objective functions are focused on the minimisation of the costs assigned to the feasible duties in a solution. Moreover, most of the times, side constraints are discarded or neglected from the model. In fact, many difficulties that commonly arise in the implementation of solutions produced by optimisation techniques are related to the fact that these solutions do not satisfy some side constraints. In this chapter we identify the most significative side constraints of the BDSP arising in Portuguese mass transit companies. In order to take into account these issues, the approach we propose explicitly involves the planner, by allowing him to control those constraints.
CHAPTER 5. A MULTIOBJECTIVE APPROACH TO THE BDSP

In our work we consider a set of variables related to the leftovers. A leftover is a set of contiguous pieces-of-work that do not satisfy the rules imposed to any duty type. In other words, a leftover is an infeasible straight duty. Our approach is based on the experience with several Portuguese transport companies that, by simply using automatic or semi-automatic methods, cannot guarantee that a solution does not contain leftovers. In fact, part of the daily work of the scheduling planners is to manually convert the leftovers into feasible duties, as we have described in Section 2.3.4. Leftovers are an important part of the problem and to exclude them from a solution is, as Ackoff [1] says, "to contract the problem to fit available methods and techniques".

We have seen that most approaches rely on the set partitioning and on the set covering formulations, but neither of them is able to effectively deal with leftovers. In fact, all these approaches consider that a solution with leftovers is an infeasible solution. In a set partitioning solution (see Section 3.3.1), all the pieces-of-work are covered exactly once by a feasible duty. However, for a real world bus driver scheduling problem there is no guarantee that a feasible set partitioning solution exists. Eventually, this issue could be solved if each combination of pieces-of-work is considered as a "dummy" duty with very high costs. However, the size of the problem would increase enormously, and the particular rules and costs of these dummy duties would be difficult to establish.

In a set covering formulation there is always a feasible solution, since a piece-of-work can be covered by more than one feasible duty. The problem is that, in order to cover a given piece-of-work, we may have several other pieces-of-work covered by two or more duties. In practice, a piece-of-work covered by two or more duties, an overcover, means that the driver does part of his daily duty as a passenger of the bus. However, for the BDSP there are no practical reasons for an overcover to occur. In fact, why should the driver start his duty at a given relief point to be a passenger of the bus during a certain period of time? We believe it is more reasonable if he stays at the depot (for a period of time that does not need to match a planned driving work), thus being available to satisfy any eventual needs. Clearly, in airline or train companies, the problem is quite different, since the relief points can be in distant cities or in different countries. In these companies overcovers may be used as an economical way to guarantee that the crews start their duty at the right place and at the right time.

Consider the example shown in Figure 5-1. We can see part of a solution with several overcovers, that resulted from using an approach based on the set covering formulation described in [107]. This example, based on a real problem of Carris, has two relief points, the depot (M_Est) and Alcântara Terra (A_Ter). All the pieces-of-work, except the first and the last ones of each running board (a bus schedule), have a duration equal or smaller than one hour. This solution clearly shows that a set covering formulation is not the most adequate for the BDSP.

Consider for example duty 16: the driver starts his duty at 9h48m at Alcântara Terra and he drives
the bus until 12h27m. Then, "driver 5" starts to work replacing "driver 16". However, "driver 16" remains in the bus as a passenger until 13h16m, when he has a break, leaving the vehicle at Alcântara Terra. Why should he remain in the vehicle during 49 minutes, if he does not drive the bus and if he leaves the vehicle at the same place? He could leave the vehicle at 12h27m and restart his duty at 15h10m (or even at 16h02m). The practical reason for that is that if he had left the bus at 12h27m, duty 16 would not be feasible. In fact, we must add some non-driving work to this duty in order to make it feasible, but it is not necessary to constrain this work to any relief opportunity.

Consider now what happens to the driver assigned to "driver 5". If he effectively drives the bus until 15h55m and is replaced by "driver 9", why must he remain in the vehicle until 19h27m? It would be more reasonable if he goes to the depot or to another place where he is really needed. Then, he could perform the rest of his daily job, not necessarily ending at 19h27m, since his duty is no longer constrained to a relief opportunity. This example contains several other analogous situations.

![Diagram](image.png)

**Figure 5-1:** An example of a solution with overcovers

From our point of view, it is more realistic to consider a relaxed set partitioning solution, such as the presented in Figure 5-2. This solution, which was obtained by manually removing all the overcovers in the previous example, is just an example of how a BDSP solution should be analysed in practice. We have used an arbitrary criterion for removing the duties with overcovers. The solution presented in Figure 5-2 has several pieces-of-work which are not covered by any duty, but it is closer to the planners solution. In fact, he can only cover part of the schedule with feasible duties. The remaining part must be processed manually, by building incomplete duties or assigning extra work to the duties already defined. Frequently, the planner slightly modifies the rules previously defined for a type of duty and builds new duties, considered to be acceptable in practice.

One can argue that overcovers can be easily removed manually, by cutting parts of duties with overcovered pieces-of-work. However, this is not always an easy task, since some pieces-of-work may be covered by several duties and some of them would become too short. Moreover, these short duties are
equivalent to leftovers and are therefore handled as leftovers.

5.2.1 Redefining some fundamental concepts

The concept of duty is perhaps the most important in the BDSP. However, it is a somehow fuzzy concept, since it is not perceived equally by all the actors involved in this process. In our work with Portuguese companies we found some variations in this concept that deserve clarification. Usually, a feasible duty is defined as a set of pieces-of work that satisfy all the constraints of the problem, but this is not always the case. In order to set up a precise framework for our research, and to avoid ambiguity, we present now a set of definitions related to the duty concept.

Definition 5.1 (Duty) A driver duty is the daily job of a single driver. It corresponds to a set of vehicle trips that must be performed by the same driver during a day.

Definition 5.2 (Feasible duty) A feasible duty is a duty that satisfies all the constraints and operational rules defined by the transport company.

Notice that a duty does not always correspond to a feasible duty or, in other words, a duty may violate some of the predefined constraints. In real world instances, a significative percentage of the duties are, in fact, infeasible duties.

Definition 5.3 (Leftover) A leftover is a set of contiguous pieces-of-work from a given vehicle schedule, that are not covered by a feasible duty for a particular BDSP solution.

Notice that, according to this definition, a set of contiguous non-allocated pieces-of-work should be considered as a single leftover. For example, in Figure 5-2, we can observe that there are four sets of uncovered pieces-of-work (four leftovers).

Figure 5-2: Replacing overcovers with leftovers
Definition 5.4 (Problem leftover) A problem leftover is a set of contiguous pieces-of-work from a given vehicle schedule, that cannot be covered by any feasible duty of a BDSP instance.

A problem leftover occurs when there is no feasible duty that covers the corresponding set of pieces-of-work. In a real world problem instance, this situation happens very often, since the rules defined for the duty types may exclude certain combinations of pieces-of-work. The set of problem leftovers can be detected in a preprocessing routine and discarded from the problem instance.

Definition 5.5 (Feasible BDSP solution) A feasible BDSP solution is the set of feasible duties and leftovers for a particular BDSP instance.

Definition 5.6 (BDSP solution) A BDSP solution is composed by the set of duties and leftovers for a particular BDSP instance.

A feasible BDSP solution can be obtained automatically by any procedure based on the set of generated feasible duties. A BDSP solution corresponds to a solution that is actually implemented by the company. Such a solution may contain duties that are not feasible because they do not satisfy the constraints previously imposed to the problem. It can be obtained by manual adjustments to an automatic solution or through heuristics that simulate the manual procedures. Our experience with Portuguese companies has shown that almost all actual solutions contain infeasible duties, either because they are incomplete or because they exceed the values defined for some constraints. Hence, the comparison between a proposed automatic solution and the actual (manual) solution should be performed very carefully, since the automatic solution will never contain infeasible duties.

5.2.2 The complexity of a problem

The size and complexity of a BDSP instance depend mainly on the set of pieces-of-work and on the rules defined for the types of duties. The size of a problem clearly depends on the number of pieces-of-work and on the number of generated feasible duties. The complexity of a problem depends not only on its size, but also on some topological features and on the rules that define the set of feasible duties.

In this section we explore the features that contribute to increasing the size and the complexity of a BDSP instance and we present some strategies to control those features.

Groups of pieces-of-work

A relief point (see Section 2.3) is a node of the network where a driver can be replaced. A relief opportunity is the time at which a bus passes at a relief point. Usually, the depot is a relief point as well as few other nodes of the transport network. However, the number of relief points may vary
significantly from company to company. For example, a small Portuguese transport company we work with (Transportes Urbanos do Barreiro), considers that every bus stop of the network is a relief point. This company operates in the metropolitan area of Lisbon, involving urban and suburban populations, the frequency of the trips being high. Furthermore, the bus driver scheduling process is not divided by routes or groups of routes. Instead, it is handled as a single process involving the whole day operation of the company. Hence, we can see that, if all nodes are relief points, the number of pieces-of-work is extremely large and their duration is very short (few minutes). Even if the number of buses is small and the rules are tight, this problem can be very hard to solve.

On the other extreme, another small transport company (Transportes Urbanos de Aveiro) has a single relief point, the depot. This company faces another type of problems concerning the driver scheduling process, as the driver duties are defined jointly with the bus schedules.

Fortunately, most companies define a small set of relief points, that include the depot and a few other relevant nodes. Even in these companies, we can analyse the relief opportunities, discarding some of them, hopefully not loosing many good solutions. For example, the relief opportunities that are very close to a depot can be discarded or deactivated, since we can assume that a driver would not begin his duty at a depot, driving the bus for 5 or 10 minutes and then, change to another vehicle. The problem presented in Figure 5-2 shows some relief opportunities that were discarded at the beginning and at the end of the bus schedules (these are represented by small vertical marks under the horizontal line that represents the bus schedule).

Duty types

Each company has their own duty types that are mainly characterised by the number of stretches. As we have seen is Chapter 2, all Portuguese companies have two duty types, namely straight duties (with one single stretch) and split duties (with two stretches). Additionally, some companies have three shift duties and others divide the straight duties into day and night duties. Naturally, the larger the number of stretches, the larger the number of feasible duties that are generated for that duty type.

Some of the rules defined for the different duty types are constrained by the legislation and by union agreements. Usually, these rules are related to the maximum working time, the maximum driving time, the minimum length of a mealbreak and the total spreadover. These rules are very rigid in the sense that planners should respect them integrally. However, paradoxically, different companies interpret these rules differently, these interpretations being accepted both by drivers and unions. This was the main reason why we have distinguished a duty from a feasible duty.

Some other rules are not imposed by legislation but by the convenience of the planning process. These rules concern features like the number of vehicle changes, the number and duration of duty interruptions
and the minimum time that a driver should remain in the same vehicle. These rules are not very rigid and the planners control their application as a way to establish trade-offs between flexibility and complexity. In fact, controlling the bounds imposed by these rules may considerably affect the number of feasible duties. For example, increasing the number of duty interruptions exponentially increases the number of feasible duties and, consequently, the dimension and complexity of the problem. When the value of this parameter is high, it is more likely that a solution with less duties can be found. However, companies usually do not want a solution with many small interruptions, since they correspond to paid but unproductive time.

Operational rules

Operational rules define parameters that are applied to all duty types. These rules concern the meal periods (lunch and dinner), the night period and the extra work fraction, and they deeply affect the complexity of the problem. In fact, the amplitude of the meal periods is one of the issues that is more often discussed with unions and drivers. When the amplitude of the meal period is too tight, it is sometimes very difficult to find solutions that cover all the planned trips. These periods are usually settled between the company and the drivers or the unions, and they are not easy to change. Moreover, the time at which the meal break occurs is considered an important indicator of the quality of a duty.

Problem structure

The problem structure results from all the items described above plus the offer structure. The offer structure corresponds to the number of buses in operation throughout the day. As we have seen in Chapter 2, this number is larger in those periods of the day with larger demand. For urban weekdays, these periods correspond to the rush hours, when most people are travelling to or from work. Usually weekend offer is very different from that of weekdays.

The difference between the number of buses in operation at each period of the day affects considerably the complexity of the problem because it becomes difficult to build the required number duty stretches out of the rush hours. The offer structure is not controllable by the planners and the most they can do is to combine groups of routes with different offer structures trying to balance the whole offer structure, but this obviously leads to larger, more complex scheduling problems.

The problem structure, viewed as a combination of different factors, namely the pieces-of-work, the duty types, the operational rules and the offer structure, determines the global complexity of a problem. A small problem with an irregular offer structure may be more difficult to solve than a large problem with a regular offer structure.
5.2.3 Evaluating solutions: a guide to the fundamental issues

The evaluation of a BDSP solution is a very complex task, especially when all decision factors are taken into consideration. This task involves the identification of the main criteria, the assignment of their priorities and the establishment of compromises between desirable and undesirable features of the duties. Traditional mathematical programming approaches, as we have seen in Chapter 3, base the evaluation of a solution on the sum of costs estimates for the individual duties. However, these costs often include two components, one that involves direct costs and another involving indirect costs. Direct costs are relatively easy to compute but, for the indirect costs, it is usually very difficult to obtain accurate estimates.

From the companies point of view, the approach for the evaluation of a BDSP solution is completely different. Planners look at a solution as a whole, and not as a sum of parts. Naturally, each duty is analysed individually, but there are certain criteria that involve subsets of duties with similar features.

The multiobjective nature of this problem will be analysed in detail in the next sections. In this section we aim at alerting the reader for the relevance of this topic, which is key in our work and motivates our main contributions. In fact, the complexity of the BDSP arises from its combinatorial and multiobjective features. If the combinatorial nature of the problem has been exhaustively studied (see Chapter 3), we cannot say the same about its multiobjective aspects. The gap between the model and reality is not easy to overcome and the solutions proposed by the different techniques are very difficult to implement in practice.

In our work with Portuguese companies, we have identified some of the criteria that guide the planners in the evaluation of a BDSP solution. Some companies only consider a subset of these criteria and the "weight" assigned to each criterion is different from company to company. The most important criteria used by Portuguese companies to evaluate a bus driver schedule are described below.

Number of drivers required to perform all the trips

This criterion does not only include the duties, but also the leftovers in the solution. It is not always easy to determine the number of drivers that are needed when a solution includes some leftovers, even if the number and the duration of the leftovers are important issues to be considered. Sometimes, a leftover can be assigned to a driver as extra work, but some other times it can be assigned to a driver as an incomplete (and infeasible) duty or it can be combined with another leftover (from another group of lines) to form a feasible duty.

Number of leftovers

This criterion is related to the previous one but, if considered separately, the number of leftovers should be as small as possible since they represent additional costs to the company. However, reducing
the number of leftovers often imply increasing the number of duties. The trade-off between the number of duties and the number of leftovers is not easy to establish and planners want to know how many additional duties are needed in order to reduce the number of leftovers.

**Total duration of leftovers**

Combined with the previous criterion, the total duration of the leftovers is an important indicator of the quality of a solution. Some companies prefer a solution with a large number of short leftovers instead of a solution a small number of long leftovers. This fact is related to each company's practice regarding leftovers. If each leftover has a duration of 15 or 30 minutes, some companies will assign them as extra work, even if the final solution turns out to be composed by a large number of duties with extra work. However, other companies will prefer to assign the leftovers to incomplete duties and, in this case, larger leftovers (e.g. 3 hours long) will be preferred.

**Average duration of leftovers**

This criterion is a combination of the two previous criteria. Some companies would like to establish an average duration of the leftovers in order to guarantee that a good trade-off between their number and their duration would be achieved.

**Average duration for each duty type**

This is one of the most important criterion used by the planners for the evaluation of a solution. In some companies, several other criteria (including the number of duties) are sacrificed in order to achieve the desired duration for each duty type. Although the individual length of each duty can attain a large value, the average duration of a given group of duties of the same type should be bounded by pre-defined smaller values. This situation occurs in companies where, in the rostering phase, the rotation of the drivers is performed in groups of duties of the same type. The average work of each driver is not calculated on a daily basis, but rather for the time horizon associated to the rotation group. This horizon can vary from five or seven days to several weeks.

**Average duration of duties**

This criterion is a simplified version of the previous one, since the average duration is computed for the total set of duties, independently of their type. It is used by companies where the rotation groups are formed according to the average duration of the duties in the group (that may include duties of different types).
Percentage of each duty type

Some companies want the percentage of some duty types not to exceed a given value. This situation occurs when some types of duties are not desirable by the drivers, although there is no legal constraints imposing it. However, the planners know that a solution that exceeds a given percentage for some duty types would be very difficult to implement, because it would decrease the global quality of the drivers work. On the contrary, some other companies want to impose a minimum percentage for some duty types. Usually, in these companies, the drivers and the company clearly prefer some duty types and the solutions should reflect this common preference.

Percentage of duty interruptions

Most companies allow the duties to have small interruptions paid by the company as effective work. The duration of these interruptions should be smaller than the minimum duration of the duty break and, it often implies a vehicle change. The number of small interruptions allowed for each duty varies with the company, since there is no legal restriction on this matter. As these interruptions correspond to unproductive working time, companies try to minimise their occurrence. However, allowing small interruptions in the duties increases the number of feasible duties and often provides solutions with less duties and less leftovers. Hence, this is a very important criterion, used by the planners to evaluate solutions, although the correct trade-off between the number of drivers needed and the percentage of small interruptions in a solution should to be carefully assessed.

Average number of duties assigned to each vehicle

All companies agree that a vehicle should be assigned to the minimum number of drivers, preferably not exceeding two drivers. The reason is that the maintenance costs of the vehicles are usually considered to be significantly reduced when a small number of drivers are assigned to each vehicle. In this case, the drivers responsible for accidents or mechanical problems can be easily identified and, usually, they are more careful with the vehicles assigned to them. Naturally, it is not always possible to guarantee that each bus is assigned to the minimum number of drivers, and we have noticed that this concern is more important in smaller companies.

Finally, it should be noted that the direct economic cost of the duties was not included in the above list of criteria because it is not explicitly used to evaluate a solution. In fact, some of those criteria can be included in the individual "cost" of the duties, thus reflecting a penalisation on the occurrence of features that are not desired. This approach is frequently adopted by the traditional techniques, although it cannot be easily applied when the criteria involve the evaluation of the solution as a whole and not simply the addition of values from each single duty.
5.3 Hard constraints

From the planner's point of view, the concept of a feasible schedule is somehow fuzzy and subjective, as we have tried to show in the previous section. However, in order to achieve an objective evaluation of a solution, we must define precisely what constraints should be taken into account or, in other words, which are the main features of a feasible BDSP solution. The process that guarantees that the hard constraints are never violated will be described in Chapter 6. In this section we intend to clarify their purpose and meaning from the planner's perspective.

In our approach the planner either chooses hard partitioning constraints or hard covering constraints. It should be noted that users are usually very reluctant to use the hard covering constraints. This attitude probably results from previous experiences with other approaches, in which the number of overcovers (or overcover time) in the final solutions is too large. As a consequence, these solutions implied a lot of manual changes and decision making. Moreover, allowing the existence of overcovers in the solutions will increase the number of feasible solutions and the complexity of the search space. Hence, we have decided to impose a limit to the overcovers allowed in the solutions, in order to reduce the large size of the search space. This limit is an external parameter, fully controlled by the users. The value of this parameter depends on the specific problem instance and on the company's attitude concerning overcovers.

Next, we introduce some notation that will be used in the following sections.

Notation 5.1 For each particular problem, consider the following sets:

- $I$ is the set of the pieces-of-work, $I = \{i_1, i_2, \ldots, i_m\}$.
- $J$ is the set of generated duties, $J = \{j_1, j_2, \ldots, j_n\}$.
- $S$ is the set of duties in a solution, $S \subseteq J$.
- $I_S$ is the set of pieces-of-work covered by the duties in $S$, $I_S \subseteq I$.
- $I_j$ is the set of pieces-of-work covered by duty $j$, $I_j \subseteq I$.
- $I_S^c$ is the set of the overcovered pieces-of-work in solution $S$.

Consider also the following functions:

- $\text{dur}(i)$ returns the duration of piece-of-work $i$.
- $\text{bus}(i)$ returns the running board that contains piece-of-work $i$.
- $\text{type}(j)$ returns the type of duty $j$.
- $\text{start}(i)$ returns the start time of piece-of-work $i$.
- $\text{finish}(i)$ returns the finish time of piece-of-work $i$.

Hard Partitioning Constraints

Hard partitioning constraints guarantee that in a solution each piece-of work is covered at most by one duty ($I_S^c = \emptyset$). These constraints are very restrictive, since the resulting number of feasible solutions
is much smaller than the number of solutions for the BDSP with hard covering constraints. However, we can not guarantee that all pieces-of-work are covered, hence some undercover may exist.

**Hard Covering Constraints**

These constraints allow the existence of overcovers, although their number or duration can be limited by the planner. Due to these constraints, allowing overcovers does not imply that some undercover cannot happen. If the allowed number or duration of overcovers is high enough, the algorithm can produce a pure covering solution (with no uncovered pieces-of-work). On the contrary, if the maximum number of overcovers is too small, we may obtain a solution for which it is impossible to cover a particular piece-of-work without violating the maximum number of overcovers.

We have identified four groups of hard constraints involving overcovers.

**Maximum overcover time for each duty**

These constraints allow the existence of overcovers as long as they do not exceed a fixed total duration for each duty (see Expression 5.1). This time limit for the overcovers is settled by the user (the parameter \texttt{DUR\_MAX\_OVERCOVER}) and it must be satisfied by all duties of the candidate solutions found by the algorithm.

\[
\forall j \in S, \overline{\text{dur}}(j, S) = \sum_{i \in f \in S} \text{dur}(i) \leq \text{DUR\_MAX\_OVERCOVER} \tag{5.1}
\]

For example, an "overcover duration of 2 hours" means that no overcovered duty can have more than 2 hours "shared" with other duties. It should be noticed that the overcover time can be distributed by several duties. In Figure 5-3 we present an example that illustrates the result of applying this constraint. The overcover time of duty 2 is distributed by duties 1 and 3 and it should not be larger than a given value (in this case, 2 hours).

![Figure 5-3: Example of a solution where a constraint imposing a maximum overcover time for each duty has been imposed](image-url)
5.3. HARD CONSTRAINTS

Maximum overcover time for each pair of duties

This constraint concern the maximum overcover time for each particular pair of duties (see Expression 5.2).

\[ \forall j_1, j_2 \in S, \text{dur}(j_1, j_2) = \sum_{i \in I_{j_1} \cap I_{j_2}} \text{dur} (i) \leq \text{DUR\_MAX\_OVERCOVER} \quad (5.2) \]

In Figure 5-4 we present an example of this situation: duty 1 and duty 2 have two pieces-of-work in common, with a total duration of 1h35 and duty 2 has one piece-of-work in common with duty 3, with a total duration of 1h20. The overcover time of each pair of duties (duties 1 and 2 and duties 1 and 3) is less than the maximum allowed, but the total overcover time of duty 2 is more than 2 hours.

![Figure 5-4: Example of a solution where constraints that impose a maximum overcover time for each pair of duties were imposed](image)

These constraints are a good alternative to the previous ones, since they are much easier and faster to implement and check. In fact, we do not need to search all the pieces-of-work of a duty to compute its overcover time. They can be useful when we want to drastically reduce the number of uncovered pieces-of-work, because they allow a larger number of feasible covering solutions. However, they can also produce solutions with undesirable features. For example, we can have solutions with duties that are almost completely filled with overcovers while satisfying these constraints for each pair of duties.

This drawback led us to choose the first type of constraints for modelling purposes. Therefore, in the forthcoming sections, whenever we refer to the maximum overcover time, we will be talking about the maximum overcover time for each duty.

Maximum total overcover time

Additionally, we can impose an upper limit to the total overcover time in the solution (see Expression 5.3).

\[ \overline{\text{dur}}(S) = \sum_{i \in I} \text{dur} (i) \leq \text{TOTAL\_DUR\_MAX\_OVERCOVER} \quad (5.3) \]
For example, we can impose that no overcover can exceed one hour for each duty and, at the same time, that the solution cannot have more than five hours of overcover time. This constraint is very useful when we want to allow the existence of some overcovers, but we also want to bind the total overcover time in the solution.

**Maximum number of overcovers**

This constraint can be used alone, or combined with one or several of the previous constraints. Used alone, it simply refers to the number of overcovers of each duty, and does not take into account the time involved in the overcovers (see Expression 5.4). It is set by the user through the parameter `NUM_MAX_OVERCOVER` and it is very easy to implement and test, since it does not need to use the time information about the pieces-of-work.

\[ |\tilde{T}_s| \leq \text{NUM\_MAX\_OVERCOVER} \quad (5.4) \]

Although this constraint has been used in some way by other authors to reduce the number of overcovers [68], it does not seem to fit into the BDSP instances of the Portuguese companies we have been working with. The minimisation of the number of overcovers alone could be interesting if the duration of the pieces-of-work was approximately the same (for example, if the bus schedules involved are very regular and involve few relief points). In fact, if the pieces-of-work have similar durations, minimising the number of overcovers in the solution is analogous to minimising the overcover time, although we must know in advance the average duration of the pieces-of-work in order to fix a limit on the number of overcovers. However, most of the problems we have solved have several relief points and pieces-of-work with very different durations.

However, it can be interesting to use this constraint combined with other constraints involving overcovers. For example, we can impose that an overcover cannot exceed one hour for each duty and that we do not allow more than three overcovers in the solution. Furthermore, we can impose that each duty cannot exceed one hour of overcover time, that a solution cannot have overcovers that last more than three hours and, at the same time, that this solution cannot have more than five overcovers.

### 5.4 Soft constraints involving leftovers

As we have already discussed, leftovers are unavoidable although undesirable in a solution. Generally, planners manually manipulate solutions in order to, at least, control some attributes of the leftovers. In this section we propose two different groups of soft constraints, that help planners in controlling these undesirable features when an automatic solution is produced.
5.4.1 Number and overall duration of leftovers

The number of leftovers in a solution, that we want to minimise, can be viewed as a soft constraint. In our approach, each leftover is not a single uncovered piece-of-work, but a set of contiguous uncovered pieces-of-work. Two pieces-of-work are contiguous if they belong to the same running board and one of them finishes at the same relief point and at the same time the other one starts. Hence, we can define a leftover \( l_S \) of a solution \( S \) as a sorted tuple of uncovered pieces-of-work satisfying the following conditions:

\[
l_S = \left\{ (i_1, i_2, \ldots, i_k) \in (I \setminus I_S)^k : \text{bus}(i_1) = \ldots = \text{bus}(i_k) \land \text{finish}(i_{j-1}) = \text{start}(i_j), j = 2, \ldots, k \right\}
\]

(5.5)

However, in order to avoid the occurrence of long leftovers, we consider that each leftover that lasts more than a predefined value (usually, the maximum driving time) is divided into two leftovers. The underlying idea is that if a leftover lasts more than the maximum driving time, the company has to assign two drivers to it. The violation of this soft constraint is computed according to Expression 5.6, where \( \text{dur}(l_S) = \sum_{i_k} \text{dur}(i_k) \).

\[
eval_1(S) = \sum_{l_S} \left[ \frac{\text{dur}(l_S)}{\text{MAX\_DRIVING\_TIME}} \right]
\]

(5.6)

As an alternative, or combined with this criterion, we can minimise the overall duration of the leftovers. For this purpose we add up the duration of all the uncovered pieces-of-work (see Expression 5.7).

\[
eval_2(S) = \sum_{l_S} \text{dur}(l) = \sum_{i \in I \setminus I_S} \text{dur}(i)
\]

(5.7)

These two criteria contribute in different ways to the quality of a solution, depending on the companies point of view. Some companies want to obtain a minimum number of leftovers, because they will have to assign the corresponding working time to drivers in a subsequent phase of the planning process (see Chapter 2). Some of these companies do not care about the overall duration of the leftovers, since they get only a few. Some other companies consider that the uncovered time should be as small as possible. These companies do not care if the solution includes a lot of small uncovered pieces-of-work if the overall duration of the uncovered schedule is small. These extreme attitudes result from considering the two criteria alone and they can lead to unbalanced solutions: the first criterion alone can produce solutions with a small number of leftovers, but with a considerable duration of uncovered work. On the contrary, considering only the second criterion, we can get solutions with a small percentage of the schedule left
uncovered, but with a large number of small leftovers.

Therefore, in practice, a reasonable approach for most companies would be to consider these two criteria altogether instead of considering just one of them. This approach is illustrated in Example 5.1.

**Example 5.1 Number/overall duration of leftovers**

Table 5.2 illustrates the application of these soft constraints to a small problem from Carris. Solution 1 was obtained by considering only the minimisation of the number of leftovers. Solution 2 was obtained by considering the minimisation of the overall duration of the leftovers alone, and in Solution 3 both soft constraints have been considered.

<table>
<thead>
<tr>
<th>Sol.</th>
<th>Soft constraints</th>
<th>left.</th>
<th>tot.dur.left</th>
<th>duties</th>
<th>%cov.sch.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>n° left.</td>
<td>2</td>
<td>6:26</td>
<td>21</td>
<td>96.14</td>
</tr>
<tr>
<td>2</td>
<td>left. dur.</td>
<td>5</td>
<td>3:02</td>
<td>21</td>
<td>98.09</td>
</tr>
<tr>
<td>3</td>
<td>n° left.+left. dur.</td>
<td>2</td>
<td>4:48</td>
<td>21</td>
<td>97.99</td>
</tr>
</tbody>
</table>

*left: number of leftovers; duties: number of duties; tot.dur.left: total duration of leftovers; % cov.sch: covered schedule rate*

Table 5.2: Number/overall duration of leftovers example

Considering both soft constraints (Solution 3) we can effectively reduce the number and the overall duration of the leftovers.

5.4.2 Controlling the duration of a leftover

We have implemented another subset of soft constraints whose main objective is to give the user additional control on the duration of leftovers. If we pose to any user the following question: "Suppose that you cannot avoid leftovers in your solution. Which are the main features they should have?", we will get many different answers, some of them fuzzy and subjective, but most of all will include these kind of comments: "we do not want leftovers with more than two hours", or "each leftover should have at least one hour", or "the average duration of the leftovers should be around 2h30m". According to the planners, the leftovers that satisfy certain features will be easier to assign to drivers as extra time or as incomplete duties. Certain companies prefer a solution with more duties and in which the leftovers do not exceed a given minimum duration than a solution with less duties and in which there exists one or more leftovers that exceed that minimum duration.

In order to control this kind of features we propose two alternatives:

1. to control the minimum and/or the maximum durations of leftovers

2. to control the average duration of the leftovers.
5.4. SOFT CONSTRAINTS INVOLVING LEFTOVERS

For each alternative we need to introduce the desired values and the corresponding degree of penalisation. In our work, we have considered three levels for this degree of penalisation, with the following meaning:

None: do not penalise the constraint violation;

Normal: the constraint violation is penalised;

Strong: the constraint violation is severely penalised.

We have chosen to face the users with this simple ranking procedure, instead of a continuous scale, because it is not easy for them to fully understand the effect of a control parameter in an objective function.

Controlling the minimum and/or the maximum duration of the leftovers

Let \( MINDUR \) be the minimum duration of a leftover and \( MAXDUR \) the maximum duration of a leftover. We compute the deviation from the minimum and maximum duration of each leftover according to Expressions 5.8 and 5.9 respectively.

\[
\text{devMin}(l_s) = \begin{cases} 
  \text{MINDUR} - \text{dur} (l_s), & \text{dur} (l_s) < \text{MINDUR} \\
  0, & \text{otherwise}
\end{cases} 
\] (5.8)

\[
\text{devMax}(l_s) = \begin{cases} 
  \text{MAXDUR} - \text{dur} (l_s), & \text{dur} (l_s) > \text{MAXDUR} \\
  0, & \text{otherwise}
\end{cases} 
\] (5.9)

The leftovers which duration is less than a given minimum value are penalised according to Expression 5.10, while the leftovers which duration is higher than a given maximum value are penalised according to Expression 5.11. The degree of penalisation is expressed by \( \text{penalMin}, \text{penalMax} \in \{ \text{None}, \text{Normal}, \text{Strong} \} \)

\[
\text{eval}_{3.1} (S) = \text{penalMin} \times \sum_{l_s} \text{devMin}(l_s) 
\] (5.10)

\[
\text{eval}_{3.2} (S) = \text{penalMax} \times \sum_{l_s} \text{devMax}(l_s) 
\] (5.11)

The total violation of this soft constraint is computed according to Expression 5.12

\[
\text{eval}_3 (S) = \text{eval}_{3.1} (S) + \text{eval}_{3.2} (S) , \text{where} 
\] (5.12)
Controlling the average duration of the leftovers

As an alternative, the planner may choose to penalise the solutions which average duration of leftovers is far from a given value ($AVGDUR$). In this case, we compute the deviation in the solution $S$ from this value according to Expression 5.13.

$$\text{devAvg} = \left| AVGDUR - \frac{\sum_{l_S} \text{dur} (l_S)}{\sum_{l_S} \frac{\text{dur} (l_S)}{\text{MAX\_DRIVING\_TIME}}} \right|$$  \hspace{1cm} (5.13)

The violation of this soft constraint is computed according to Expression 5.14, where $\text{penalAvg}$ is the degree of penalisation ($\text{penalAvg} \in \{\text{None}, \text{Normal}, \text{Strong}\}$).

$$\text{eval}_3 (S) = \text{penalAvg} \times \text{dur} \times \text{devAvg}$$  \hspace{1cm} (5.14)

Note that these two soft constraints cannot be chosen at the same time or, in other words, the planner either chooses to control the minimum/maximum duration of the leftovers or the average duration of the leftovers.

Moreover, these soft constraints must be combined with one or both of the previous set of constraints, thus increasing the user control on some features of the solutions. Alone, these constraints do not lead to interesting solutions.

Consider Example 5.2, using the same problem instance of the previous example:

Example 5.2 Minimum, maximum and average duration of leftovers

In this example we illustrate the effect of considering soft constraints that control the duration of the leftovers. We present three distinct solutions that show how this group of soft constraints can effectively handle different planning scenarios.

Solution 1 corresponds to Solution 3 of the previous example, when the minimisation of the number of leftovers and of the overall duration of leftovers have been both considered. Solution 2 was obtained by adding limits to the minimum and maximum duration of leftovers ($\text{min:1h00m, max: 2h00m}$). Solution 3 was obtained by adding an average duration of 1h30m to the leftovers.

We can observe that Solution 2 presents more leftovers and a larger overall duration than the other two solutions, but the leftovers satisfy the condition imposed. Solution 3 has effectively reduced the average duration of the leftovers without significantly increasing the overall duration of the leftovers.
Table 5.3: Controling the duration of leftovers

In conclusion, the simultaneous consideration of different types of soft constraints involving leftovers gives the planner more control on some of their features. He can quickly try and analyse different alternatives and choose that solution representing the best trade-off from the company’s point of view.

5.5 Soft constraints on duties

Traditionally, when we do not want to include into the solution the duties that have some particular feature, we use a penalty, that is added to the duty cost. However, it is often important to evaluate not only the single duties, but rather the solution as a whole. For example, if a particular company wants to impose that no more than 30% of the duties in the schedule would be straight duties, this constraint involves the whole solution.

The soft constraints presented in this section try to incorporate into the solution some attributes that cannot be included in the individual duty cost, imposing restrictions to the whole set of duties selected for a solution. We have classified these constraints into two subgroups: those that involve the average duration of the duties in the solution, and those that involve the duty types in the solution.

5.5.1 Constraints on the average duration of duties

In general, the rules imposed by legislation allow a wide range of values for the duration of a duty (varying for example between 5h00m and 8h00m). However, companies usually want balanced schedules, i.e., different schedules should have similar average durations of duties, in order to simplify the rostering process. Usually, in the rostering phase, drivers will rotate (daily or weekly) through the duties of a schedule and, at the end of the week, they must average a certain amount (for example, 7h00m) of daily work. For this reason, if the average duration of the duties in the schedule is near to the desired value it will be easier for the planner to build balanced rosters for all drivers.

We have added a soft constraint that penalises the schedules whose average duties duration is far from the desired average value (AVDUR). First, we compute the target number of duties, by dividing the total duration of the schedule by the desired average value. The target number of duties corresponds to
the number of duties in the schedule if the desired mean duration is achieved. The violation of this soft constraints is calculated according to Expression 5.15, by subtracting the target number of duties from the total number of duties in $S$.

$$\text{eval}_4(S) = |S| - \frac{\sum_{i \in I} \text{dur}(i)}{\text{AVDUR}}$$  \hspace{1cm} (5.15)

This soft constraint can be used together with other soft constraints or objective functions. In certain companies it acts almost as an hard constraint, in the sense that any solution that exceeds the average duration of the duties is not implemented. Other companies use this soft constraint simply as a guideline along the schedule construction.

Example 5.3, taken from a real problem, illustrates the effect of the application of this constraint.

Example 5.3 Average duration of duties

The first solution (Solution 0) in Table 5.4 was obtained by trying to minimise the number of duties and the number of leftovers and it is composed by 52 duties and no leftovers. Suppose that, in this particular company, the average duration of the duties should be at most 6h15m. Solution 0 would be difficult to implement, since the average duration of the duties is 6h30m. Hence, we have added a soft constraint that imposes the value 6h15m to the average duration of duties and we obtained Solution 1, with one more duty but with an average duration of duties that satisfies the company objective (6h14m). Note that both solutions have no leftovers.

<table>
<thead>
<tr>
<th>Sol.</th>
<th>additional constraints</th>
<th>left.</th>
<th>tot.dur.left.</th>
<th>duties</th>
<th>%cov.sch.</th>
<th>av.dur.duties</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>52</td>
<td>100</td>
<td>6:30</td>
</tr>
<tr>
<td>1</td>
<td>av.dur.duties</td>
<td>0</td>
<td>0</td>
<td>53</td>
<td>100</td>
<td>6:14</td>
</tr>
</tbody>
</table>

left: number of leftovers; duties: number of duties; tot.dur.left: total duration of leftovers \%
cov.sch: covered schedule rate; av.dur.duties: average duration of duties

Table 5.4: Average duration of duties

5.5.2 Constraints on duty types

With this group of constraints we try to impose some restrictions to the duty types allowed in the solution. A simple procedure consists in adding to each individual duty a penalty that reflects the undesirability of its type. Hence, if we minimise the global costs of the duties, we could expect to obtain solutions that contain the duty types we prefer. However, sometimes this is not enough. We have identified two other situations that need particular attention: when we want to impose an average duration for each duty type and when we want to limit the number or the percentage of each duty type in the solution.
Average duration for each duty type

Instead of imposing an average duration for all the duties in the schedule, as presented before, some companies want to impose an average duration to each duty type. Usually, this situation arises when the planner anticipates some problems that may occur in the rostering phase, for example, when we have to replace a duty in a roster by a similar one of the same type (i.e., a compatible duty). If all schedules present similar average values for each duty type, one may expect to have a higher number of compatible duties in the rostering phase. A similar situation occurs for special duty types (for example, night duties or long duties) that individually can have long durations but, on average, are constrained to quite smaller durations.

With this soft constraint we allow the user to define the average duration for each duty type and how the deviation should be penalised. In Expression 5.16a, we present how this violation is computed.

The value of $penDutyType(t)$ (5.16b) is calculated for each duty type, multiplying a weighted duty minute cost ($DTymCost(t, S)$) by the deviation $dev(t, S)$ (5.16d) from the desired average duration of the duty type, $avDur(t)$.

The meaning of the weighted duty minute cost (5.16c) is that the cost of each minute of a duty is higher if the average duration of the corresponding duty type in the solution ($avDur(t, S)$) is smaller than the desired one, and lower in the opposite case.

For each duty type, $penDutyType(t)$ is computed according to the value of a parameter, $PENDEV(t)$. This parameter takes the value 1 if we want to penalise the positive deviation and $-1$ if we want to penalise the negative deviation. If, for a particular type of duty $t$, we do not want to penalise the deviation from the desired mean duration the value of $PENDEV(t)$ is 0.

\[
eval_5(S) = \sum_{t \in T} penDutyType(t) \tag{5.16a}
\]

\[
penDutyType(t) = \begin{cases} 
DTymCost(t, S) \times |dev(t, S)|, & PENDEV(t) \times dev(t, S) < 0 \\
0, & otherwise
\end{cases} \tag{5.16b}
\]

\[
DTymCost(t, S) = \frac{avDur(t)}{avDur(t, S)} \times DTymCost \tag{5.16c}
\]

\[
dev(t, S) = avDur(t) - avDur(t, S) \tag{5.16d}
\]

where $T$ is the set of duty types and $S$ is the set of duties in the solution.

Example 5.4 shows how this constraint affects the solutions for a real problem.
Example 5.4 Average duration for each duty type

Assume we have three duty types: straight duties, split duties and long break duties. Assume also that, for each duty type, a planner has established the following average lengths: straight duties: 6h00m; split duties: 8h00m; long break duties: 8h00m. For comparison purposes, we present Solution 0, obtained without any additional constraints beyond the number/overall duration of the leftovers. Solution 1 is the solution obtained with the addition of a constraint for the average length for each duty type.

<table>
<thead>
<tr>
<th>Sol.</th>
<th>additional constraints</th>
<th>left.</th>
<th>tot.dur.left.</th>
<th>duties</th>
<th>%cov.sch.</th>
<th>av.dur.duties</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>3</td>
<td>5:39</td>
<td>56</td>
<td>95,34</td>
<td>6:40</td>
</tr>
<tr>
<td>1</td>
<td>av.dur.type</td>
<td>4</td>
<td>8:08</td>
<td>54</td>
<td>97,89</td>
<td>7:03</td>
</tr>
</tbody>
</table>

left: number of leftovers; tot.dur.left: total duration of leftovers; duties: number of duties; %cov.sch: covered schedule rate; av.dur.duties: average duration of duties; av.dur.type: average duration for each type of duty

Table 5.5: Average duration for each duty type

<table>
<thead>
<tr>
<th>Sol.</th>
<th>Straight duties</th>
<th>Split duties</th>
<th>long break duties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>duration</td>
<td>%</td>
<td>duration</td>
</tr>
<tr>
<td>0</td>
<td>5h50</td>
<td>39,29</td>
<td>7h32</td>
</tr>
<tr>
<td>1</td>
<td>6h00</td>
<td>27,78</td>
<td>7h44</td>
</tr>
</tbody>
</table>

Table 5.6: Duration and percentages of each type of duty for the solutions obtained

As shown in Table 5.5, although Solution 1 presents one more leftover (4) than Solution 0 (3), it also contains less duties (54 instead of 56). Moreover, concerning the average duration of each type of duty presented in Table 5.6, Solution 1 is probably closer to the goals expressed by our imaginary planner than Solution 0.

Minimum and maximum percentage of each duty type

These constraints impose lower and upper limits to the percentages of each duty type in the solution and have been implemented as a combination between soft and hard constraints. On the contrary to the other soft constraints, the procedure that ensures that these constraints are satisfied is embedded in the genetic algorithm operators (see Section 6.4.1). Since we cannot guarantee that a solution satisfying a particular combination of percentage values exists, we allow the existence of solutions that violate these constraints. However, we present to the user only those solutions that satisfy all the constraints.

Example 5.5 presents two situations that illustrate how solutions can be affected by these constraints.

Example 5.5 Minimum and maximum percentages for each type of duty

In this example we have imposed upper limits to certain duty types. Once again, Solution 0 was obtained with no additional constraints, in order to be compared with Solutions 1 and 2.
5.5. SOFT CONSTRAINTS ON DUTIES

Solution 1: no more than 50% of split duties and no more than 30% of the long break duties; no limit to the straight duties.

Solution 2: no more than 30% of straight duties and no more than 40% of long break duties; no limit to the split duties.

<table>
<thead>
<tr>
<th>Sol.</th>
<th>Additional constraints</th>
<th>left.</th>
<th>tot.dur.left.</th>
<th>duties</th>
<th>% cov.sch.</th>
<th>av.dur.duties</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>3</td>
<td>5:39</td>
<td>56</td>
<td>98.54</td>
<td>6:40</td>
</tr>
<tr>
<td>1</td>
<td>Case 1</td>
<td>4</td>
<td>6:24</td>
<td>55</td>
<td>98.35</td>
<td>6:57</td>
</tr>
<tr>
<td>2</td>
<td>Case 2</td>
<td>4</td>
<td>6:00</td>
<td>53</td>
<td>97.94</td>
<td>7:11</td>
</tr>
</tbody>
</table>

left: number of leftovers; tot.dur.left: total duration of leftovers; duties: number of duties
% cov.sch: covered schedule rate; av.dur.duties: average duration of duties

Table 5.7: Percentages for each type of duty

<table>
<thead>
<tr>
<th>Sol.</th>
<th>Straight duties</th>
<th>Split duties</th>
<th>Long break duties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>duration</td>
<td>%</td>
<td>duration</td>
</tr>
<tr>
<td>0</td>
<td>5h50</td>
<td>40</td>
<td>7h32</td>
</tr>
<tr>
<td>1</td>
<td>6h02</td>
<td>35</td>
<td>7h37</td>
</tr>
<tr>
<td>2</td>
<td>5h49</td>
<td>19</td>
<td>7h37</td>
</tr>
</tbody>
</table>

Table 5.8: Duration and percentages of each type of duty for the solutions obtained

Analyzing the tables above, we can see that Solution 0 is not a feasible solution if the additional constraints are considered (both in the case of Solution 1 and in the case of Solution 2). In fact, it contains 34% long break duties and 40% of straight duties. Both Solutions 1 and 2 contain less duties than Solution 0 and just one more leftover.

Assigning preferences to duty types

In many companies the planners do not want to assign percentages to each duty type. However, they know exactly which duty types they prefer and the duty types they would exclude from the solution, if possible. Hence, we have added a soft constraint that penalises the solutions that include the duty types less preferred. The preference of a duty types is defined as an integer value, ranging from 1 to the number of duty types (T). The lowest values correspond to the highest preferences. The violation of this soft constraints is calculated according to Expression 5.17.

\[
eval_a(S) = \sum_{t \in T} (\text{pref}(t) \times |S_t|)
\]

(5.17)

where \( t \in T, \text{pref}(t) \in P = \{1, 2, \ldots, T\} \) and \( S_t = \{j \in S : \text{type}(j) = t\} \).

Example 5.6 (Assigning preferences to duty types) In this example we have assigned preferences to the duty types. Solution 0 was obtained with no additional constraints, in order to be compared with
Solution 1. This solution was obtained after assigning the following preferences to the duty types: to straight duties: 1, split duties: 2, night duties: 3, long break duties: 4)

<table>
<thead>
<tr>
<th>Sol.</th>
<th>Additional constraints</th>
<th>Left.</th>
<th>tot. dur. left.</th>
<th>duties</th>
<th>av. dur. duties</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>1</td>
<td>4:00</td>
<td>61</td>
<td>6:14</td>
</tr>
<tr>
<td>1</td>
<td>Preferences of duty types</td>
<td>1</td>
<td>4:55</td>
<td>61</td>
<td>6:13</td>
</tr>
</tbody>
</table>

Table 5.9: Assigning preferences to duty types

<table>
<thead>
<tr>
<th>Sol.</th>
<th>N. Straight duties</th>
<th>N. Split duties</th>
<th>N. long break duties</th>
<th>N. Night duties</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>99</td>
<td>4</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>42</td>
<td>10</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 5.10: Number of duties for each duty type for the solutions obtained

5.5.3 Constraints on the number of vehicle changes

Some companies do not want their drivers to change of vehicle frequently in the same day as they state that this will result in better maintained buses, with more responsible drivers. With this soft constraint we try to minimise the number of bus changes in a solution (see Expression 5.18), simply by counting the number of vehicles involved in each duty and the number of leftovers in the solution.

$$eval_7(S) = \sum_{j \in S} num_{bus}(j) + \sum_{i \in L} \left\lceil \frac{dur(i)}{MAX\_DRIVING\_TIME} \right\rceil$$

where

$$num_{bus}(j) = |\{bus(i) : i \in L_j\}|$$

is the number of vehicles involved in duty j.

Example 5.7 illustrates the influence of this soft constraint in a solution.

Example 5.7 Number of vehicle changes

Consider the solution partially presented in Figure 5-5, with two vehicles, each one needing four different duties. In order to perform all the trips, we need six duties. Consider now the solution for the same problem presented in Figure 5-6, obtained with the additional soft constraint that tries to minimise the number of vehicle changes. We can see that the number of duties assigned to each vehicle goes from four to three, without increasing the total number of duties (6). In Table 5.11, we present the global results for this problem. Solution 0 was obtained without this soft constraint while in Solution 1 this constraint was added. We can see that number of vehicle changes has been considerably reduced.
5.6. **OBJECTIVE FUNCTIONS**

![Diagram](image)

*Figure 5-5: Example of a solution with no additional soft constraints*

![Diagram](image)

*Figure 5-6: Example of a solution where a soft constraint to minimise the number of vehicle changes was added*

### 5.6 Objective functions

In the previous sections we have presented a set of soft constraints, considered here as objective functions in an attempt to minimise their violation. In some problems, like in the examples presented above, all objective functions represent soft constraints.

In this section, we present the "real" objective functions, that can be combined with any group of soft constraints. We suggest a procedure to evaluate the cost of a solution and we propose two different objective functions based on this cost: the minimisation of the total cost of the duties and the minimisation of the number of the duties, which is a special case of the former.

#### 5.6.1 Evaluating the cost of a duty

The minimisation of the total cost of the duties is the most common objective function used for the BDSP, although there is often some confusion between the direct cost of a duty and the penalties assigned to some of its features. This confusion is the main reason why single objective approaches usually have to face so many implementation obstacles, since the final solution (supposed to be of minimum cost) frequently does not fit the planner's expectations. In fact, when the cost of a duty includes several penalty terms, we are in practice dealing with another multiobjective problem, since we want to minimise several undesired features but some of them may be conflicting. However, the planner does not perceive the multiobjective nature of the cost function when he gets the final solution, which may therefore be viewed as being unsatisfactory.
From our point of view, the evaluation of each duty cost should be based on features that can be easily computed and measured in money, and that involve clear additional costs, such as the break time (which can involve a meal), the night time or the extra time. All other features that cannot be directly translated into direct costs (such as the duty type, the number of small breaks or the number of bus changes) should be handled very carefully and they could probably be measured as a percentage of the fixed direct cost of a duty. For example, if we want to penalise the existence of duties with small breaks, we should ask the planner how much this feature will increase the cost of a duty. If no additional costs are considered and the planner is able to state that a duty with a small break is 20% more expensive than a duty without a small break, he must be aware that having five duties with small breaks is equivalent to having six duties without small breaks. The awareness of the trade-offs between the features of the duties in the solution is very important to allow a meaningful and objective evaluation of a solution.

In our approach we have considered direct costs and penalisations as separate components, and we have explicitly left to the planner the option of including the penalisations into the cost function. Expression 5.19 highlights these two components of the cost of a duty \( j \).

\[
C(j) = \text{DirectCost}(j) + \text{Penalisation}(j)
\]  

(5.19)

5.6.2 Evaluating the cost of a leftover

The cost function must include a penalisation for the leftovers, otherwise we would obtain a solution with no duties (and cost 0). The evaluation of the cost of each leftover is carried out in two phases: first, we compute the minimum number of drivers required to perform the leftover; next we assign a cost to it.

In order to compute the minimum number of drivers needed for each leftover \( l \) (see Section 5.4.1), we divide the duration of the leftover \( (\text{dur}(l)) \) by the maximum driving time \( (\text{MAX\_DRIVING\_TIME}) \) and we take the rounding up of this value (see Expression 5.20). In fact, the current Portuguese legislation imposes five hours as maximum driving time, which is at the same time, the minimum duration of a straight duty, for most companies. For example, if a leftover lasts 6h30m we need, at least, two drivers and thus, two duties, to perform it. Therefore, each leftover is equivalent to a number of duties that is
dependent on its duration and on the maximum driving time.

\[
\text{num\_drivers}(l) = \left[ \frac{\text{dur}(l)}{\text{MAX\_DRIVING\_TIME}} \right]
\] (5.20)

In the second phase, the cost of each leftover \( l \) (see Expression 5.21) is evaluated multiplying the minimum number of drivers by a fixed amount that is proportional to the most expensive duty in the set of generated duties (see Expression 5.22). By default, \( \lambda = 1 \).

\[
C(l) = \text{FIXED\_COST} \times \text{num\_drivers}(l)
\] (5.21)

\[
\text{FIXED\_COST} = \lambda \times \max_{j \in J}(C(j)), \lambda > 0
\] (5.22)

There are two reasons for adopting this approach:

- first, the constraint that limits the maximum driving time, imposed by legislation, must be respected, even by the leftovers, independently on the way they are handled.

- second, we should be able to differently evaluate two or more solutions with the same number of leftovers and the same original cost. A simple criterion to rank solutions is by the duration of the leftovers. In fact, as we are trying to minimise the cost / number of leftovers, if we do not consider the duration of the leftovers in the cost function, we could obtain solutions with few, but very large leftovers and, therefore, a large portion of the schedule would remain uncovered.

### 5.6.3 Minimizing the total cost of a solution

To evaluate the total cost of a solution we add up the costs (and eventually, the penalisations) assigned to the duties and to the leftovers.

Let \( S \) be a solution of the problem, \( j \) a feasible duty and \( l \) a leftover. Expression 5.23 defines the total cost of the solution.

\[
eval_S(S) = C(S) = \sum_{j \in S} C(j) + \sum_{l \in S} C(l) = \sum_{j \in S} C(j) + \lambda \times \max_{j \in J}(C(j)) \times \sum_{l \in S} \text{num\_drivers}(l)
\] (5.23)
5.6.4 Minimizing the number of duties

The minimisation of the number of duties is a special case of the minimisation of the total cost of the duties, when all duties have unitary costs \((C(j) = 1, \forall j \in S)\). In fact, this function tries to minimise the number of drivers needed to perform all the duties as displayed in Expression 5.25.

\[
eval_g(S) = |\{j : j \in S\}| + \lambda \sum_{l \in S} \text{numdrivers}(l)
\]  

(5.25)

Depending on its duration, each leftover is then considered as a set of duties with unitary costs. In this case, the way to divide leftovers into a set of duties according to their duration is crucial, otherwise we could obtain a solution with leftovers large enough to contain several duties.

Example 5.8 shows that minimizing the number of duties or minimizing the total cost of a solution can produce very different results.

Example 5.8 Objective functions

Solution 1 was obtained for the minimisation of the total cost of duties, while Solution 2 was obtained for the minimisation of the number of duties. Both solutions were obtained with no additional soft constraints. We can see that, although Solution 1 has a lower cost, it contains more duties than Solution 2, which has a higher cost. Since both solutions present the same number of leftovers, Solution 2 contains more expensive duties than those contained in Solution 1. In this particular example, the difference between the number of duties in the two solutions is too high (five duties) and the planner may want to check the values he had defined for the duty costs, before accepting Solution 1. Or, otherwise, he may prefer Solution 2 due to the smaller number of duties, despite of its cost.

<table>
<thead>
<tr>
<th>Sol.</th>
<th>objective functions</th>
<th>left.</th>
<th>tot.dur.left.</th>
<th>duties</th>
<th>%cov.sch.</th>
<th>total cost duties</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>total cost of duties</td>
<td>3</td>
<td>6:08</td>
<td>59</td>
<td>98,2</td>
<td>326000</td>
</tr>
<tr>
<td>2</td>
<td>number of duties</td>
<td>3</td>
<td>8:01</td>
<td>54</td>
<td>97,9</td>
<td>392000</td>
</tr>
</tbody>
</table>

left: number of leftovers; tot.dur.left: total duration of leftovers duties: number of duties; %cov.sch: covered schedule rate

5.7 Summary

In this chapter we have presented and integrated analysis of the BDSP. The major issues that determine the complexity of the problem have been identified and discussed. Evaluating the quality of a bus driver schedule is certainly a complex task. Some of the criteria used by the planners have been outlined.

We have identified two main groups of constraints: hard constraints that define feasible solutions and soft constraints, that help us to characterise the quality of the solutions. Furthermore, hard constraints
can be divided into two subgroups, related to two different types of solutions: partitioning solutions and covering solutions. Independently of the group of hard constraints that we choose for a particular problem, we can impose several distinct soft constraints.

In our GA approach, that will be described in the next chapter, the crossover and the mutation operators maintain the satisfaction of the hard constraints, while each soft constraint is transformed into an objective function to be minimised. Thus, all solutions produced by our multiobjective GAs are feasible for the group of hard constraints considered.

The choice of the constraints to use in a particular problem depends on the structure of the problem and on the operational planning rules of the company. However, the user is able to test different sets of hard and soft constraints, run several experiments and analyse and compare alternative scenarios.

We have also defined a set of objective functions, involving the estimated costs of duties or penalisations for the leftovers. Although a precise estimation of the cost of each leftover can be hard to perform, we have proposed a general and simple way to compute this cost.
Chapter 6

Multiobjective Genetic Algorithms for the BDSP

6.1 Introduction

In this chapter we describe the multiobjective genetic algorithms that we have designed and implemented for the BDSP. These algorithms have been based on two models, referred to here as the Aggregate (Agg) model and the Non-Dominated (ND) model.

The coding schemes that we have implemented are discussed in Section 6.2, and the GA operators that are independent of the model, are presented in Section 6.4.

Section 6.5 described the Agg model, which is based on the traditional approach that converts the original multiobjective problem into a single objective problem, by performing a linear combination of the objectives. This is one of the most common approaches used to solve multiobjective problems. Its simplicity is very attractive and it often gives good results when few different objectives are considered [36]. However, for complex, non-structured problems, with several conflicting objectives, it becomes quite difficult to parametrize the aggregate function.

Section 6.6 describes the ND model, which is based on a "Pareto dominance" approach. In this model the different objectives are considered independently and the final result is a set of approximations to the non-dominated solutions, hopefully representative of the different trade-offs, instead of a single solution as in the Agg model.

In Section 6.7 we discuss and describe several diversification strategies that we have implemented.
6.2 Coding schemes

In a GAs approach to a Combinatorial Optimisation Problem (COP), a solution can be represented by a finite sequence of parameters or variables that forms a string of values known as a chromosome. Each parameter on the string is called a gene and the value of a parameter is known as an allele. In the classical GA proposed by Holland a chromosome is represented by a finite sequence of binary digits and hence in such representation each variable or parameter must be coded in a binary form. Holland [91] and Goldberg [80] state that theoretically, a binary representation has advantages when compared with higher cardinality alphabets. The argument is that a binary representation maximises the information that the GA processes, by means of implicit parallelism. This mechanism is one of the explanations for the good performance of GAs and is based on the Schema Theorem ([91]).

Consider a binary coding and the alphabet \( \{0, 1, *\} \), in which the wildcard means any symbol in \( \{0, 1\} \). A schema is a pattern of gene values which can be represented by a string of characters in the alphabet \( \{0, 1, *\} \). All chromosomes that match a particular schema are said to belong to that schema. For example, the schema \( * * 1 * 1 * 0 * \) has 32 members which include 10111000 and 10101001. Similarly, each particular chromosome contains several different schemata (for example, 10111000 contains the schemata \( 1 * * 1 * * 0 \) and \( * * 1 * * 00 \)). The length of a schema is the distance between the first and the last fixed symbols and the order of a schema is the number of defined positions. For example, the schema \( * * 1 * 1 * 0 * \) has length 4 and order 3.

Holland considered that individuals with higher fitness also contain good schemata and the Schema Theorems says that allocating reproductive trials to individuals in proportion to their fitness, the good schemata receive an exponentially increasing number of trials in successive generations. These good schemata, known as building blocks, correspond to short, low order, above average schemata.

Moreover, since each individual has a large number of different schemata, the number of schemata that are effectively processed in each generation is of the order \( n^3 \) (for a binary coding, where \( n \) is the population size). This property is known as implicit parallelism and it is the major argument for using binary codings. In fact, the Principle of Minimal Alphabets ([91]) states that given two possible codings, the one with the lower cardinality alphabet gives a greater number of schemata sampled by a given population.

However, higher cardinality codings have been used with success in many GA applications, namely for certain COPs, such as the travelling salesman problem or the job shop scheduling problem [45]. Moreover, Antonisse [3] and Koehler [96] have concluded that, on the contrary, high cardinality alphabets can process more schemata than binary ones and this result gave rise to several theoretical discussions [81]. Nevertheless, GA practitioners have used several different non-binary codings that include floating point representation, linked lists or matrices.
In order to guide the GA practitioner in choosing an appropriate coding, Goldberg [80] proposed the *Principle of Meaningful Building Blocks*, which states that we should select a coding so that short, low order schemata are relevant to the underlying problem and relatively unrelated to schemata over other fixed positions. Hence, the selection of a coding scheme should be guided by the following principles [9]: related genes should be close together on the chromosome and there should be little interaction between genes. These two conditions are not always easy to meet. Interaction (also referred as *epistasis*) between genes means that the contribution of a gene to the fitness depends on the value of other genes in the chromosome. While some interaction between genes always exists in multimodal fitness functions, it is often very difficult to design a coding that is able to reduce it to the minimum. At the same time, in order to keep related genes close together one should know in advance what is the relationship between them and this is not always possible. Even when simple relationships exist it may still be impossible to reflect them in the coding scheme.

Choosing the most appropriate coding scheme is a crucial decision for most genetic algorithms. It should encapsulate the relevant information of a solution of the problem in a simple data structure, enabling crossover and mutation operators to operate effectively. Moreover, for constrained problems, the coding scheme should be chosen in such a way that it only represents feasible solutions and the recombination operators only produce feasible solutions. Some authors claim that finding a coding scheme with these features is the most important decision for the success of the genetic algorithms [91], [45]. Unfortunately, for some constrained problems (and the BDSP is one of them) one could not find until now any coding scheme that ensures the feasibility of the solutions when the traditional operators are applied.

In a standard GA, the operators are problem independent and do not need to know the meaning (semantics) of the information. They operate directly on the coded solutions, mixing and destroying *building blocks* in order to effectively search the space of the solutions and do not use extra information. The coding scheme organises the information into sequences of building blocks that must not be destroyed by the recombination operators. Therefore, the effectiveness of the search is deeply dependent on the chosen coding scheme.

In non standard (hybrid) GA, we can use crossover and mutation operators that are problem dependent. These operators are designed specifically for a particular problem, using problem knowledge that may not be encapsulated in the coding scheme. They may operate directly in the coding scheme, but they can also use extra information about the problem. In this case, the effectiveness of the search relies more on the design of the operators than on the coding scheme. Instead, the coding scheme must promote the efficiency of the search and its simplicity depends on the problem structure and on the design of the recombination operators.
In an attempt to efficiently tackle our problem, we propose in this section a partition of the global available information into two levels with increasing degree of abstraction. Two different coding schemes result directly from this partition, as presented in Figure 6-1.

All the information about the problem is resumed in global data structures. This information is available independently of the coding scheme used.

At the first level, information is organised directly as a *duties based coding scheme* (DC), which is the most common coding scheme for the application of genetic algorithms for the BDSP [103], [33], [99].

At the second level, we have a higher degree of abstraction, leading to the design of a *pieces-of-work based coding scheme* (PWC) [53] which was chosen for our multiobjective genetic algorithms. As we will see in Section 6.2.3, PWC represents a good trade-off between the representation of a (feasible) solution for the BDSP and the efficiency of the search in the global data structures.

![Diagram](image)

**Figure 6-1: Information Levels**

### 6.2.1 Global data structures

There are two main sources of information for the BDSP: the set of active pieces-of-work and the set of generated duties (see Chapter 2).

These two types of information will be searched very often, thus implying that access should be easy and fast. Direct access to the database could be a slow process, and the frequency of the queries could considerably slow down the algorithm. We have therefore chosen to query the database at the beginning of the process and keep the information in fast access memory. In this process we only select the information that is relevant to the algorithm. Hence, for each instance of the problem, we have in
memory two main data structures: a list with the active pieces-of-work and a list with all the generated duties.

We have designed the data structures to quickly answer the following questions:

- which duties cover piece-of work \( t_i \)?
- which are the pieces-of-work that cover duty \( s_j \)?
- which is the piece-of-work next to \( t_i \)?

We have chosen a linked list for the pieces-of-work because they are mainly searched sequentially. In fact, this list is sorted by bus and time, as search is usually performed by adjacent pieces-of-work. On the other hand, the generated duties are accessed directly, so an array was chosen for this purpose.

The linked list of pieces-of-work

The set of active pieces-of-work, is organised as a linked list and each item corresponds to a piece-of work (see Figure 6-2). Each item of the list is composed by two data structures:

1. the characterisation of the piece-of-work, \( t_i, i \in \{1, \ldots, m\} \) which includes:
   - the identification key of the piece-of work
   - the associated running board identification key
   - the start and finish relief opportunities with the information about the corresponding nodes, routes and times.

2. the set of the generated duties that cover that piece-of-work: \( \{s_{t_i,1}, \ldots, s_{t_i,k_i}\} \)

The linked list is sorted by the running board key and within each bus, it is sorted by ascending starting time.

![Figure 6-2: Data structure used to store the list of pieces-of-work](image-url)
The array of generated duties

The set of generated duties is stored in an array that contains the following information for each element (see Figure 6-3):

1. the characterisation of the duty, \( s_j, j \in \{1, \ldots, n\} \) which includes:

   • the identification key of the duty
   • the duty cost
   • the duty penalty
   • the duty type
   • the duration of the duty

2. the linked list of the pieces-of-work of the duty, sorted by ascending starting time

![Figure 6-3: Data structure used to store the set of generated duties](image)

In fact, a hybrid genetic algorithm is able to operate directly on these data structures. In this case, each solution of the problem would be represented by a subset of the list of generated duties containing the duties in the solution. The number of elements in each list would vary according to the number of duties in the solution. The GA would then operate in the solutions themselves, instead of operating in their representation.

This approach has several disadvantages:

• the evaluation of a solution is a very hard operation: an exhaustive search of the global data structures must be performed for each solution, and this may be very inefficient, in particular when evaluating the feasibility of a solution (the satisfaction of hard constraints);

• the standard operators cannot be applied, since the representation of each solution is not binary and has variable length (the length is the number of duties in the solution); moreover, the standard operators do not guarantee that hard constraints are satisfied;
the application of specialised operators would necessarily imply exhaustive search of the global data structures, in order to preserve the feasibility of the solutions.

In conclusion, the global data structures should be designed carefully, in order to minimise the memory requirements and to speed up the search. However, there is no advantage in operating directly on the data structures, independently of the genetic algorithms we use. So, in the next sections, we present and discuss two alternative coding schemes that correspond to natural representations of a solution for the BDSP.

6.2.2 First level information: Duties Based coding scheme

The coding scheme that is most commonly used is given by a binary vector with fixed length, where each gene is associated with a duty, and its value is either one or zero, according to the presence or absence of the duty in the solution. Consider, for example, a problem with 14 pieces-of-work and 20 candidate duties and a solution composed by duties 1, 5, 8, 11 and 12. The corresponding chromosome will have length 20 (see Table 6.1).

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1: Example of a DC chromosome

One advantage of this scheme, that we will call duties based coding scheme (DC), comes from the fact that a binary representation is used, which makes the data structures extremely simple, compact and easy to manipulate by the standard operators. In this case, additional procedures or operators should be implemented in order to handle the unfeasible solutions that may be produced.

However, this coding scheme has several disadvantages, briefly described below.

- Each chromosome contains only the information about the subset of duties in the solution. To get any additional information, such as the pieces-of-work in each duty, one needs to perform a set of rather time consuming operations. Besides, it does not include any information about the uncovered or the overcovered pieces-of-work in the solution. In Figure 6-4 we present the search stages we need to perform in order to obtain the information needed to evaluate each solution. For each duty in the solution we must search the list of generated duties to get the pieces-of-work that it covers and then, we must put all this information together, to obtain the uncovered (and, eventually, the overcovered) pieces-of-work.

Moreover, building the set of leftovers of a solution could be a relatively slow operation, since we must search in the set of uncovered pieces-of-work for those that are contiguous, in order to build
CHAPTER 6. MULTIOBJECTIVE GENETIC ALGORITHMS FOR THE BDSP

each leftover.

\[
\begin{array}{cccccc}
1 & 2 & 3 & \cdots & \cdots & n-1 & n \\
1 & 1 & 0 & \cdots & \cdots & 0 & 1
\end{array}
\]

![Diagram of coding scheme](image)

Figure 6-4: Duties based coding scheme

- This coding scheme does not satisfy the *Principle of Meaningful Building Blocks*, since the building blocks are distributed across the chromosome and are easily destroyed by the recombination operators. In fact, the duties in the chromosome are sorted according to their key, which has no special meaning. The "good duties" are not necessarily near to each other and there is a strong interaction between the genes.

- We must apply some additional procedure to handle unfeasible solutions. In fact, several authors have used this coding scheme in the application of GAs for set partitioning [33] and set covering problems [16]. Any how, these models do not allow the existence of leftovers in the solutions. Specialised operators have therefore been designed to handle unfeasible solutions. The evaluation of a solution in these GAs for SPP or SCP is very simple, since it is based only in the cost of the duties. However, due to the search operations involved, it is neither easy nor efficient to guarantee the satisfaction of the hard and soft constraints for the BDSP (see Chapter 5).

6.2.3 Second level information: Pieces-of-Work based coding scheme

As an alternative to the DC scheme, we propose the *pieces-of-work coding scheme* (PWC), that aggregates the two fundamental types of information contained in a solution: duties and pieces-of-work. In fact, each solution for the BDSP is composed by duties and leftovers and each leftover corresponds to a set of contiguous uncovered pieces-of-work.
In the PWC scheme each gene of a chromosome is associated with a piece-of-work and its value is the duty that covers that piece-of-work in that particular solution (see Figure 6-5). The genes are ordered by bus and within each bus they are ordered by ascending time, exactly as the global list of pieces-of-work.

Figure 6-5: Pieces-of-work based coding scheme

Consider, for example, the chromosome in the PWC coding scheme presented in Table 6.2, which represents a solution for a problem with 18 pieces-of-work. The value of each gene is the duty that covers the piece-of-work (e.g. the duty 7 covers pieces-of-work 1, 2 and 3). When it is not possible to cover all the pieces-of-work, those that remain uncovered are given the value 0, such as piece-of-work 4. Those pieces-of-work that remain uncovered for all possible solutions, corresponding to problem leftovers (see Chapter 5), are given a negative integer value (in fact, -2). This solution contains 6 duties (3, 7, 8, 9, 11, 14) and one problem leftover (the piece-of-work 14). To obtain the set of leftovers for this solution we simply have to check if the uncovered pieces-of-work in consecutive genes can be considered a single leftover (if they belong to the same running board and there is no time gap between them) or if they correspond to more than one leftover. In the example of Figure 6.2, we would have to make this test for pieces-of work 4 and 5, and for pieces-of work 8 and 9.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>8</td>
<td>8</td>
<td>11</td>
<td>-2</td>
<td>9</td>
<td>9</td>
<td>11</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 6.2: Getting the leftovers for a PWC chromosome

The PWC scheme corresponds to a natural representation of partitioning solutions, but it can also represent covering solutions. In fact, for partitioning solutions, the array data structure chosen for the
chromosome is a natural way to guarantee the satisfaction of hard constraints (as we cannot have more than one duty in each piece-of-work).

For covering solutions, we can use the same coding scheme but, in this case, a chromosome does not contain all the information needed to evaluate a solution (it is a "corrupted" chromosome). Consider the example presented in Figure 6.2, and suppose that we want to transform this partitioning solution into a covering solution by adding duty 10, that covers pieces-of-work 3, 4 and 6. Piece-of-work 4 is uncovered, but pieces-of-work 3 and 6 are already covered by duties 7 and 3, respectively. In fact, we will have the solution presented in Table 6.3:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>10</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>8</td>
<td>8</td>
<td>11</td>
<td>-2</td>
<td>9</td>
<td>9</td>
<td>11</td>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.3: A possible representation of a solution with overcovers

However, it is not necessary to put duty 10 as a second duty in pieces-of-work 3 and 6, since it is the only duty that covers piece-of-work 4. Thus, PWC scheme is enough to represent all the duties in a solution (see Table 6.4), even if the solution has some overcover time, as every duty in a solution must cover at least one piece-of-work that is not already covered.

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|
| 7 | 7 | 7 | 10| 0 | 3 | 3 | 0 | 0 | 14| 8 | 8 | 11 | -2 | 9 | 9 | 11 | 14 |

Table 6.4: A simplified representation of a solution with overcovers using the PWC scheme

The main disadvantage of the PWC scheme for covering solutions is that we cannot directly identify the pieces-of-work covered by duty 10 that are overcovered. This information is needed to evaluate the hard constraints related with covering solutions (see Section 5.3), such as the number of overcovered pieces-of-work or the overcovered time, but it can be obtained by inspecting directly the array of generated duties.

In conclusion, the main features of the PWC scheme are the following:

- It is not binary, and thus standard operators cannot be applied.

- Each chromosome contains almost all the information that is needed to evaluate a solution. In fact, a simple inspection of the chromosome will tell us which are the duties in the solution, and also the pieces-of-work that remained uncovered. Since the genes are ordered by time within each running board, it is very easy to obtain the set of leftovers for a solution. Problem leftovers are also easily identified.
6.3. **HANDLING HARD CONSTRAINTS**

- Information is stored in a compact form. In general the number of candidate duties (several thousands) is considerably larger than the set of pieces-of-work (only few hundreds), but each solution is composed by a small number of duties (at most, equal to the number of pieces-of-work). Therefore, each chromosome in the PWC scheme is very small, when compared to a chromosome in the DC scheme and, at the same time, it contains all the relevant information for a particular solution (the covered and uncovered pieces-of-work and the duties of that solution).

- Specialised operators that do not destroy the building blocks can be easily designed. The PWC scheme, although not binary, is extremely simple and natural, making chromosome manipulation very easy by the specialised operators we have designed (see Section 6.4). In fact, this coding scheme corresponds to a "spatial" representation of a solution, in the sense that it gives for each bus the relative position of the duties in time. Therefore, any set of consecutive genes corresponds to a meaningful building block, since it states which duties were chosen for a particular bus at a particular time. It is easy to build specialised operators that take advantage of these building blocks, as we will see in Section 6.4.3.

- The satisfaction of the hard constraints can be easily guaranteed, for partitioning and covering solutions, as we will see in the next section.

The coding scheme for a non standard genetic algorithm should in general be strongly related to the specialised operators designed for the problem. The PWC scheme is well suited for the BDSP as it is compact, flexible and guarantees the efficiency of a set of specially designed operators.

### 6.3 Handling hard constraints

There are several ways to guarantee the satisfaction of hard constraints. These different ways are described in this section.

(i) **Generation of all the possible solutions, discarding the unfeasible ones.**

   In this approach, any solution in the search space can be generated, but when an unfeasible solution is produced, it is discarded from the population. This approach allows the application of standard operators, but in highly constrained problems it may be quite difficult to obtain a single feasible solution. If the search is not oriented to the feasible region of the search space, we risk finding no more than a small subset of all feasible solutions.

   This is precisely what happens with the partitioning hard constraints identified for the BDSP: these constraints, imposing that no piece-of-work can be covered by more than one duty, are so
restrictive that, if we allow the generation of unfeasible solutions, we risk to observe the evolution of the algorithm without obtaining a single feasible solution. Moreover, if crossover and mutation operators do not maintain the feasibility of solutions, we may loose the good characteristics of the parents, by only producing unfeasible solutions.

For the hard constraints of the covering model we can make similar remarks, although the number of feasible solutions may be much higher. In fact, the number of feasible solutions will increase with the number/duration of the overcovers. If we allow an unlimited number of overcovers or an unlimited overcover time, all the generated solutions are feasible solutions. In this case, the size of the search space is extremely large. Moreover, if we do not include some penalty to the overcovers we will not be able to differently evaluate the candidate solutions involving overcovers.

We have discarded this procedure at the beginning of our work, since we have faced a lot of difficulties in obtaining good solutions in reasonable time. However, generation of unfeasible solutions might produce good results when combined with procedures (ii) and (iii).

(ii) The application of a heuristic procedure, called repair operator, to transform each unfeasible solution into a feasible one. Although this approach can be computational expensive, it has been successfully applied to several combinatorial optimisation problems [32], [58].

(iii) The design of a function (the unfitness function) to measure the degree of unfeasibility of the solutions. The separation between the fitness and unfitness functions may be very useful when it is difficult to build feasible solutions [32], [107]. This procedure will transform an hard constraint into a soft constraint.

(iv) The design of a coding scheme that ensures the feasibility of the solutions. Although being an interesting alternative, it does not always guarantee that the traditional crossover and mutation operators will maintain the feasibility of the solutions. Hence, this approach is usually associated with the design of specialised operators that incorporate problem specific knowledge.

(v) The design of specialised operators that, working with feasible solutions, do only produce feasible solutions. The design of such operators may be very difficult for some problems and quite natural for others. For instance, for the set partitioning problem it may be extremely hard to construct a feasible solution based on two (or more) feasible solutions. However for the set covering problem and for the relaxed partitioning problem it is quite simple to design these operators.

In our work we have designed specialised operators that maintain the feasibility of solutions (v) and a coding scheme that checks their feasibility (iv).
We emphasise that the choice of the process for handling hard constraints is not a trivial decision. It depends on the structure/topology of the search space and on the search process adopted. Due to the complexity of the search space of the BDSP, a successful application of GAs deeply depends on the way we understand and characterise that space, and how we deal with the concept of feasible solutions (see Chapter 5).

6.4 Independent operators

As referred in the Chapter 4, a genetic algorithm may be structured in the following stages (see Figure 4.3): Initialization, Evaluation, Fitness Assignment, Selection, Crossover, Mutation and Population Replacement. The multiobjective genetic algorithms models that we have implemented, the Aggregate model and the Non-dominated model, follow this generic structure. However, the techniques involved in some of these steps vary with the model while, for other steps, they remain the same.

In this section, we present and discuss the techniques and procedures that are common to both models, namely the process of selecting the eligible duties and the procedures involving:

- the generation of the initial population: this procedure builds the chromosomes for the initial population, guaranteeing the satisfaction of the hard constraints;

- the crossover and mutation operators: these procedures are applied to the chromosomes selected for reproduction, guaranteeing the satisfaction of the hard constraints that have been imposed to a particular problem.

The Fitness Assignment and the Selection stages are different for each model and hence they will be described in Sections 6.5 and 6.6.

Next, we present the notation that is commonly used for the BDSP and also the notation we used in our multiobjective genetic algorithms:

Notation 6.1 (BDSP notation) For each particular problem, consider the following sets:

$I$ is the set of the pieces-of-work, $I = \{i_1, i_2, \ldots, i_m\}$.

$J$ is the set of generated duties, $J = \{j_1, j_2, \ldots, j_n\}$.

$S$ is the set of duties in a solution, $S \subseteq J$.

$I_S$ is the set of pieces-of-work covered by the duties in $S$, $I_S \subseteq I$.

$I_s$ is the set of pieces-of-work covered by duty $s$, $I_s \subseteq I$.

$\overline{T}_S$ is the set of the overcovered pieces-of-work in solution $S$.

Consider also the following functions:
dur (i) returns the duration of piece-of-work i.

bus (i) returns the running board that contains piece-of-work i.

type (j) returns the type of duty j.

Notation 6.2 (GA notation) In our genetic algorithms we have the following components:

\( P_t \) is a population of chromosomes (a set of PWC coded solutions), \( P_t = \{C_{S_1}, C_{S_2}, \ldots, C_{S_{\text{max}}}\} \).

\( C_S \in P_t \) is a single chromosome that corresponds to a PWC representation of solution \( S \). According to this coding scheme, each chromosome \( C_S \) is an array with \( m \) elements, \( (C_S[i], i = 1 \ldots m) \).

6.4.1 Defining and selecting eligible duties

The concept of eligible duty is central to all the operators we have implemented. We say that a duty is eligible for a partial solution \( S \), if we can add it to \( S \) without violating any hard constraint. This concept is used for the first time in the generation of the initial population, and it is used in all the crossover and mutation operators. In fact, all these operators are based on simple operations related with the set of eligible duties, as we will see in the next sections. For example, adding a duty to a chromosome (solution) means that a set of eligible duties has been defined, some procedure was used to select a duty from this set, the selected duty was removed from the set and, finally, that the set of eligible duties was updated. Hence, the set of eligible duties is constantly changing, according to the operations performed on it. All these operations are straightforward, except the duties selection process.

We have implemented different procedures to select a duty from the set of eligible duties in order to add it to the solution under construction. These procedures are crucial to the performance of the GA, independently of the model adopted. In this section we describe the general process of building the set of eligible duties and the different ways to manipulate it.

Building the set of eligible duties

Consider the following definitions:

Definition 6.1 (Overcover time of a duty in a solution) The overcover time of a duty \( j \in J \), \( \bar{\text{dur}}(j, S) \), is the total duration of the common pieces-of-work of \( j \) and the duties in \( S \):

\[
\bar{\text{dur}}(j, S) = \sum_{i \in I \cap J \cup S} \text{dur} (i).
\]
6.4. INDEPENDENT OPERATORS

Definition 6.2 (Overcover time of two duties) The overcover time of two duties, $\overline{\text{dur}}(j_1, j_2)$, is the total duration of the common pieces-of-work of any two duties $j_1, j_2 \in J$:

$$\overline{\text{dur}}(j_1, j_2) = \sum_{i \in I_{j_1} \cap I_{j_2}} \text{dur}(i).$$

Definition 6.3 (Overcover time of a solution) The overcover time of a solution $S$, $\overline{\text{dur}}(S)$, is the total duration of the overcovered pieces-of-work in that solution:

$$\overline{\text{dur}}(S) = \sum_{i \in I_S} \text{dur}(i).$$

Definition 6.4 (Eligible duty for a solution) Let $j \in J \setminus S$. We say that duty $j$ is eligible for solution $S$ iff:

(i) $\text{dur}(j, S) \leq \text{DUR\_MAX\_OVERCOVER}$

(ii) $|I_{S \cup \{j\}}| \leq \text{NUM\_MAX\_OVERCOVER}$

(iii) if duty $j$ is added to solution $S$, all other duties in $S$ still satisfy (i):

$$\overline{\text{dur}}(j_1, S \cup \{j\}) \leq \text{DUR\_MAX\_OVERCOVER}, \forall j_1 \in S \cup \{j\}$$

(iv) $\overline{\text{dur}}(S \cup \{j\}) \leq \text{TOTAL\_DUR\_MAX\_OVERCOVER},$

where $\text{DUR\_MAX\_OVERCOVER}$, $\text{NUM\_MAX\_OVERCOVER}$ and $\text{TOTAL\_DUR\_MAX\_OVERCOVER}$ are the parameters described in Section 5.3. Obviously, when overcovers are not allowed, these parameters are set to zero.

The set Eligible\_duties is composed by all the eligible duties for the solution under construction. For each new chromosome, the set Eligible\_duties is initially composed by all the generated duties, since the chromosome has no duties. As new duties are added to the chromosome, the set Eligible\_duties is updated. This process ends when Eligible\_duties is the empty set. The values of parameters DUR\_MAX\_OVERCOVER, NUM\_MAX\_OVERCOVER and TOTAL\_DUR\_MAX\_OVERCOVER considerably affect the number of elements in Eligible\_duties and the speed of the updating procedure.

The generation of each new chromosome is based on a procedure that selects duties from Eligible\_duties. Therefore, we guarantee that each new chromosome satisfies the hard constraints.

The example 6.1 illustrates the process for updating the set of eligible duties.

Example 6.1 In Figure 6-6 we can see several duties from the set of generated duties $J$ ($J_1 = \{s_1, s_2, s_3, s_{10}, s_{11}, s_{12}, J\}$). Suppose that for this particular problem we allow the existence of overcovers and the maximum dura-
tion of each overcover is 2 hours (\texttt{dur}_{\text{MAX\_OVERCOVER}} = 2). Consider the partial solution composed by duties \(s_1\) and \(s_2\), \(S = \{s_1, s_2\}\).

The set \texttt{Eligible\_duties} is then \(\{s_3, s_10, s_{11}, s_{12}, s_{13}\}\). In fact, all duties in \(J_1\), except the duties in \(S\) and duty \(s_{14}\) (since \(\text{dur}(s_2, s_{14}) > 2h\)) belong to \texttt{Eligible\_duties}. If we add duty \(s_3\) to the partial solution \(S\), duties \(s_{11}\) and \(s_{12}\) are removed from \texttt{Eligible\_duties}. At this moment, \(S = \{s_1, s_2, s_3\}\) and \texttt{Eligible\_duties} = \(\{s_{10}, s_{13}\}\).

Now, suppose that duty \(s_{10}\) is added to \(S\). In this case, we must remove \(s_{13}\) from \texttt{Eligible\_duties}, since if it were selected for the solution, duty \(s_{10}\) would have an overcover time larger than the maximum allowed. Then we have now \texttt{Eligible\_duties} = \(\emptyset\), and the process ends with a solution composed by duties \(\{s_1, s_2, s_3, s_{10}\}\).

\begin{figure}[h]
  \centering
  \includegraphics[width=\textwidth]{figure6-6.png}
  \caption{Updating the set of eligible duties}
\end{figure}

Selecting eligible duties

We have defined three different procedures to select duties from \texttt{Eligible\_duties} which are based on information about the duty types. In fact, we realised that the number of generated duties of each type may in practice be very different. Some duty types can represent near 90\% of the total number of generated duties (see Chapter 7 for examples of this situation) and these duty types may be very undesirable. This occurs when companies define some duty types with very flexible rules (see Chapter 5) in order to avoid extra working time. Often, these duty types result from direct negotiations between the company and the drivers or the unions. Since they are constrained by quite flexible rules, concerning the length of the stretches and the break times there is a huge number of possible combinations of pieces-of-work that form a feasible duty. However, in practice, companies want to minimise the number
of duties of these types. Hence, the great majority of the generated duties are from the least desirable types. If we use a fully random procedure to select the eligible duties, the duty types more represented in the set of generated duties have higher chances to be selected. This can lead to a biased representation of the search space influenced by the distribution of the duty types.

To overcome this problem, we have implemented the two following additional procedures beyond the pure random one.

1. **Maximum Percentage Procedure**: in this procedure, the planner defines the maximum percentage allowed for each duty type. The duty type is randomly selected first and an eligible duty of the selected type is added to the chromosome. The process is repeated until the maximum percentage is reached for a duty type. Then, all remaining duties of this type are removed from the set of eligible duties. The process goes on until the set of eligible duties is empty. When all types have the same maximum percentage (which is the default option), this procedure leads to an uniform distribution of the duty types.

2. **Priority Level Procedure**: in this procedure, the planner assigns priority levels to the different duty types. Since there can exist different duty types with the same level of priority, the set of eligible duties is divided into subsets associated to the existing priority levels. Then, duties are selected randomly from the subset having to the highest priority. When this set becomes empty, the procedure goes on with the subset associated to the next highest priority level. When all types have equal priority (which is the default option), this procedures provides an uniform distribution of duty types.

Both procedures are available to the planner, together with the random process. For each of the independent operators described in the forthcoming sections, the planner can dynamically choose which procedure to apply, depending on the features of the company and of the problem.

### 6.4.2 Generating the initial population

The first step in a GA is the generation of the initial population. In this procedure all the chromosomes of the first generation are often built in a random way, trying to obtain a diversified representation of the search space. The number of chromosomes that are generated depends on a parameter, the size of the population (**POPSIZE**), that is set by the user and cannot be changed during the execution of the algorithm.

Each chromosome making part of the initial population satisfies the hard constraints of the particular problem instance under analysis. A procedure to check constraint satisfaction is integrated in the chromosome generation process and is based on a set of **Eligible duties**.
The procedure that generates the initial population, briefly described in Algorithm 1, simply builds chromosomes and adds them to the population until the population size is reached. Whenever a duty is selected from the Eligible duties set, according to one of the procedures described in the previous section, and then added to the chromosome, this set is updated.

The initial population does not have duplicates (identical chromosomes) in order to promote diversification and a good representation of the search space (see Section 6.7).

In Algorithm 1 below, consider $P_0$ to be the initial population (at generation $t = 0$), $S$ a partial solution for a particular problem and $C_S$ the chromosome that encodes $S$.

**Algorithm 1 (Generation of the initial population) Generation ($P_0$)**

\[
P_0 = \emptyset
\]

\[\text{while} \ (|P_0| < \text{POPSIZE})\]

\[
S = \emptyset
\]

\[
\text{Eligible duties } = \{j \in J : j \text{ is an eligible duty for } S\}
\]

\[\text{while} \ (\text{Eligible duties } \neq \emptyset)\]

\[
j_1 = \text{selection procedure} (\text{Eligible duties})
\]

\[
S = S \cup \{j_1\}
\]

\[
\text{Eligible duties } = \{j \in J : j \text{ is an eligible duty for } S\}
\]

\[\text{if} \ (C_S \notin P_0) \text{ then } P_0 = P_0 \cup \{C_S\}\]

### 6.4.3 Crossover operators

Crossover operators play a very important role in a genetic algorithm, since they are responsible for creating new solutions, hopefully better than their parents, and in this way, effectively search the solutions space. In the design of specialised operators for a constrained problem, such as the BDSP, there are two major issues that must be taken into account:

(i) How to handle the hard constraints.

As we have seen in the previous section, the process that generates the initial population guarantees that all chromosomes satisfy the hard constraints. In the design of the crossover operators we have followed the same approach, imposing that only feasible solutions are produced.

(ii) How to promote the exchange of "good" blocks of information between parents.

This means that, in the design of operators, we should guarantee that the good features of a solution are kept in the resulting children. As we have seen in Section 6.2, this can be partially achieved by an appropriate coding scheme.
In this section we describe three different crossover operators:

- *Union* crossover
- *Two_Point* crossover
- *Swap* crossover

The Union crossover can use any number of parents to produce a single child. Both the Two_Point and the Swap crossover use two parents and they also produce a single child. Imposing that each application of crossover operators just leads to one child is in fact a way to promote the diversification of the population.

The *Union* crossover operator

The Union crossover operator is based on the assumption that the quality of a solution is mainly guided by the set of duties it contains. In this sense, if we merge two or more high-quality solutions (chromosomes with high fitness values) and use the sets of duties from each to build a new chromosome, we should also obtain a high quality solution.

This operator can use two or more parents to produce a child. We have tried this operator with two, three and four parents and we have tested the influence of the number of parents on the effectiveness of the operator. We have noticed that this number does not significantly affect the performance of the operator and, thus, we have decided to always use two parents. The procedure implemented to test the performance of the operator and the influence of the number of parents is described in Appendix A.

The new chromosome (the child) is built in a two phases procedure: first, we randomly select eligible duties from the parents and after this procedure, the empty pieces-of-work are filled with duties from the set of generated duties that are still eligible for the new chromosome.

In the first phase, the set composed by the duties from both parents is the initial set of eligible duties. A duty is selected, based on a particular selection procedure (see Section 6.4.1). Then, it is added to the solution under construction and the set of eligible duties is updated. This procedure goes on until the set of eligible duties is empty.

The second phase is very similar but the initial set of eligible duties is now composed by a subset of all the generated duties.

Next, we present an example that illustrates how this operator works when overcovers are not allowed. The application of this operator to solutions with overcovers depends only on how eligible duties are defined.
Example 6.2 (Union Crossover) Consider chromosome parent1 = \{2, 3, 8, 10, 11, 12, 13, 14\} that corresponds to the solution represented in Figure 6-7. This solution has 8 duties and 3 leftovers. Chromosome parent2 = \{1, 4, 5, 6, 7, 8, 9, 15\}, corresponding to the solution represented in Figure 6-8 contains 8 duties and 2 leftovers.

Phase 1: Build the set Union = parent1 ∪ parent2 = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}

Duty 9 was randomly selected from the Union set and added to the child chromosome, under construction. Then, duties 3 and 14 are no longer eligible for the new solution and Union = \{1, 2, 4, 5, 6, 7, 8, 10, 11, 12, 13, 15\}. This process goes on until Union is the empty set. At the end of this phase the child chromosome is composed by duties \{1, 2, 5, 6, 7, 8, 12\}.

Phase 2: Randomly select a duty from the set of all the generated duties. For this example, duty 16 was selected and added to the child chromosome. Thus, the new chromosome, child1 = \{1, 2, 5, 6, 7, 8, 9, 12, 16\}, represented in Figure 6-9, is composed of 9 duties and 0 leftovers.

Figure 6-7: A solution selected to be a parent for Union Crossover: parent1

Figure 6-8: A solution selected to be a parent to Union Crossover: parent2

A more formal description of the Union crossover operator is presented in Algorithm 2, where S is the child (solution) produced by parents S1 and S2.
Algorithm 2 (Union crossover) \( S = \text{Union}(S_1, S_2) \)

\[
S := \emptyset \\
\text{Eligible\_duties}_{S_1 \cup S_2} := S_1 \cup S_2 \\
\text{while} (\text{Eligible\_duties}_{S_1 \cup S_2} \neq \emptyset) \\
\quad j := \text{selection\_procedure} (\text{Eligible\_duties}_{S_1 \cup S_2}) \\
\quad S := S \cup \{j\} \\
\quad \text{Eligible\_duties}_{S_1 \cup S_2} := \{j \in S_1 \cup S_2 : j \text{ is an eligible duty for } S\} \\
\text{Eligible\_duties} := \{j \in J : j \text{ is an eligible duty for } S\} \\
\text{while} (\text{Eligible\_duties} \neq \emptyset) \\
\quad j := \text{selection\_procedure} (\text{Eligible\_duties}) \\
\quad S := S \cup \{j\} \\
\quad \text{Eligible\_duties} := \{j \in J : j \text{ is an eligible duty for } S\} \\
\text{return} S
\]

The analysis of the performance of this operator (with two parents), described in detail in Appendix A, allowed us to conclude that, on average, 73% of the children produced are better (have higher fitness) than the average of the population and that, on average, 39% of the children have fitness values higher than their parents. When comparing it with the others, the values obtained for this operator show that it has the best performance in improving the average of the population, and the worst performance in surpassing the parents.

The Two-Point crossover operator

The Two-Point crossover operator can be considered an adaptation of the standard Two Point crossover operator [91], since each of the parents is divided into three parts and the duties belonging to each part are exchanged to build a new chromosome. The underlying idea is that the quality of a solution is based
not only on the individual contribution of each duty but also on the contribution of some sets of duties. In other words, the meaningful building blocks are composed by sets of adjacent duties.

This operator uses two parents to produce one or two children. However, in order to promote the diversity of the population, we have decided that a single child will be produced (from the two possible ones).

The procedure for Two-Point crossover is composed by two distinct phases: in the first phase, both parents are divided into three parts, not necessarily of the same size. One of the parents is selected randomly to contribute to the first and third parts and the other one contributes to the middle part. The order by which the parents contribute to the new chromosome is important, since the same duty can cover pieces-of-work that belong to different parts (e.g., to the first and second part). Hence, the parent selected to contribute first has potentially more influence on the child than the second one. In order to overcome this bias, we randomly select the parent that is going to be the first.

The second phase is very similar to that of the Union crossover operator, since we try to fill the uncovered pieces-of-work with eligible duties selected from the set of generated duties J. The selection of the eligible duties from J can be based on a completely random process or alternatively, it can be based on the Maximum Percentage Procedure or on the Priority Level Procedure.

Example 6.3 illustrates how this operator works. In this example we do not allow overcovers, but the Two-Point crossover is also applicable to solutions with overcovers.

Example 6.3 (Two-Point crossover operator) Consider a problem with 10 pieces-of-work and 25 generated duties. Consider the following two solutions associated to chromosomes \( p_1 \) and \( p_2 \):

\[
\begin{array}{ccccccc}
1 & 1 & 2 & 3 & 4 & 5 & 2 \\
\downarrow & & & & & & \\
8 & 9 & 9 & 10 & 4 & 10 & 11 & 12 & 4 & 0 \\
\end{array}
\]

Suppose that parents \( p_1 \) and \( p_2 \) have been divided into three parts on the fourth and ninth positions, identified in the figure by a \( \downarrow \) symbol. Hence, considering that the filling process begins with \( p_1 \), we could obtain the following child \( c_1 \):

\[
\begin{array}{ccccccc}
1 & 1 & 2 & 10 & 4 & 10 & 2 \\
\downarrow & & & & & & \\
1 & 1 & 2 & 10 & 4 & 2 & 4 & 6 \\
\end{array}
\]

We can observe that, while each of the parents has 6 duties (\( p_2 \) also has a leftover), the child \( c_1 \) is composed by five duties and no leftovers. Moreover, this example also shows that the first parent (\( p_1 \)) contributes with more duties than the other parent. In fact, \( c_1 \) is composed of 4 duties that came from \( p_1 \) (duties 1, 2, 4 and 6) and one single duty from \( p_2 \) (duty 10). In this example, the second phase was not necessary, since the child is completely "covered" at the end of the first phase.
6.4. INDEPENDENT OPERATORS

We have analyzed the performance of this operator (see Appendix A) and we have concluded that it is effective in producing children with higher fitness than their parents (44% on average) and in producing children with a fitness higher than the average fitness of the population (65% on average).

A more formal description of the Two-Point crossover operator is presented in Algorithms 3 and 4.

Algorithm 3 (Two-Point crossover) \( S = \text{Two\_Point}(S_1, S_2) \)

\[
S := \emptyset \\
pos_1 := \text{rand}(I) \\
pos_2 := \text{rand}(I) \\
\text{if } (pos_1 > pos_2) \text{ then swap(pos_1, pos_2)} \\
\text{start} := \text{rand}(1) \\
S := \begin{cases} 
\text{phase1}(S_1, S_2, S_1) & , \text{start} < 0.5 \\
\text{phase1}(S_1, S_2, S_2) & , \text{otherwise}
\end{cases} \\
\text{Eligible\_duties} := \{j \in J : \text{j is an eligible duty for } S\} \\
\text{while } (\text{Eligible\_duties} \neq \emptyset) \\
\quad j := \text{selection\_procedure}(\text{Eligible\_duties}) \\
\quad S := S \cup \{j\} \\
\quad \text{Eligible\_duties} := \{j \in J : \text{j is an eligible duty for } S\} \\
\text{return } S
\]

Algorithm 4 (Phase1: Two-Point Crossover) \( S = \text{phase1}(S_1, S_2, S_{\text{first}}) \)

\[
\text{if } (S_{\text{first}} = S_1) \quad S_{\text{last}} := S_2 \\
\text{else } S_{\text{last}} := S_1 \\
\text{Elig\_first} := \{j \in S_{\text{first}} : \text{j is eligible for } S \land \exists i \in I_j : 1 \leq i \leq pos_1 \lor pos_2 \leq i \leq |I|\} \\
\text{while } |\text{Elig\_first}| > 0 \\
\quad j := \text{selection\_procedure}(\text{Elig\_first}) \\
\quad S := S \cup \{j\} \\
\text{Elig\_first} := \{j \in S_{\text{first}} : \text{j is eligible for } S \land \exists i \in I_j : 1 \leq i \leq pos_1 \lor pos_2 \leq i \leq |I|\} \\
\text{Elig\_last} := \{j \in S_{\text{last}} : \text{j is eligible for } S \land \exists i \in I_j : pos_1 \leq i \leq pos_2\} \\
\text{while } |\text{Elig\_last}| > 0 \\
\quad j := \text{selection\_procedure}(\text{Elig\_last}) \\
\quad S := S \cup \{j\} \\
\text{Elig\_last} := \{j \in S_{\text{last}} : \text{j is eligible for } S \land \exists i \in I_j : pos_1 \leq i \leq pos_2\} \\
\text{return } S
\]
The Swap crossover operator

The Swap crossover operator was inspired by the procedure used by the many planners when they build a new schedule manually (see Section 2.3.4). In this manual procedure, the first and last pieces-of-work for each bus are covered in the first place. The first piece-of-work for each running board always starts at a depot and the last one always ends at a depot. With this procedure, planners try to guarantee that there is always a driver to start the trips for each bus and that there is always a driver that leaves the bus at the depot. After this process is complete, they try to cover the pieces-of-work at the middle of the day.

The Swap operator adds a level of knowledge to the crossover procedure: the running board of each bus. In fact, each chromosome in the PWC coding is implicitly divided in several blocks corresponding to the buses of the problem. This operator is divided into three phases:

Phase 1 Build a set, named Fill_first, with the pieces-of-work that start and finish each running board. Then, build a set, named Duties_Fill_first, with the duties from both parents that cover the pieces-of-work in Fill_first. Select eligible duties from this set using any of the procedures described in Section 6.4.1 and add them to the new chromosome, until there are no more eligible duties in Duties_Fill_first.

Phase 2 Build the set Eligible_Union, composed by the remaining eligible duties from the parents. Use a selection procedure to select duties from this set and add them to the child under construction until the set is empty.

Phase 3 Finally, if there are still some eligible duties in the set of all generated duties \( J \), add them to the child until there are no more eligible duties to include in the solution.

Example 6.4 illustrates the application of Swap crossover to a problem that does not allow overcovers. Like the other two crossover operators, the Swap crossover is also applicable to solutions with overcovers. The only difference is the way in which eligible duties are defined.

Example 6.4 (Swap crossover operator) Consider chromosome parent1, associated to the solution represented in Figure 6-10. This solution has 12 duties and 0 leftovers. Chromosome parent2, associated to the solution represented in Figure 6-11 also contains 12 duties and 2 leftovers.

Phase 1: Find the initial and final pieces-of-work for each running board: Fill_first := \{1, 18, 19, 35, 36, 50, 51, 66\}.
Build the set Duties_Fill_first, composed by the duties from parent1 and parent2 that cover the pieces-of-work in Fill_first:

Duties_Fill_first := \{1, 2, 3, 4, 6, 8, 9, 12, 20, 21, 22, 23, 25, 26, 28, 30\}
The child is then filled with eligible duties (in this case, randomly) selected from the set Duties__Fill__first. At the end of this phase, the child chromosome is composed by duties \{1, 3, 6, 8, 21, 23, 28, 30\}.

Phase 2: Build the set Eligible__Union, with the duties from parent1 and parent2 that are still eligible. Eligible__Union = \{7, 27\}. Select duties from this set (randomly) until there are no more eligible duties from the parents.

Phase 3: select (randomly) an eligible duty from the set of all the generated duties. For this example, duties 32 and 33 were selected and added to the child chromosome. Thus, the new chromosome, child1 = \{1, 3, 6, 7, 8, 21, 23, 27, 28, 30, 32, 33\}, represented in Figure 6-10, is composed by 12 duties and 1 leftovers.

Figure 6-10: A parent chromosome for to Swap Crossover: parent1

Figure 6-11: A parent chromosome for to Swap Crossover: parent2

The global performance of the Swap crossover has been analyzed and compared with the other two operators (see Appendix A). The Swap crossover operator presents the best values in terms of improving the fitness of the parents (50%, on average), while 66% of the children produced with this operator contribute for the improvement of the average fitness of the population.

A more formal description of the Swap crossover operator is presented in Algorithm 5.
Algorithm 5 (Swap crossover) \( S = \text{Swap}(S_1, S_2) \)

\[
S := \emptyset \\
\text{Fill}_\text{first} := \{ i, i + 1 \in I : \text{bus}(i) \neq \text{bus}(i + 1) \} \\
\text{Duties}_\text{Fill}_\text{First} := \{ j \in S_1 \cup S_2 : \text{Fill}_\text{first} \subseteq I_{S_1} \cup I_{S_2} \} \\
\text{Eligible}_\text{duties}_{\text{First}} := \{ j \in \text{Duties}_\text{Fill}_\text{First} : j \text{ is an eligible duty for } S \} \\
\text{while } (\text{Eligible}_\text{duties}_{\text{First}} \neq \emptyset) \\
\quad j := \text{selection}_\text{procedure}(\text{Eligible}_\text{duties}_{\text{First}}) \\
\quad S := S \cup \{ j \} \\
\quad \text{Eligible}_\text{duties}_{\text{First}} := \{ j \in \text{Duties}_\text{Fill}_\text{First} : j \text{ is an eligible duty for } S \} \\
\text{Eligible}_\text{duties}_{S_1 \cup S_2} := S_1 \cup S_2 \\
\text{while } (\text{Eligible}_\text{duties}_{S_1 \cup S_2} \neq \emptyset) \\
\quad j := \text{selection}_\text{procedure}(\text{Eligible}_\text{duties}_{S_1 \cup S_2}) \\
\quad S := S \cup \{ j \} \\
\quad \text{Eligible}_\text{duties}_{S_1 \cup S_2} := \{ j \in S_1 \cup S_2 : j \text{ is an eligible duty for } S \} \\
\text{Eligible}_\text{duties} := \{ j \in J : j \text{ is an eligible duty for } S \} \\
\text{while } (\text{Eligible}_\text{duties} \neq \emptyset) \\
\quad j := \text{rand}(\text{Eligible}_\text{duties}) \\
\quad S := S \cup \{ j \} \\
\quad \text{Eligible}_\text{duties} := \{ j \in J : j \text{ is an eligible duty for } S \} \\
\text{return } S
\]

6.4.4 Destroying and repairing mutation operators

In a standard GA, the role of the mutation operator is to provide a higher level of diversification, in order to guarantee that the whole solution space is searched. Since our approach to the BDSP is based on a non-standard GA with specialised operators, this particular issue is not so critical.
We have divided mutation operators into two different groups: *destroying* operators and *repairing* operators. We have implemented a single destroying operator (*Destroy*) that destroys part of a solution and tries to rebuild it in a different way. Repairing operators act as local search heuristics which try to improve a particular solution, searching in its neighborhood. We have implemented three repairing operators (*Improve, Reduce_Empty* and *Reduce_Bus_Changes*) which try to improve a solution according to some given criterion. These three operators have been designed taking into account some real problems and the comments of the planners. The planner can choose which of them he wants to apply and also the associated probability rates. Several other operators, of the same type, can be implemented in the future.

Usually, a mutation operator has a small chance to occur in each generation. It is applied to a single chromosome, and it consists in changing the values of a small percentage of the genes of the chromosome. Furthermore, as the BDSP is a highly constrained problem, a small change in a single gene can turn a feasible solution into an unfeasible one. The mutation operators that we have designed guarantee the satisfaction of all hard constraints.

The *Destroy* mutation operator

The *Destroy* mutation operator is based on the following simple idea: remove part of the solution and rebuild it with new duties. The number of duties to be removed depends on a parameter (*DESTROY_RATE*) that can be defined by the user or by an internal random procedure. This parameter is the percentage of the total number of pieces-of-work from which duties are going to be removed. The duties are removed sequentially or, in other words, the chromosome is traversed sequentially for duties removal.

The procedure works as follows:

**Step 1** Randomly select a covered piece-of-work (corresponding to a gene of the chromosome) \( i \leq m \).

**Step 2** Starting from this gene, sequentially traverse the chromosome and remove all the duties from the chromosome until the desired percentage (*DESTROY_RATE*) of uncovered pieces-of-work is achieved. If the last gene is found before the *DESTROY_RATE* of uncovered pieces-of-work is obtained, repeat the process from the beginning of the chromosome (the first piece-of-work).

**Step 3** Build a set with all generated duties that are still eligible duties for the chromosome. Select duties from this set and add them to the chromosome until there are no more eligible duties in \( J \).

This operator allows a reorganisation of a small part of the chromosome without destroying many good building blocks. This explains why the duties are removed sequentially and not randomly or by any other selection process.
The Example 6.5 shows the application of this operator to a solution without overcovers, although it can also be applied to a solution with overcovers.

Example 6.5 (Destroy mutation operator example) Consider the solution represented in Figure 6-13, that corresponds to the chromosome \( c_1 \). This solution has 7 duties and 2 leftovers.

\[
c_1 = 1 \ 0 \ 7 \ 7 \ 9 \ 9 \ 9 \ 3 \ 3 \ 3 \ 2 \ 2 \ 0 \ 6 \ 6 \ 6 \ 5 \ 1 \ 1 \ 2 \ 7 \ 7
\]

We have chosen a destroy rate of 0.2, which means that we are going to remove duties from four pieces-of-work \((0.2 \times 22 = 4.4)\). The 20th piece-of-work was arbitrarily chosen to begin the process. We will remove all the duties that cover the pieces-of-work 20, 21, 22 and 1. This corresponds to removing duties 1, 2, and 7. The temporary chromosome is named \( c'_1 \):

\[
c'_1 = 0 \ 0 \ 0 \ 0 \ 9 \ 9 \ 9 \ 3 \ 3 \ 3 \ 0 \ 0 \ 0 \ 6 \ 6 \ 6 \ 5 \ 0 \ 0 \ 0 \ 0 \ 0
\]

Finally, we fill the uncovered pieces-of-work with eligible duties. The resulting chromosome is denoted by \( c_2 \) and the corresponding solution is represented in Figure 6-14.

\[
c_2 = 8 \ 8 \ 7 \ 7 \ 9 \ 9 \ 9 \ 3 \ 3 \ 3 \ 4 \ 4 \ 4 \ 6 \ 6 \ 6 \ 5 \ 10 \ 10 \ 10 \ 7 \ 7
\]

![Figure 6-13: Part of a solution selected for application of Destroy mutation](image)

![Figure 6-14: Result of the application of Destroy mutation](image)

A more formal description of the Destroy mutation operator is presented in Algorithm 6.
6.4. INDEPENDENT OPERATORS

Algorithm 6 (Destroy mutation) $S = \text{Destroy}(S, \text{DESTROY}_\text{RATE})$

- $\text{first} := \text{rand}(I)$
- $\text{last} := \text{first} + m \times \text{DESTROY}_\text{RATE}$
- $\text{last}_1 := \begin{cases} \text{last} & \text{last} \leq m \\ m & \text{otherwise} \end{cases}$
- $\text{last}_2 := \begin{cases} 0 & \text{last} \leq m \\ \text{last} - m & \text{otherwise} \end{cases}$
- $\text{Duties}_\text{removed}_1 := \{j \in S : \exists i \in I_j : \text{first} \leq i \leq \text{last}_1\}$
- $S := S \setminus \text{Duties}_\text{removed}_1$
- $\text{Duties}_\text{removed}_2 := \{j \in S : \exists i \in I_j : 1 \leq i \leq \text{last}_2\}$
- $S := S \setminus \text{Duties}_\text{removed}_2$
- $\text{Eligible}_\text{duties} := \{j \in J : j \text{ is an eligible duty for } S\}$
- while ($\text{Eligible}_\text{duties} \neq \emptyset$)
  - $j := \text{selection}_\text{procedure}(\text{Eligible}_\text{duties})$
  - $S := S \cup \{j\}$
- $\text{Eligible}_\text{duties} := \{j \in J : j \text{ is an eligible duty for } S\}$
- return $S$

The Improve mutation operator

The motivation for designing this operator came from the analysis of the solutions produced by the GA. In some solutions we have observed that there were some small uncovered pieces-of-work that could be easily covered if we were able to extend one of the adjacent duties (as we can see in Figure 6-15). However, the GA may never find these larger duties only with the application of the crossover and mutation operators described. The Improve mutation operator prevents these situations to occur and, in some sense, it can be viewed as a repairing operator. However, we have noticed that if it was applied to all the solutions in a population, it could lead to the premature convergence of the algorithm. Hence, we decided to use it as a mutation operator for the whole population and as repairing operator for the best element in the population. This means that, in each generation it has a small probability of being used, but it is always applied to the best chromosome in each population. The application of this operator always reduces or, at least, maintains the number of leftovers and it never increases the number of duties.

The operator works as follows: for each uncovered piece-of-work that is next to a covered one, build the set $\text{Possible}_\text{duties}$. Each duty in this set covers the uncovered piece-of-work and also all the pieces-of-work covered by one of the adjacent duties (duties that are immediately after or before the uncovered
piece-of-work). If this set is not empty, select one possible duty and replace one of the adjacent duties in the chromosome by this one. In the selection process we guarantee that no additional leftovers are introduced in the solution. The replacement is not always possible, since the new duty may cover other pieces-of-work already covered by other duties in the solution. When this happens, we must take into account if overcovers are allowed. In this case, the new duty must not violate the constraints that were imposed to the overcovered pieces-of-work.

The Example 6.6 shows how the Improve mutation operates on a solution without overcovers.

**Example 6.6 (Improve mutation operator example)** Figure 6-15 shows a solution with 8 duties and 4 uncovered pieces-of-work. We can see that the small piece-of-work between duties 1 and 5 could be covered if duty 1 could be extended or if duty 5 could start a little earlier. A similar situation occurs with the piece-of-work at the end of duty 5, and with the piece-of-work between duties 8 and 3. In Figure 6-16 we can see the result of the application of the Improve mutation operator to this solution. Duties 1 and 5 were replaced by duties 100 and 200 respectively, which are a little longer, and duty 8 was replaced by duty 300.

---

**Figure 6-15:** Part of a solution selected for application of Improve mutation

**Figure 6-16:** Result of the application of Improve mutation
The following example illustrates a situation where the Improve mutation operator is applied to a solution with overcovers.

Example 6.7 (Improve mutation example with overcovers) In Figure 6-17 we can see a solution with two overcovered pieces-of-work (both of them by duty 3) and one leftover. Applying the Improve mutation operator to this solution lead to the solution presented in Figure 6-18. In this solution, where duty 9 was replaced by duty 400, we managed to reduce the number of leftovers and also the number of overcovers.

Figure 6-17: Part of a solution selected for application of Improve mutation with overcovers

Figure 6-18: Result of the application of Improve mutation with overcovers

A more detailed description of the Improve mutation operator is presented in Algorithm 7.

Algorithm 7 $S = \text{Improve}(S)$

$Empty := \{i \in I : i \notin I_S \land (i - 1 \in I_S \land \text{bus}(i - 1) = \text{bus}(i)) \lor (i + 1 \in I_S \land \text{bus}(i + 1) = \text{bus}(i))\}$

$Possible\_duties = \{j \in J : \exists i \in Empty \cap I_j \land (i - 1 \in I_j \lor i + 1 \in I_j)\}$

$Eligible\_duties_{possible} = \{j \in Possible\_duties : j \text{ is eligible for } S\}$

while $\{Eligible\_duties_{possible} \neq \emptyset\)$
\[ j := \text{selection\_procedure}(\text{Eligible\_duties}_{\text{possible}}) \]
\[ S := \begin{cases} S \setminus \{s_1\}, & s_1 \in S \land i - 1 \in I_{s_1} \cap I_j \\ S \setminus \{s_2\}, & s_2 \in S \land i + 1 \in I_{s_2} \cap I_j \end{cases} \]
\[ S := S \cup \{j\} \]
\[ \text{Eligible\_duties}_{\text{possible}} = \{j \in \text{Possible\_duties} : j \text{ is eligible for } S\} \]
\[ \text{return } C_S \]

The **Reduce\_Empty** mutation operator

The Reduce\_Empty mutation operator is very similar to the Improve mutation operator but it allows the replacement of more than one duty. In fact, in order to cover an empty piece-of-work, two or more duties may have to be replaced. Consider the following example.

**Example 6.8 (Reduce\_Empty mutation operator)** Figure 6-19 shows part of a solution that was selected for the Reduce\_Empty mutation operator. We can see that this running board contains two duties and one uncovered piece-of-work. We cannot extend duty 213 to the right, since it becomes unfeasible (the overall duration becomes larger than the maximum allowed by the company: 10h). However, the empty piece-of-work could be covered by a feasible duty that starts an hour later, like duty 817 in Figure 6-20. But in order to replace duty 213 by duty 817, we had also to replace duty 15 by duty 6.

![Figure 6-19: Part of a solution selected for the application of Reduce\_Empty mutation](image)

![Figure 6-20: Result of the application of Reduce\_Empty mutation](image)

This operator involves a local search in the "neighborhood" of a leftover. This neighborhood is defined as the set of the duties contiguous to the leftover plus the duties contiguous to these. In the previous example, the neighborhood of the leftover we tried to cover is composed by duties 213 and 15.

This mutation operator is computationally expensive and is only applied to a small set of the best individuals in the population, according to a probability set by the planner.
The Reduce_Bus_Changes mutation operator

The Reduce_Bus_Changes mutation operator has been designed taking into account a real concern of some of the companies that do not want many vehicle changes in a given duty. In most companies this is a second order criterion, in the sense that it only becomes relevant after the other objectives are satisfied. After achieving a good solution, the planner tries to reduce the bus changes without worsening the existing solution. However, for some other companies, this is a criterion of major importance, since the planner almost never allows more than one bus change in a duty.

We could ask why do these planners do allow bus changes in the duties generation phase. In fact, they can impose that no duties with vehicle changes can be generated. However, this is an extreme position because some (only a few) vehicle changes are sometimes important to improve the solutions (regarding other criteria, of course).

The question concerning the number of vehicle changes has already been posed in section 5.5, when we have described the soft constraint (criterion) that tries to minimise the total number of vehicle changes. However, the selection of this constraint as a criterion is not effective enough when the number of vehicle changes is of major importance to the company. We must notice that this mutation operator behaves as a repairing operator and it is only applied to a small set of the best individuals in the population.

Example 6.9 (Reduce_Bus_Changes mutation operator) Consider the solution that is partially represented in Figure 6-21. We can see three buses with a total of seven duties, three of them with a vehicle change. The mutation operator simply removes from the solution all the duties with vehicle changes and tries to replace the empty pieces-of-work with as many as possible duties with no vehicle changes. The resulting solution is presented in Figure 6-22.

Figure 6-21: Part of a solution selected for the application of Reduce_Bus_Changes mutation
6.4.5 Population replacement strategies

The replacement of the existing chromosomes by the new ones is a fundamental stage of the evolution of a GA and there are several different strategies, that determine how individuals are replaced at each generation.

In our work we have always used a steady-state replacement\[45\] with elitism. In other words, at each generation a percentage of the chromosomes are replaced by the new individuals, and the fittest one always survives (elitism). The percentage of population that is replaced at each generation, called generation gap, is a parameter that can be dynamically adjusted by the user. By default, half of the population is replaced at each generation (the generation gap is 50%) but if this value increases until 100%, we have a generation replacement approach.

The selection of the individuals that will be replaced in a steady-state replacement strategy can be based on the following criteria:

1. Random selection: the chromosomes that will be replaced are randomly selected from the population.

2. Ranking selection: the chromosomes in the population are sorted according to their fitness and only the worst ones are replaced.

3. Heuristic procedures: in general, these procedures are a combination of the two previous ones, since they use the information about the fitness of chromosomes with some degree of randomness. Somehow, they are similar to selection operators, in the sense that chromosomes must compete to survive as well as they compete to reproduce. In our work, we have implemented two heuristic procedures for the population replacement: a binary tournament and a Boltzmann tournament.

In the Binary Tournament Replacement, a parent and a child are randomly selected and, if the fitness of the child is higher than the fitness of the parent, it replaces the parent in the population.
Otherwise, the parent remains in the population. The process is repeated until the generation gap is reached.

In the Boltzmann Tournament Replacement, we also randomly select a parent and a child. However, in this case, a child that is worse than the parent can still replace it according to a probability that varies with time. This process, usually applied in approaches that integrate Genetic Algorithms and Simulated Annealing [110], is based on the following idea: at initial generations, the probability of a bad child to replace a good parent should be high (near 50%). However, as the algorithm proceeds, this probability should decrease, meaning that it should become more difficult to loose the good individuals. The main goal of this strategy is to promote the population diversity and, consequently, to prevent premature convergence.

In our work, we have noticed that Ranking selection and heuristic procedures, such as the Binary Tournament Replacement or the Boltzmann Tournament Replacement, lead to premature convergence earlier than the Random selection. In order to keep the diversity of the population, we have therefore decided to use this last criterion.

6.5 The Aggregate model

In this section we present the structure of the Aggregate Model (Agg) model and the particular procedures that we have implemented for it, namely the Fitness Assignment and the Selection operator. This model, represented in Figure 6.23, is supported by the structure of a traditional GA.

After the generation of the initial population (Section 6.4.2), the chromosomes are evaluated according to a given criterion and its fitness value in the population is computed. The fitness function is described in Section 6.5.1. Then, a selection operator is applied to build a parents pool, with the individuals selected for reproduction. The size of this set depends on the selection process that is used. Such selection operators are described in Section 6.5.2.

After the application of the crossover operators (Section 6.4.3) to the parents pool, a new set is formed (the children pool), with the individuals resulting from recombination. These individuals may now be subjected to the mutation operators (Section 6.4.4). The next step is to replace part of the population with the new chromosomes, thus creating a new generation (Section 6.4.5). Finally, a termination criterion is applied and, if it is verified, the individual (solution) with highest fitness value and also its twins (those with the same values) are presented to the planner. If the termination criterion is not satisfied, the process continues until a satisfying solution or set of solutions is found. Due to the complex nature of the problem and since the GA is to be applied in real problems, the termination criterion is the number of generations as decided by the user. Then, the user analyses the solutions presented and, if he
is not satisfied with any of them, he continues running the algorithm for a new number of generations until he finally chooses a solution. Furthermore, at each generation, the user can change any parameter of the algorithm (except the size of the population) as well as the criteria.

![Diagram of the general structure of the Agg model](image)

Figure 6-23: General structure of Agg model

### 6.5.1 Evaluation and fitness assignment

The main difference between the Aggregate Model (Agg) model and the Non-Dominated (ND) model lies on the evaluation function and in the fitness assignment method. In the Agg model, the criteria selected by the user are merged into a single evaluation function according to the weights defined by the user (see Chapter 4). Then, the weights (in a scale ranging from 1 to 10) are normalised and, for all the solutions in the population, the value for each individual criterion is computed. Note that the weighting coefficients do not reflect proportionally the relative importance of the criteria unless all functions are expressed in units of approximately the same numerical values. Hence we compute the normalised objective function values, $eval_k (C_S)$, which are obtained by multiplying the raw values $eval_k (C_S)$ by a constant multiplier
$c_k$ that will scale properly the objectives [145], [36] (see Expression 6.1).

$$\text{eval}_k(C_S) = \text{eval}_k(C_S) \times c_k$$  \hspace{1cm} (6.1)

where $c_k = \frac{1}{R_k}$ and $R_k$ is the (approximate) range of criterion $k$ [94].

The evaluation function simply adds up the normalised weighted values of each criterion. Each chromosome $C_S$ is evaluated using the function defined by Expression 6.2, where $ncrit$ is the number of criteria, $w(k)$ is the weight associated to criterion $k$ and $\text{eval}_k(C_S)$ is the normalised value of $C_S$ in criterion $k$.

$$\text{eval}(C_S) = \sum_{k=1}^{ncrit} \frac{w(k)}{\sum_{i=1}^{ncrit} w(i)} \text{eval}_k(C_S)$$ \hspace{1cm} (6.2)

The criteria considered in the GA are the objective functions and also the soft constraints involving leftovers and duties. All these criteria have been presented in detail in Chapter 5.

The fitness function ranks the individuals in the population, according to their evaluation. In our problem, the fittest elements in the population have the smallest value in the evaluation function, since the evaluation function is a minimisation function. The fitness function is a (maximisation) function that assigns the highest value to the individual to the smallest evaluation.

There are several fitness assignment techniques that are deeply dependent on the selection procedure adopted. We have used a technique based on linear ranking [45] and a variation of this technique that we have called lexical linear ranking.

**Linear ranking**

In this technique, the individuals in the population are sorted according to their evaluation, and the fitness values are assigned in descending order using a linear function, from the best element to the worst. Consider that the elements in a population $P$ are sorted in ascending order of their evaluation. The fitness function, presented in Expression 6.3, assigns a fitness value to the chromosome $P(i)$, classified as the $i^{th}$ smaller in $P$ by the evaluation procedure.

$$\text{fitness}(P(i)) = a - b \times i, \text{ where } a \text{ and } b \text{ are constants}$$ \hspace{1cm} (6.3)

Linear ranking tries to prevent the premature convergence of the algorithm when fitness proportionate selection is used, balancing the contribution of the fittest individuals to the next generations.
Lexical linear ranking

We have slightly modified the linear ranking technique, this variation being referred here as lexical linear ranking, since we have observed that in each generation a considerable number of different individuals had the same evaluation. For example, in a population with 100 elements, we could have 15 or more individuals with the smallest evaluation. However, from the planner’s point of view they are distinct and it could be a hard task for him to analyze each solution in order to choose one. The linear ranking technique would arbitrarily choose one individual as the best solution and assign descending fitness values to the others. The lexical linear ranking helps to differentiate between solutions with the same evaluation, by considering an additional set of criteria that the algorithm takes into account in the fitness assignment process. Besides selecting the criteria for evaluation purposes, the planner can assign different levels of priority to the criteria that have not been selected. This will help to differentiate the solutions with the same value.

The lexical linear ranking technique is composed by two phases: in the first phase, the solutions are sorted according to their value. In the second phase, the whole population is divided into clusters formed by the individuals with the same value. Within each cluster, solutions are lexicographically sorted based on their evaluations according to the new set of criteria. Finally, Expression 6.3 is applied to all the individuals in the population considering the new ranking. It should be noted that, if the user does not specify the new set of criteria, the standard linear ranking is applied.

A simple case of the application of this technique is presented in Example 6.10.

Example 6.10 (Lexical linear ranking) Consider a problem with three criteria and a population with 10 individuals, with values as presented in Table 6.5. Criteria 1 and 2 were selected for evaluating the solutions and Expression 6.2 was used to compute the evaluation function. Then, criterion 3 was used to distinguish the solutions with equal values. The solutions were sorted and numbered in ascending order from the best to the worst according to their evaluation. The GA found 3 groups of solutions with the same evaluation (considering only criteria 1 and 2). At this moment, the solutions in each group are cannot be differentiated. In the second phase, criterion 3 was used to rank the solutions within each group and, from the best to the worst individual, the new ranking is: 3, 1, 2, 5, 4, 6, 9, 10, 7, 8.

<table>
<thead>
<tr>
<th>Weights</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crit.1</td>
<td>0.5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Crit.2</td>
<td>0.5</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Crit.3</td>
<td>-</td>
<td>5</td>
<td>10</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>0</td>
</tr>
</tbody>
</table>

| Evaluation | 7.5 | 7.5 | 7.5 | 12.5 | 12.5 | 12.5 | 25 | 25 | 25 | 25 |

Table 6.5: Evaluation of the elements in the population
6.5. THE AGGREGATE MODEL

The process can be easily understood by observing the 3D plot presented in Figure 6-24. In this figure, the elements of the population are represented according to their evaluation in the three criteria. We can easily identify the three groups and, for each group, the new ranking of the solutions is clear.

![3D plot of a population with criteria axes](image)

*Figure 6-24: Three criteria evaluation plot of a population*

6.5.2 Parent selection

After the evaluation and the fitness assignment of all elements in the population, we need to select some of them for reproduction. The fittest elements will have higher chances to be chosen. The selected elements will be the parents of the chromosomes of the next generation and thus, their features will have higher chances to survive. The selection operators applied to the Aggregate Model are the Roulette Wheel Selection, the Binary Tournament Selection and the Boltzmann Selection.

Roulette Wheel Selection [80] assigns to each chromosome a probability of selection that is proportional to its fitness. Hence, those chromosomes with higher fitness values have also higher probabilities of selection. The selection probability, \( p_{sel} (C) \), of chromosome \( C \) is given by Expression 6.4.
\[
    p_{\text{sel}}(C_S) = \frac{\text{fitness}(C_S)}{\sum_{C_S \in P} \text{fitness}(C_S)} \quad (6.4)
\]

In **Binary Tournament Selection** two individuals are randomly selected from the population. The individual with the highest fitness is selected for reproduction. The Binary Tournament Selection is a particular case (with \( k = 2 \)) of the **Tournament Selection** \([84]\), which consists of sampling \( k \) individuals from the population and choosing the fittest one.

Another selection operator, that will be further discussed in section 6.7, is the **Boltzmann Selection** \([110]\) operator, that is based on the Simulated Annealing paradigm \([147]\). At the first generations, individuals with low fitness values can be selected for reproduction but, as the algorithm evolves, the selection pressure increases. This approach promotes the diversification of the population at the initial stages, trying to avoid premature convergence. At the following stages, increasing intensification will promote the reproduction of the fittest elements. The selection probability \( p_{\text{sel}}(C_S, t) \) of a chromosome at time \( t \) depends on a temperature parameter \( T \) that linearly decreases with time \( (T = T_{\text{max}} - \text{decr\_rate} \times t) \). This probability is defined in Expression 6.5, where \( \text{fitness}_P \) corresponds to the average fitness of current population \( P \).

\[
    p_{\text{sel}}(C_S, t) = e^{\frac{\text{fitness}(C_S) - \text{fitness}_P}{T}} \quad (6.5)
\]

We can notice that as the temperature decreases, the difference will increase between the selection probability of an individual with a high fitness value and that of another individual with a low fitness value.

Experiments with these three selection operators have shown that the Boltzmann Selection shows a better performance in preventing the premature convergence of the algorithm (see Section 6.7). However, it involves the definition of an additional parameter, \( T \), and it is computationally more expensive than the Binary Tournament. Nevertheless, we have used it as the default selection strategy for the Agg model.

### 6.6 The model based on non-dominated solutions

In this section we describe the model based on non-dominated solutions (the NDmodel) and its structure is presented in Figure 6-25. This model was inspired in the SPEA algorithm proposed by Zitzler \([162]\), which has emerged as one of the leading evolutionary algorithms for multiobjective optimisation. The main difference between the ND model and the Agg model is that it uses the concept of Pareto dominance to rank the individuals in the population. This will imply considerable changes in the fitness assignment
6.6. THE MODEL BASED ON NON-DOMINATED SOLUTIONS

procedure and in the Selection operator.

Many GAs based on Pareto dominance maintain, in each generation, as a set of candidate solutions, the non-dominated solutions found so far [36], [163]. In practice, this corresponds to an elitist strategy applied to a multiobjective problem (see Section 4.5). However, when compared to a single objective model as the Agg, the application of elitism to a multiobjective model is considerably more complex. In fact, instead of a single best solution (or a small set of twin solutions), we have a set of solutions that can be considerably larger than the population itself.

Handling this potentially huge set involves two main issues:

1. the *storage* process: the set of non-dominated solutions can be incorporated into the population or, in alternative, it can be externally stored;

2. the *management* process: this question involves issues such as keeping the set of non-dominated solutions within a manageable dimension as well as the assignment of fitness values to the chromosomes in this set.

In the ND model, we have decided to externally store the set of non-dominated solutions, following the SPEA approach. This set, denoted as the *External Set* is updated at each generation. In order to control its size, two clustering procedures have been implemented. In Section 6.6.1, we describe how this set is built and updated. In Section 6.6.2, we present the evaluation and the fitness assignment procedures. We have modified the fitness assignment procedure proposed by Zitzler, by adding to it some new concepts. The Selection operator is described in Section 6.6.3.

### 6.6.1 The external set

The external set is a subset of all the non-dominated solutions found so far. It is updated at the end of each generation by adding the new non-dominated solutions and by removing the solutions that became dominated by the new ones. The maximum size of the external set, \( N' \), is predefined by the user, but it can be changed at each generation. When the number of non-dominated solutions is larger than this value, a filtering procedure is applied, in order to reduce the number of non-dominated solutions in the external set.

The number of non-dominated solutions can grow very fast and become very large, often several times the size of the population itself. This number and its growth rate depend on the particular problem, but also on the set of objective functions. In particular, if we choose to minimise the number of leftovers and the number of duties, the number of non-dominated solutions will probably be larger than if we choose to minimise the total cost of the solution and the deviation to the desired mean duration of the duties. In fact, since the objective space of the first vector of objective functions is composed of integers, we
might obtain several non-dominated solutions with the same values for the two objective functions. On the other hand, the objective space corresponding to the second vector is a continuous set and, in this case, the number of solutions with equal values for both objective functions would probably be smaller.

Since it would be impracticable to present the user all the non-dominated solutions found, we have chosen to let him decide the maximum number of solutions he wants to analyze simultaneously. Moreover, the size of the external set affects the running time and also the search speed. In fact, as described in Section 6.6.3, the external set is used by the selection operator and, if it is too large, the selection pressure can reduce and slow down the search.

We have implemented two different filtering procedures for selecting the non-dominated solutions in the external set. The first is based on the same cluster analysis technique used by SPEA [120], that tries to guarantee a well distributed representation of the Pareto set. The other is a new lexicographic sorting
6.6. THE MODEL BASED ON NON-DOMINATED SOLUTIONS

process based on a vector of priorities defined by the user.

The following definitions adapt the concept of Pareto dominance given in Chapter 4 and the distance between two points to the genetic algorithms context.

**Definition 6.5** Let $P$ be a population, $P = \{C_{S_1}, C_{S_2}, \ldots, C_{\text{chrom}}\}$. $F : (f_1, f_2, \ldots, f_K)$ is a vector with the $k$ minimisation objective functions for the BDSP and $F(C_S)$ is the objective vector for $C_S$. For any $C_{S_i}$ and $C_{S_j}$,

$F(C_{S_i}) = F(C_{S_j})$ iff $f_k(C_{S_i}) = f_k(C_{S_j}), \forall k \in \{1, 2, \ldots, K\}$

$F(C_{S_i}) \leq F(C_{S_j})$ iff $f_k(C_{S_i}) \leq f_k(C_{S_j}), \forall k \in \{1, 2, \ldots, K\}$

$F(C_{S_i}) < F(C_{S_j})$ iff $F(C_{S_i}) \leq F(C_{S_j}) \land F(C_{S_i}) \neq F(C_{S_j})$

**Definition 6.6** (Pareto dominance in a population)

$C_{S_i}$ dominates $C_{S_j}$ ($C_{S_i} < C_{S_j}$) iff $F(C_{S_i}) < F(C_{S_j})$

$C_{S_i}$ weakly dominates $C_{S_j}$ ($C_{S_i} \leq C_{S_j}$) iff $F(C_{S_i}) \leq F(C_{S_j})$

$C_{S_i}$ is indifferent to $C_{S_j}$ ($C_{S_i} \sim C_{S_j}$) iff $F(C_{S_i}) \not< F(C_{S_j}) \land F(C_{S_i}) \not> F(C_{S_j})$

**Definition 6.7** (Non-dominated solution) A solution $C_S$ is said to be non-dominated regarding $P$ iff $\not\exists C \in P_i, C_S < C$.

**Definition 6.8** (Non-dominated set) The set $ND(P_i) \subset P_i$ is the set of non-dominated solutions regarding $P_i$: $ND(P_i) = \{C_S \in P_i : \not\exists C \in P_i, C_S < C\}$.

**Definition 6.9** (Distance in the decision space) Given two individuals $S_i$ and $S_j$, the distance function in the decision space is given by the number of distinct values in the correspondent genes of the two chromosomes:

$$d_D(i, j) = |\{t : C_{S_i}[t] \neq C_{S_j}[t], t = 1 \ldots m\}|.$$  \hspace{1cm} (6.6)

For example, consider the two following chromosomes representing solutions $S_1$ and $S_2$:

$C_{S_1} = \{2, 2, 2, 3, 3, 1, 8, 8, 4, 4, 3, 4, 5, 0, 5\}$ and

$C_{S_2} = \{6, 6, 6, 3, 6, 6, 8, 8, 4, 4, 3, 4, 7, 7, 7\}$.

The distance in the decision space between $S_1$ and $S_2$ is given by:

$$d_D(1, 2) = |\{1, 2, 3, 6, 7, 14, 15, 16\}| = 8.$$

**Definition 6.10** (Distance in the objective space) Given two individuals $S_i$ and $S_j$, the distance function in the objective space is the standard euclidean distance:

$$d_O(i, j) = \|F(C_{S_i}) - F(C_{S_j})\|_2 = \sqrt{\sum_{k=1}^{K} (f_k(C_{S_i}) - f_k(C_{S_j}))^2}$$  \hspace{1cm} (6.7)
Building the external set

For a given generation \( t \), the external set, denoted by \( P'_t \), is built in three stages:

**Initialization:** Build the set \( \overline{P}_t \), by performing the union between the external set of the last generation, \( P'_{t-1} \), and the set of non-dominated solutions found in the population of generation \( t \),

\[
P_t : \overline{P}_t = P'_{t-1} \cup ND(P_t) ; \text{ when } t = 0, \overline{P}_0 = ND(P_0).
\]

**Removing dominated solutions:** Remove the dominated solutions regarding \( \overline{P}_t \) and build the set \( ND(\overline{P}_t) \subset \overline{P}_t \).

**Filtering solutions:** If the number of elements of \( ND(\overline{P}_t) \) is larger than the maximum size of the external set, \( |ND(\overline{P}_t)| > N' \), apply a filtering procedure to \( ND(\overline{P}_t) \) and build the external set of generation \( t \): \( P'_t = \text{Filtering}(ND(\overline{P}_t)) \).

In order to control the size of the external set, we have implemented two filtering procedures for extracting a subset of non-dominated solutions. The size of the external set, \( N' \), is defined by the user and it can be dynamically changed at the end of each generation.

One of these procedures is based on the Average Linkage method, a known cluster analysis technique [120], while in the other one, called Lexicographic Selection, solutions are sorted based on a vector of priorities defined by the user.

**Average Linkage method**

The basic idea of the Average Linkage method is to divide the set of non-dominated solutions into several subsets, called clusters, each one representing part of the search space. From each cluster, one chooses a single solution, the centroid, to be an element of the final external set. This process involves the following four phases:

**Initialization:** each cluster of the initial set of clusters, \( Cl \), is composed by a single non-dominated solution:

\[
Cl = \bigcup_{C_S \in ND(\overline{P}_t)} \{C_S\}
\]

**Defining clusters:** while the number of clusters in \( Cl \) is larger than the maximum size of the external set, \( |Cl| > N' \),
6.6. THE MODEL BASED ON NON-DOMINATED SOLUTIONS

(i) Compute the distance between all pairs of clusters, \( d_{cl} (c_{li}, c_{lj}) \), given by the average distance between pairs of elements across two clusters:

\[
\forall i, j = 1 \ldots |Cl|, d_{cl} (c_{li}, c_{lj}) = \frac{1}{|c_{li}| \cdot |c_{lj}|} \sum_{i \in c_{li}, j \in c_{lj}} d(i, j),
\]

where the distance function \( d \) can be either the distance \( d_D \) or \( d_O \).

(ii) Find the two clusters \( c_{li} \) and \( c_{lj} \) with minimum distance \( d_{cl} \). These two clusters are merged into a larger cluster and the set of clusters is updated:

\[
Cl = Cl \setminus \{c_{li} \cup c_{lj}\} \cup \{c_{li}, c_{lj}\}
\]

Choosing the representative element in each cluster: For each cluster \( c_{li} \), choose a single representative element, given by the cluster centroid \( \bar{c}_i \). The cluster centroid is the element with minimum average distance in the decision space to all other elements in the cluster:

\[
\bar{c}_i \in c_{li} : d_{\bar{c}_i} = \min \left\{ d_k : d_k = \frac{1}{|c_{li}| - 1} \sum_{k=1}^{|c_{li}|} d_D (k, k_1), k, k_1 = 1 \ldots |c_{li}| \right\}
\]

Building the external set: the external set is composed by the centroids of all clusters:

\[
P^e_t = \bigcup_{\bar{c}_i \in c_{li}} \{\bar{c}_i\}
\]

This method is based on the concept of distance between two solutions and it can be applied with any of the distance functions defined above. Using the distance function \( d_D \) (6.6), we expect to obtain a good representation of the solution space. If instead we use the distance function \( d_O \) (6.7), we expect to obtain a good representation of the objective space.

For choosing the cluster centroid, we always use the distance in the decision space \( (d_D) \), as in practice many different solutions have the same values for the objective functions and differentiation is better performed in the decision space. The clustering procedure puts together these solutions in the same cluster. Hence, using the \( d_O \) distance function to find the centroid of each cluster will no longer help to differentiate the solutions, since they all have the same value.

Naturally, the clustering procedure only applies if the number of non-dominated solutions is larger than the maximum allowed size of the external set.

Example 6.11 illustrates the application of the Average Linkage method and the difference between the two distance functions defined.
Example 6.11 Consider the 5 non-dominated solutions presented in Table 6.6, obtained for a problem with 16 pieces-of-work. For each solution, we present the corresponding chromosome (PWC coding) and the values for the three criteria selected. Criterion $C_1$ is related to the uncovered schedule (in hours), $C_2$ to the number of duties and $C_3$ to the number of leftovers (all to be minimised).

<table>
<thead>
<tr>
<th>N.sol</th>
<th>Chromosomes for non-dominated solutions</th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 2 2 3 3 1 1 2 2 4 4 3 4 5 0 5</td>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>6 6 6 9 3 6 6 0 0 4 4 3 4 7 7 7</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>6 6 6 9 9 6 6 9 0 4 4 0 4 5 0 5</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>8 8 8 9 9 1 1 9 0 4 4 8 4 7 7 7</td>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2 2 2 3 3 0 0 2 2 4 4 3 4 7 7 7</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.6: Chromosomes and criteria values for 5 non-dominated solutions

Consider also that the maximum size of the external set is 2, $N' = 2$, so we have to select 2 solutions from the set of solutions presented.

Table 6.7 shows the different external sets that can be obtained when we use $d_D$ or $d_O$ distance functions.

If we apply $d_D$ distance function for building the clusters and for choosing the centroid of each cluster, two possible external sets can be built, one of them composed by solutions 3 and 5 and another composed by solutions 4 and 5, since the centroid of the second cluster can be either solution 3 or solution 4. In fact, if a cluster is composed by two elements, any one of them can be used as the centroid, since they both have the same minimum average distance. The algorithm would arbitrarily choose one of them.

Using the $d_O$ distance function for building the clusters and choosing the centroids, both $c_{l_1}$ and $c_{l_2}$ have two possible candidates for centroids and the algorithm would produce one of four alternative external sets.

We propose to use $d_O$ for building the clusters and $d_D$ for choosing the centroids. In this case, we could obtain a single external set. In conclusion, the identification of the clusters and corresponding centroids can be done more clearly if the $d_D$ function is used, alone or combined with the $d_O$ function.

**Lexicographic Selection**

<table>
<thead>
<tr>
<th>Distance</th>
<th>Clusters</th>
<th>Centroids</th>
<th>Possible external sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_D$</td>
<td>$c_{l_1} = {1, 2, 5}$</td>
<td>$\bar{c}_1 = 5$</td>
<td>$P'_l = {3, 5}$ or $P'_l = {4, 5}$</td>
</tr>
<tr>
<td></td>
<td>$c_{l_2} = {3, 4}$</td>
<td>$\bar{c}_2 = 3$ or 4</td>
<td></td>
</tr>
<tr>
<td>$d_O$</td>
<td>$c_{l_1} = {1, 4}$</td>
<td>$\bar{c}_1 = 1$ or 4</td>
<td>$P'_l = {1, 2}$ or $P'_l = {1, 5}$</td>
</tr>
<tr>
<td></td>
<td>$c_{l_2} = {2, 3, 5}$</td>
<td>$\bar{c}_2 = 2$ or $5$</td>
<td>$P'_l = {2, 4}$ or $P'_l = {4, 5}$</td>
</tr>
<tr>
<td>$d_0 + d_D$</td>
<td>$c_{l_1} = {1, 2, 4, 5}$</td>
<td>$\bar{c}_1 = 5$</td>
<td>$P'_l = {3, 5}$</td>
</tr>
<tr>
<td></td>
<td>$c_{l_2} = {3}$</td>
<td>$\bar{c}_2 = 3$</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.7: Applying different distance functions in the clustering process
This method is analogous to the Lexical Linear Ranking, presented in Section 6.5.1 for the Agg model. The non-dominated solutions in the external set are ranked according to a set of priorities defined by the user. The set of criteria used for ranking the solutions in the external set can include criteria that were not selected for evaluating the solutions.

The Example 6.12 shows how Lexicographic Selection could be applied to the same set of non-dominated solutions that were used in Example 6.11.

Example 6.12 Consider the set of non-dominates solutions presented in Example 6.11. Suppose that we have selected a new criterion $C_4$ (with the highest priority) that was not chosen for evaluating the solutions. The solutions are lexicographically sorted according to the priorities assigned to the criteria as follows: $\{4, 2, 3, 1, 5\}$. Considering that the maximum size of the external set is 2, the external set will be composed by the first two solutions in this rank, $\{4, 2\}$.

<table>
<thead>
<tr>
<th>N.sol</th>
<th>Priority</th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_3$</th>
<th>$C_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 6.8: criterion priorities and values for 5 non-dominated solutions

6.6.2 Evaluation and fitness assignment

Fitness assignment in the model based on Non-Dominated solutions (ND) is very different from that of the Aggregate model (Agg). While in the Agg model, all the criteria selected by the user are merged into a single evaluation function according to the weights defined by the user, in the ND model, for each chromosome we have a vector with the values for each objective function.

Inspired by the SPEA approach [162], [164], we rank the individuals in the population and also the individuals in the external set, since in the selection process both sets of solutions will be used. Two different fitness functions are defined, one for the population and another one for the external set.

We have introduced a new concept, referred here as population density which can be better understood if we first consider Example 6.13.

Example 6.13 (Population density) Consider a maximisation problem with two objectives $f_1$ and $f_2$. For this problem, consider two different populations pop$_1$ and pop$_2$ as presented in Figure 6.26. The solutions in the external set are represented by letters (A, B, C, D) while the solutions in the population
are represented by numbers (from 1 to 9). Analyzing both graphics we can observe that in a) the solutions are closer to each other than in b), where the solutions are more distributed.

![Figure 6-26: Example of two different populations for a maximization problem with two objectives](image)

In fact, the total number of times that the solutions in a) are dominated by other solutions is 43, while in b) this number is only 25. This intuitive concept is called density of the population, and will be formally defined below.

Loosely speaking, we should give more opportunities to reproduce to solution D in a) by assigning it a higher fitness value, in order to spread the solutions in the search space.

Without loss of generality, we consider here that all criteria are minimisation functions, and that the fitness is a maximisation function (the best individuals have the highest fitness values).

Let $P_t$ be a population of size $N$ and $P'_t$ the corresponding external set with size $N'$.

**Definition 6.11 (Population density)** Let us define the set of all dominance relations that occur in $P_t$ as

$$\delta^t = \{(k_1, k_2) \in (P'_t \cup P_t) \times P_t : k_1 \preceq k_2\}$$

The number of dominance relations in the population at generation $t$ is defined as the density of the population and is given by $|\delta^t| \leq (N' + N - 1) \times N$.

The external set is composed by non-dominated solutions regarding the current population. In order to rank the solutions in this set, we have to define a relation between the elements in the external set and those in the current population. For each solution in the external set we take into account the following two sets:

(i) the set of solutions it dominates and

(ii) the set of solutions that also dominate the solutions in (i).
In order to obtain a good representation of the potentially non-dominated solutions, each solution in the external set should dominate a small number of solutions in the population and each of these dominated solutions should in turn be dominated by only a few other solutions.

Loosely speaking, we want to penalise those solutions in the external set that "share" its dominance relations with many other solutions. This concept, that we have called dominance sharing is formally defined below.

For each solution $k$ in the $P_t \cup P'_t$, we define the projection of $\delta^t$ in $k_1$ as

$$D_k = \{(k_1, k_2) \in \{k\} \times P_t : k \leq k_2\}.$$

$D_k \subset \delta^t$ is the set of the solutions in the $P_t$ dominated by $k$. Note that $|D_k| \leq N$ if $k \in P'_t$ and $|D_k| \leq N - 1$ if $k \in P_t$. Note also that $|D_k|$ is analogous to strength concept in SPEA [164].

For each solution $j$ in the population $P_t$, we define the projection of $\delta^t$ in $k_2$ as

$$Td_j = \{(k_1, j) \in (P'_t \cup P_t) \times \{j\} : k_1 \leq j\}.$$

$Td_j \subset \delta^t$ is the set of solutions that dominate $j$ and $|Td_j| \leq N + N' - 1$.

Note that $\sum_{k \in P_t \cup P'_t} |D_k| = \sum_{j \in P_t} |Td_j| = |\delta^t|$.

**Definition 6.12 (dominance sharing of a solution)** The dominance sharing of a solution $k \in P_t \cup P'_t$ is given by the function $ds : P_t \cup P'_t \rightarrow [0, N \times N']$

$$ds(k) = \sum_{(k, j) \in D_k} |Td_j|$$

The fitness function for the elements in the external set is defined as follows:

$$fit_P^t : P'_t \rightarrow [1, 2]$$

$$fit_P^t(i) = 1 + \frac{N \times (N + N' - 1) - ds(i)}{N \times (N + N' - 1)} \quad (6.8)$$

In Expression 6.8, the fitness of an element of the external set decreases as the corresponding dominance sharing increases. The idea is to spread the non-dominated solutions as evenly as possible through the solutions space.

Moreover, we added 1 to the value of the fitness in order to guarantee that the elements of the external set always have better fitness values than the elements of the current population.
In order to rank the solutions in the population our main goal was to assign higher fitness values to those solutions that:

(i) are dominated by a small number of solutions and

(ii) each of these "dominant" solutions has a small dominance sharing value.

Hence for each solution in the population, we compute the sum of the dominance sharing values of the solutions dominating that solution. The resulting value, that we have called total sharing, is formally defined below:

**Definition 6.13 (total sharing of a solution)** The total sharing of a solution \( j \in P_i \) is given by

\[
t(j) = \sum_{(k,j) \in T_d} ds(k)
\]

The fitness of an element of the population is defined by Expression 6.9.

\[
\text{fit}_{P_i}(j) = \frac{\sum_{k \in P_i} t(k) - t(j)}{\sum_{k \in P_i} t(k)}
\]

In order to illustrate the concepts defined above, consider the Example 6.14.

**Example 6.14 (Dominance sharing and fitness assignment)** Consider the example presented in Figure 6.27 for a maximisation problem with two objectives \( f_1 \) and \( f_2 \). The external set is composed by 4 solutions \((A, B, C \text{ and } D)\) and the population has 9 solutions (numbered from 1 to 9). We can see that while \( D \) only dominates two solutions, both solutions \( B \) and \( C \) dominate six solutions.

In Table 6.9 we present a matrix \( A(N, N + N_r) \) where each entry \( a(i,j) \) has the value 1 if \( i \) is dominated by \( j \). The table also contains three additional rows (for \( |D_k|, ds(k) \text{ and } fit_{P_i}(k) \)) and three additional columns (for \( |T_d|, t(j) \text{ and } fit_{P_i}(j) \)). The population density is 38. We can see that \( D \) has the lowest dominance sharing \((ds(i))\) and thus the highest fitness value. Although \( B \) and \( C \) dominate the same number of solutions, solution \( B \) has a higher dominance sharing value. Hence, solution \( C \) has a higher fitness value than solution \( B \).

Concerning the solutions in the population, we can see that solution 6 has the highest value for \( |T_d| \), since it is dominated by 10 (of 13) of the total set of solutions and also the higher total sharing...
value. On the other hand, solutions 1, 3 and 9 have the lowest values for \( |TD_j| \), since they are dominated by a single solution (in the external set). Total sharing help us to differentiate these three solutions and thus to assign solution 1 the highest fitness value.

|   | A | B | C | D | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | \( |TD_j| \) | \( t(j) \) | \( fit_{P_2} \) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 |   |   |   |   |   |   |   |   |   |   |   |   | 1 | 24 | 0.906 |
| 2 | 1 | 1 |   |   | 1 | 1 | 1 |   |   |   |   |   |   | 1 | 4 | 114 | 0.555 |
| 3 | 1 |   |   |   |   |   |   |   |   |   |   |   |   | 1 | 34 | 0.867 |
| 4 | 1 | 1 |   |   | 1 | 1 |   |   |   |   |   |   |   | 1 | 4 | 130 | 0.492 |
| 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |   |   |   |   |   |   | 1 | 9 | 230 | 0.102 |
| 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |   |   |   | 1 | 1 | 1 | 10 | 247 | 0.035 |
| 7 | 1 | 1 | 1 | 1 | 1 |   |   |   |   |   |   |   | 1 | 6 | 171 | 0.332 |
| 8 | 1 |   |   |   |   |   |   |   |   |   |   |   |   | 2 | 63 | 0.754 |
| 9 | 1 |   |   |   |   |   |   |   |   |   |   |   |   | 1 | 32 | 0.875 |

| \( D_{P_2} \) | 4 | 6 | 6 | 2 | 3 | 1 | 5 | 2 | 0 | 0 | 1 | 3 | 5 | 38 | 256 |
| \( ds(k) \)  | 24 | 34 | 32 | 16 | 23 | 9 | 33 | 19 | 0 | 0 | 10 | 25 | 31 | 256 |

| \( fit_{P_2} \) | 1.78 | 1.69 | 1.70 | 1.85 |

Table 6.9: Computing the fitness of the population and of the external set

In conclusion, the fitness assignment process in the ND model takes into account the following issues:

(i) the number of dominance relations in the population (population density);

(ii) the number of dominance relations "shared" with other solutions (dominance sharing);
(iii) the sum of the dominance sharing values of the solutions (in the external set and in the population) that dominate each solution in the population (total sharing).

6.6.3 Parent selection

The selection operators for the ND model are distinct from those of the Agg model since, in the ND model we must also consider the elements of the external set. Since the elements in the external set are the best individuals found so far, we should avoid a systematic selection of parents belonging to the external set. Hence, in the ND model we have not considered the Roulette Wheel Selection technique, since in this technique the selection probability of an individual is proportional to its fitness.

Therefore, we have only considered the Binary Tournament Selection and the Boltzmann Selection. The application of these two selection strategies to the ND model is very similar to that presented in Section 6.5.2, but now the individuals called to compete in each tournament are randomly selected from a pool composed by the elements of the external set and those of the population. In our experiments, we have found that the Boltzmann Selection clearly outperformed the Binary Tournament Selection in preventing the premature convergence of the algorithm and therefore it was selected as the default selection operator for the ND model.

6.7 Diversification strategies

A genetic algorithm is composed of intensification and diversification stages that coexist, although in different degrees, as the algorithm is running. Intensification (or exploitation) aims at achieving and keeping a better solution (usually through crossover). Diversification (or exploration) is meant to find new promising regions of the search space (usually through mutation), even if those regions are composed by low fitness individuals. Initial populations are naturally diversified, so the first main objective is to produce better solutions. The best individuals survive and reproduce, and the algorithm will hopefully reach some promising areas of the search space. Then, along with the exploitation of these promising areas, the population diversity must be kept and other interesting areas should be searched. The trade-off between diversification and intensification becomes crucial now. As the algorithm evolves, it becomes more difficult to maintain diversity, since the selection pressure tends to bias the search to the already known regions, that have the best solutions found so far.

Strategies that promote diversification must be applied, in order to prevent the premature convergence of the algorithm to a local optimum. In our work, keeping the population diversity deserves special attention since the BDSP can be included in the particularly hard class of deceptive problems. A deceptive problem has a global optimum and several local optimum, and low-order, low-fitness schemata
are needed to build an higher-order and higher-fitness schema [79]. For these problems it is very difficult to jump from one region of the search space to another, and the population diversity is easily lost.

In this section we present and discuss several strategies that we have implemented in order to promote diversification and to prevent premature convergence. Any one of these strategies can be applied to both the Agg and the ND models. According to the particular problem instance we are trying to solve, the diversification strategies can vary or be combined differently. Also, we do not intend to claim or prove that one specific strategy performs better than another one. Instead, we have tried to provide our software application with some tools to slow down the selection pressure and thus preventing and controlling premature convergence.

Finally, it is important to notice that the problems are not equally affected by premature convergence. Apparently, there are certain aspects that influence the role of diversification of a GA. The size of the population is recognised as a relevant issue [46]. Small populations are more likely to converge prematurely than large ones. Some other aspects are related to the BSDP in particular, namely the structure of the problem and the set of chosen criteria. Diversification on small density problems can be harder to achieve than on high density problems. The set of chosen criteria is another issue to consider in the analysis of premature convergence. In fact, some combinations of criteria seem to promote diversification better than others. These aspects have not been not deeply studied in our work but we consider they are an interesting topic for further analysis.

6.7.1 Boltzmann selection mechanisms

One of the most promising strategies to promote population diversity is to apply Simulated Annealing techniques within a GA. In particular, the application of a Boltzmann selection mechanism tries to keep the convergence properties of Simulated Annealing using, at the same time, the implicit parallelism of GAs. In this strategy, that we have called Boltzmann Tournament Replacement (see Section 6.4.5), a Boltzmann distribution is associated to a cooling schedule to select the chromosomes that will remain in the population.

The elements of the current population and the new children compete with each other to be part of the population. A Boltzmann trial is carried out between two randomly selected individuals $i$ and $j$, in which $i$ wins with probability

$$\frac{1}{1 + e^{\frac{(f_j - f_i)}{T}}}$$

(6.10)

where $T$ is the temperature and $f_i$ is the fitness of solution $i$ (to be maximised).
When the temperature $T$ is high, $i$ and $j$ win with nearly equal probabilities but, as $T$ decreases, the better of the two solutions almost always wins. The idea is to slow down the selection pressure, allowing that at the initial generations all individuals have similar probabilities of survival. As the algorithm evolves (and temperature decreases) the best individuals have higher probabilities of survival.

Clearly, the cooling schedule (the mechanism that decreases temperature) is a key process in the effectiveness of this approach. In our work the initial temperature is a parameter set by the user and the temperature is decremented multiplying it by a positive constant less than 1. The temperature is kept constant for a certain number of generations (this number is also a parameter set by the user) and then the cooling schedule is applied. We should notice than when the Boltzmann selection mechanism is used in the population replacement, a random selection of the parents is performed.

6.7.2 Duplicates

An important decision concerning the population replacement strategy arises when one of the new individuals is "identical" to another individual that already exists in the population. Should this new chromosome be inserted into the population if he does not add any relevant information to the population?

We should emphasise that we differentiate between genotypically and phenotypically identical individuals. Genotypically identical individuals (duplicates) have the same genetic representation (they are exact copies of each other) while phenotypically identical individuals (twins) have the same values for all the objective functions considered, but they can be composed by different sets of duties. In our approach, we consider that two individuals are "identical" only if they are duplicates.

If we allow the existence of duplicates in the population, the number of identical individuals will grow rapidly and premature convergence is likely to occur.

However, a procedure to detect if an individual already exists in the population is rather time consuming and after the initial generations, producing new original chromosomes becomes more and more difficult, and each new population becomes harder to obtain.

In our work, allowing duplicates is an option of the user (by default, they are allowed) that can be changed at any iteration. Hence, he can control the convergence of the algorithm, by slowing or increasing the selection pressure as the algorithm runs.

6.7.3 Dynamic crossover and mutation rates

The traditional practice in the implementation of genetic algorithms is to assign fixed crossover and mutation rates, at the beginning of the algorithm. Several authors [46], [45] suggest that if crossover and mutation rates vary with time, premature convergence can be delayed or even avoided. At the
first generations, the crossover rate should be high (greater than 0.5) and the mutation rate should be small (less than 0.05). The crossover rate should progressively decrease, to slow intensification, and the mutation rate should increase, to promote diversification. The variation of the crossover and mutation rates should depend on the population diversity. Hence, a measure of the population diversity must be found.

In our work, the user can assign different rates to each crossover and mutation operator. Then, he can choose between static or dynamic crossover and mutation rates. Using static rates, the initial values remain unchanged throughout the algorithm or until the user explicitly changes them.

Dynamic rates can be applied to all three crossover operators, but only to the Destroy mutation operator. As presented in Section 6.4.4, we have divided the mutation operators into two groups, namely destroying mutation operators (Destroy operator) and repairing mutation operators (Improve, Reduce_Empty and Reduce_Bus_Changes operators). Repairing operators were specially designed for improving a solution under certain conditions. They are not applied to the whole population, but only to a small set of solutions.

For dynamic rates, we have experimented two different approaches: the first one is based on the entropy of the population and the other is based on the performance of the crossover operators.

**Dynamic rates based on the entropy of the population**

In this approach we measure the population diversity through a function, referred here as *entropy rate*, that is computed as follows:

Let $J$ be the set of generated duties, $|J| = n$, and let $J_i$ be the subset of $J$ composed by all the duties that, at a given generation $t$, belong to the solutions in the current population $P_i$. Hence, $J_i = \bigcup_{S_i \in P_i} S_i$. Let $n_i \leq n$ be the cardinal of $J_i$. The *entropy rate* of population $P_i$, is given by expression 6.11.

$$
entropy(P_i) = \frac{n_i}{n}
$$

(6.11)

The initial entropy depends on the population size, in the sense that for large populations a higher number of different duties would probably exist. At the initial generations the entropy is high (near 1) and it gradually decreases, getting closer to 0, as the population loses diversity. After each generation, the entropy of the population is calculated and, according to its value, the crossover and mutation rates are updated.

For this purpose, we have defined a *threshold for the entropy*. When the entropy of the population is under that threshold (Case 1), the crossover rates decrease and the mutation rates increase. On the contrary, if the entropy lies above the threshold (Case 2), the crossover rates increase and the mutation
rates decrease.

The entropy threshold gives the value under which we consider that the population is converging and loosing diversity. By default, we have defined the entropy threshold as half of the average entropy but this value can be tuned as needed.

Next, we present the functions that we have used to increase or to decrease the crossover and mutation rates, according to the entropy of the population. We have chosen these particular functions because they have shapes that match our needs in the [0, 1] interval, concerning the trade-off between the variation of the entropy and the crossover and mutation rates. However, any other function with similar features could have been chosen.

**Case 1: Low population diversity**

When the entropy rate is low \( \text{entropy}_t \leq \text{entr\_threshold} \), we face a serious risk of premature convergence in the next generations. Hence, we must decrease crossover rates in order to control intensification and increase the mutation rates to promote diversification.

For changing the crossover rates, we have used expression 6.12. According to this expression, the new crossover rate \( \text{cross\_rate}_{t+1} \) is calculated multiplying the current value \( \text{cross\_rate}_t \) by a factor involving the current entropy rate. Note that this function assures that the new crossover rate is always less than the current one and that it is kept in the [0, 1] interval.

\[
\text{cross\_rate}_{t+1} = \text{cross\_rate}_t \times \frac{\text{entropy}_t}{\text{entropy}_t + 0.05}
\]

(6.12)

In Figure 6-28, we can see how the crossover rate is updated according to the variation of entropy and the current crossover rate. The new crossover rate will decrease faster when the entropy is lower.

![Figure 6-28: 3-D plot of the function for decreasing the crossover rates](image)
6.7. DIVERSIFICATION STRATEGIES

For the increase of the mutation rate, we have used expression 6.13. In this function, plotted in Figure 6-29, as the entropy gets smaller, the increment of the mutation rate increases. At the same time, when the current mutation rate is small, the increment is larger than when the current mutation rate is high.

\[
mut_{rate_{t+1}} = mut_{rate_t} \times entropy_t + (1 - entropy_t) \times \frac{\ln(0.2 + mut_{rate_t}) + 2}{\ln(1.2) + 2}
\]  

(6.13)

Figure 6-29: 3-D plot of the function for increasing the mutation rates

Case 2: High population diversity

When the entropy of the population is high, we can slow down mutation, since it is no longer necessary to diversify the population and we can increase the crossover rates to promote the intensification of the search. We have used the function in expression 6.14 to decrease the mutation rates.

\[
mut_{rate_{t+1}} = mut_{rate_t} \times entropy_t^2
\]  

(6.14)

In Figure 6-30, plotting this function, we can see that the higher the entropy is, the faster the mutation rate decreases.

In order to increase the crossover rates we have used expression 6.15.

\[
cross_{rate_{t+1}} = cross_{rate_t} \times (1 - entropy_t) + entropy_t \times \frac{\ln(cross_{rate_t} + 0.2) + 2}{\ln(1.2) + 2}
\]  

(6.15)
Figure 6-30: 3-D plot for the function for decreasing the mutation rates

This function, plotted in Figure 6-31, assigns higher values for crossover rates when the entropy is high and the current crossover rate is small.

Figure 6-31: 3-D plot of the function for increasing the crossover rate

Crossover dynamic rates based on performance

As presented in Section 6.4.3, we have implemented three crossover operators, namely *Union*, *Two-Point* and *Swap*. Each of these operators can be assigned a different rate, that can vary according to its performance. The performance measures that have been defined for the crossover operators are discussed in detail in Appendix A. A crossover operator that shows a good performance is rewarded, by increasing
the number of times that it will be applied in the future. On the contrary, when a crossover operator has a poor performance it is penalised by reducing its probability of being used.

The performance of each crossover operator is evaluated by three different measures, fitpar, fitmed and domin (see Appendix A), reflecting different types of population improvement.

Let \( op \in \{\text{Union, Two Point, Swap}\} \) be a crossover operator and \( n_t(op) \) be the number of times that it is applied at generation \( t \).

\( C_t \) is a chromosome of a population \( P_t \) and \( \text{Parents}(C_t) \) is the set of parents of \( C_t \).

The fitpar\(_t(op)\) performance measure is defined as the percentage of children produced by \( op \) with a fitness larger than the fitness of one of the parents (see expression A.1).

\[
\text{fitpar}\(_t(op)\) = \frac{|\{C_t : \exists \text{parent} \in \text{Parents}(C_t) : \text{fit}(C_t) > \text{fit}(\text{parent})\}|}{n_t(op)} \tag{6.16}
\]

fitmed\(_t(op)\) measures the percentage of children produced by \( op \) with a fitness larger than the average fitness of the current population (see expression A.2).

\[
\text{fitmed}\(_t(op)\) = \frac{\left\{C_t : \text{fit}(C_t) > \frac{1}{\text{pop}\_\text{size}} \sum_{C_t \in P_t} \text{fit}(C_t)\right\}}{n_t(op)} \tag{6.17}
\]

When the ND model is used, we also apply another measure, domin\(_t(op)\), that is the percentage of children produced by \( op \) that dominate, in the Pareto sense, one of the parents (see expression A.3).

\[
\text{domin}\(_t(op)\) = \frac{|\{C_t : \exists \text{parent} \in \text{Parents}(C_t) : F\,(C) \leq F\,(\text{parent})\}|}{n_t(op)} \tag{6.18}
\]

At the end of each generation (or after a fixed number of generations), the accumulated arithmetic average, avg3\(_t\)(op), of the three performance measures is updated for each crossover operator (see expression 6.19). Consider that these values have been normalised according to expression 6.20, providing a measure of the relative quality of each operator in generation \( t \).

\[
\text{avg3}_t(op) = \frac{\sum_{i=0}^{t} \text{fitpar}_i(op) + \sum_{i=0}^{t} \text{fitmed}_i(op) + \sum_{i=0}^{t} \text{domin}_i(op)}{3 \times t} \tag{6.19}
\]

\[
\overline{\text{avg3}}_t(op) = \frac{\text{avg3}_t(op)}{\sum_{i \in \{\text{Union, Two Point, Swap}\}} \text{avg3}_i(i)} \tag{6.20}
\]

Consider that the crossover rates of the three operators have also been normalised in the \([0, 1]\) interval
satisfying \( \sum_{i \in \{\text{Union, Two Point, Swap}\}} cr_t(i) = 1 \), thus providing a measure of the relative contribution of each operator to the population.

The process for updating the crossover rate for a given operator is based on the following idea: each crossover operator should be rewarded or penalised according to its relative performance.

These normalised values, \( cr_t(op) \) and \( \bar{avg}_3(op) \), respectively, are used to compute raw values for the new crossover rates, \( Rcr_{t+1}(op) \) given by Expression 6.21.

\[
Rcr_{t+1}(op) = cr_t(op) \times (1 + \bar{avg}_3(op))^{\frac{1}{2}} \tag{6.21}
\]

The new crossover rate for each operator is computed as the normalised raw crossover rate according to Expression 6.22. The rates of the crossover operators with above average relative performance are increased while those with below average performance are decreased.

\[
cr_{t+1}(op) = \frac{Rcr_{t+1}(op)}{\sum_{i \in \{\text{Union, Two Point, Swap}\}} Rcr_{t+1}(i)} \tag{6.22}
\]

The two approaches presented in this section for the assignment of dynamic rates to the crossover and mutation operators have proved to be effective in delaying premature convergence of the algorithm.

The results concerning the performance of the crossover operators, described in Appendix A, can be directly applied by assigning a higher initial rate to the Swap crossover operator, since it has proved to consistently outperform the other two crossover operators in producing high quality solutions. It is beyond the scope of this work to comprehensively compare these two approaches. Our aim was just to provide the algorithm with effective diversification tools to avoid premature convergence and to promote the exploration of the solutions space.

### 6.7.4 Population reinitialization

All diversification strategies that were presented so far are focused on trying to avoid premature convergence. Some of these strategies can be very effective but, under certain circumstances, premature convergence can occur. When this happens, we reinitialize the entire population. This procedure replaces the existing population with a new population, randomly generated. However in order to assure that the best solution(s) found so far will not be lost, we keep the fittest individual(s) in the Agg model, and the external population in ND model.

The population diversity can be measured using the entropy function defined by expression 6.11. When the entropy is below a pre-defined (small) value, premature convergence is detected and the
population is reinitialized. However, this threshold can be hard to determine and it varies with the problem structure. Hence, we have defined a complementary measure, the structure distance \( d_{Str} \), to help detecting premature convergence. The idea of this measure is to compute the distance between the best and the worst individuals in the current population (those with the highest fitness and the lowest fitness values). A small distance means that the elements in the population are very similar and the population should be reinitialized.

The structure distance \( d_{Str} \) is the sum of two other distance functions (see expression 6.23).

\[
d_{Str}(best, worst) = d_D(best, worst) + d_S(best, worst)
\]  

(6.23)

The first component is the distance in the decision space, \( d_D \) (6.6), that has already been presented in Section 6.6.1:

\[
d_D(best, worst) = |\{i \in I : C_{best}[i] \neq C_{worst}[i]\}|
\]

The second component, the split distance, is based on the following idea: sometimes, two solutions are composed by different duties but the stretches are the same. The duties are different because they correspond to different combinations of the same stretches. Then, the distance in the decision space can be large but in fact, these two solutions have a similar structure, in the sense that they share many splitting points.

**Definition 6.14 (Split set)** For a given individual \( S_i \), the set \( \text{Split}(C_S) \) identifies the pieces-of-work where a duty ends or splits (split points)

\[
\text{Split}(C_S) = \{t : C_S[t] \neq C_S[t + 1], t = 1 \ldots m\}
\]

**Definition 6.15 (Split distance)** Given two individuals \( S_i \) and \( S_j \), the split distance compares the split points of both chromosomes and returns the number of those that do not match:

\[
d_S(i,j) = |\text{Split}(C_i) \cup \text{Split}(C_j)| - |\text{Split}(C_i) \cap \text{Split}(C_j)|
\]

In order to illustrate the application of this procedure, consider Example 6.16 of a problem with 16 pieces-of-work.

**Example 6.15** Example 6.16 In the table below, the best and the worst individuals of the population at generation \( t \) are shown. This population has the maximum structure distance is maximized (the maximum value is \( 2m - 1 \)). In fact, no duties are common to the best and the worst individuals and the split points of both solutions are all different. The split points are represented with a underscore below
the duty that covers the corresponding piece-of-work.

\[ \begin{array}{cccccccccccccccc}
\text{i} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & \text{Total} \\
C^*_\text{best} & 1 & 1 & 0 & 2 & 2 & 1 & 3 & 3 & 2 & 2 & 2 & 2 & 2 & 4 & 4 & \\
C^*_{\text{worst}} & 6 & 5 & 5 & 5 & 6 & 0 & 0 & 0 & 7 & 8 & 8 & 7 & 8 & 9 & 9 & 7 & \\
d_D & \neq & \neq & \neq & \neq & \neq & \neq & \neq & \neq & \neq & \neq & \neq & \neq & \neq & \neq & \neq & 16 \\
d_S & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & 15 \\
d_{\text{Str}} & & & & & & & & & & & & & & \neq & \neq & \neq & \neq & 31 \\
\end{array} \]

Consider now the same problem, some generations later. Now, the two solutions have some duties in common and the distance in the decision space is smaller than in the previous population. There are also some common split points, thus the split distance has a low value.

\[ \begin{array}{cccccccccccccccc}
\text{i} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & \text{Total} \\
C^{t+k}_\text{best} & 1 & 1 & 0 & 2 & 2 & 1 & 3 & 3 & 2 & 2 & 2 & 2 & 2 & 4 & 4 & \\
C^{t+k}_{\text{worst}} & 1 & 1 & 11 & 11 & 11 & 0 & 1 & 3 & 3 & 2 & 12 & 12 & 12 & 0 & 4 & 4 & \\
d_D & \neq & \neq & \neq & \neq & \neq & \# & \# & \# & \# & \# & \# & \# & \# & \# & \# & 8 \\
d_S & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & 3 \\
d_{\text{Str}} & & & & & & & & & & & & & & & & \# & 11 \\
\end{array} \]

The threshold value that identifies premature convergence can be defined by the user but we have considered 5% of the maximum value (31) as the default threshold. In this example, the two last solutions have a split distance value of 11, which is higher than the threshold (1.5) and hence the population will not be reinitialized.

### 6.7.5 Inserting random chromosomes

The last diversification strategy that we have implemented consists in allowing the insertion of random chromosomes in a population. At any generation, the user can define a percentage of the population that should be randomly generated. This strategy can be considered as a partial reinitialization of the population, that can occur at any time the user wants. It can be used as a preventive strategy, that at all generations inserts a small percentage of random individuals, but it can also be used to impose the reinitialization of the population before the threshold is automatically reached.

### 6.8 Summary

In this chapter we have presented and discussed the multiobjective genetic algorithms designed and implemented in this work. First, we have discussed the data structures of the problem and we have proposed the PWC coding. Next, we have described the components of a GA that are independent of the model. These components include the procedure for the generation of the initial population, the
crossover and mutation operators and the population replacement strategies.

Two different models have been proposed, the Aggregate model (Agg) and the Non-Dominated model (ND). While the Agg model has a single objective function that aggregates all the criteria, by assigning them different weights, the ND model is inspired on the SPEA approach which uses the concept of Pareto dominance. For this model a new fitness assignment procedure has been proposed based on the dominance sharing of a solution. The parent selection operators have been described for each model.

Finally, we have discussed the premature convergence and we have described a couple of diversification strategies specially designed to tackle this problem.

In Figure 6-32 we present a diagram with the several components described in this chapter.

Figure 6-32: Summary of the multiobjective GA components
Chapter 7

Solving Real Problems

7.1 Introduction

The multiobjective genetic algorithms developed in this work were tested on two different problem sets. The first set is composed by SPP benchmarking test instances taken from the literature [12], [89]. The second set contains real instances from Portuguese mass transit companies.

Our approach is based on a strong interaction between the planners and the algorithm. Moreover, it uses a set of objectives and constraints that are not considered in the benchmarking test instances. Even so, we have used those instances in order to show that our GA approach can effectively find the optimum in problems for which such optimal solution is known. These computational experiments are discussed in Section 7.2.

For the second class of problems we have selected some real instances intended to be representative of the diversity of problems that arise in Portuguese mass transit companies. In Section 7.3, a brief description of the GenT application is made, that aims at showing the variety of choices provided by our multiobjective approach. In Section 7.4 we characterise the problem instances selected for testing, we describe the operational rules involved and the methodology adopted for the experiments. Finally, in Sections 7.5, 7.6 and 7.7, the bus driver scheduling process of each company is described and the results obtained are presented and compared with the solutions previously implemented in each of these companies.
7.2 Computational experiments with standard problems

Our multiobjective GA approach was first tested on a set of benchmarking problems taken from the OR-Library (http://www.brunel.ac.uk/depts/ma/research/jeb/info.html) [12]. This set of problems arises from real airline crew scheduling problems, formulated as pure Set Partitioning Problem (SPP) with no side constraints or additional information. Hoffman and Padberg [89] provided the optimal solutions for these problems, as well as a set of pre-processing routines that try to reduce the original sizes of the problems.

Chu and Beasley [17] and Levine [104] have also used this set of problems for testing their GA approach to the SPP. However, since our approach is not based on a pure SPP formulation, we have adapted and simplified our model in order to fit the assumptions of those data. We have removed from our model all side constraints based on time information and on duty types, as the available data files do not provide this type of information.

We have considered two objectives: the minimisation of the total cost of the duties in the solution and the minimisation of the number of the uncovered pieces-of-work. The purpose of the second objective function is to penalise infeasible set partitioning solutions. For the Aggregate Model (Agg) model we have aggregated these two objectives into a single one and the resulting model (a relaxed SPP formulation) is as follows:

\[
\begin{align*}
\text{minimise} & \quad f(x) = \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} z_i y_i \\
\text{subject to} & \quad \sum_{j=1}^{n} a_{ij} x_j + y_i = 1, \quad i = 1, \ldots, m \\
& \quad x_j \in \{0, 1\}, \quad j = 1, \ldots, n \\
& \quad y_i \in \{0, 1\}, \quad i = 1, \ldots, m
\end{align*}
\]  

(7.1)

where \( n \) is the number of previously generated duties, \( m \) is the number of pieces-of-work and

\[
\begin{align*}
x_j & = \begin{cases} 
1 & \text{if duty } j \text{ is in the solution} \\
0 & \text{otherwise}
\end{cases}, \\
y_i & = \begin{cases} 
1 & \text{if piece-of-work } i \text{ is not covered} \\
0 & \text{otherwise}
\end{cases}, \\
a_{ij} & = \begin{cases} 
1 & \text{if duty } j \text{ covers piece-of-work } i \\
0 & \text{otherwise}
\end{cases}
\end{align*}
\]

\( c_j \) is the cost associated to duty \( j \) and \( z_i \) is the penalty associated to not covering piece-of-work \( i \). In our experiments, we have fixed \( z_i = M, i = 1, \ldots, m \), where \( M \) is a high value empirically chosen.

Model 7.1 may be seen as a generalisation of the set packing problem (with \( z_i = 0 \) and an appropriate transformation of the cost vector \( c'_j = -c_j \))[43]. Not that in this model, the \( y_i \) are not real decision variables but rather auxiliary variables.

We have tested our GA approach for the Agg model only, since the ND approach is obviously not
interesting for this set of problems. In fact, for the ND model, the two different objectives would be considered separately: \( f_1(x_j) = \sum_{j=1}^{n} c_j x_j \) and \( f_2(y_i) = \sum_{i=1}^{m} z_i y_i \). However, since only the optimal feasible solution for the SPP is searched, a trade-off between a high cost feasible solution and a lower cost unfeasible solution will never occur.

For this set of benchmark test problems we have neither used the Swap crossover operator (see Section 6.4.3) nor the Reduce_Bus_Changes mutation operator (see Section 6.4.4), since we do not have information on the bus schedules. We have also slightly adapted our code in order to exclude all constraints not provided in the data files (such as the information about the start/finish time of the piece-of-work).

Ten independent trials of the GA has been performed for each problem. For each trial, the algorithm was stopped whenever the optimal solution was found or after 2000 generations. The population size was fixed to 100 individuals, the crossover rate was set to 0.5, the mutation rate was set to 0.01 and the generation gap was set to 0.9. These parameter values were set through some previous experimentation. All tests were performed on a Pentium III 850 MHz laptop with 120 Mb RAM.

We have used the pre-processing procedures given by Hoffman and Padberg [89] to reduce the original sizes of the problems. In Table 7.1 we present the computational results obtained for the set of selected problems after the reducing routines have been applied.

The NO and NF columns present the number of optimal solutions and the number of feasible solutions found. The BstFsb column is either the best feasible solution found, or Opt, when the optimum was found. Gap is the percentage deviation from the optimal solution of the best solution found. AvFsb is the average value of all the best solutions found. AvGap is the average percentage deviation from the optimal solution. L-Best is the best solution found by Levine [104]. RnTm is the average execution time in CPU seconds.

The algorithm found feasible solutions for all the tested problems. It failed to find the optimal solution in 3 problems. In most cases, the average percentage deviation from the optimal solution was around 1%. These results show that our Genetic Algorithm is effective for the set partitioning problem, although its results are slightly worst than those presented by Chu and Beasley [17]. In fact, these authors managed to find the optimum for all these problems. Computational times are of the same order of magnitude. For 3 of the problems, marked with an X, we have not find the optimal solution. Our results are however clearly better than those obtained by Levine [104] (see L-Best column of Table 7.1), who did not find feasible solutions for 3 of the problems (marked with an X) and has not find the optimum for 9 problems.
7.3 Using the GenT application

GenT is the software application that encapsulates our multiobjective GA. It was coded in C++ and it can be used either as a stand-alone application, embedded in the GIST System (see Section 2.4.2) or in the Duties Framework (see Section 2.4.2). It reads the input data format provided by the Duties Framework.

GenT provides an interactive environment for producing driver scheduling solutions by means of a GA approach. All parameter and options can be changed and tuned by the user at any moment of the execution. The integration of GenT and the Duties Framework allows the user to visualise the generated solutions almost instantaneously.
7.3. USING THE GENT APPLICATION

The main window of GenT is presented in Figure 7-1. The text boxes (not editable) show the details of the problem and some performance measures of the algorithm. Using the edit boxes or the buttons, the user can choose the model to apply (Agg or ND) and change the parameters of the algorithm. This window also shows the main features of the best solutions found so far.

![Figure 7-1: Main Window of GenT](image)

A default value is assigned to each parameter. The user can change the values of the parameters of the chosen model in the window presented in Figure 7-2. After changes have been performed, the user must return to the main window in order to continue the execution of the algorithm.

The user can also choose the evaluation criteria to apply for a specified number of generations defined through the main window. The window in Figure 7-3 presents the different evaluation criteria described in Chapter 5. Each row refers to a different criterion which is selected if the associated check box (on the left) is checked. If the user has chosen the Agg model, he must also assign a weight to each selected criterion, by dragging the corresponding scroll bar. The weights vary between 0 and 1 and the sum of the weights for the selected criteria is forced to be 1. Moreover, if the user wants to filter solutions using a lexicographic ranking, he must assign priorities to the non selected criteria he wants to use (see Sections 6.5.1 and 6.6.1). This window also contains buttons that open new windows that allow the user...
Figure 7-2: Parameters of the algorithm

to manipulate the parameters associated to the criteria and to the duty types.

The window in Figure 7-4 presents the values defined for parameters used by the different criteria. Their default values can be changed by the user.

The window in Figure 7-5 presents the duty types for the problem under consideration. The default values of the parameters have been previously defined in GIST or in the Duties Framework, but they can now be changed by the user.

During the execution of the algorithm the user can:

- **switch among models**: for a given number of generations the user can use a model (for example, the ND model), then change to the other model (the Agg model) and go on with the execution of the algorithm.

- **change the parameters of the algorithm and of the criteria**: the user can change the parameters of the algorithm, the parameters of the criteria, or restore the default values.

- **choose another set of criteria**: the user can add or remove any number of criteria from the set of chosen criteria.

- **replicate the current run or start a new run**: by default, when the user restarts the algorithm a new random seed is generated and a completely new population is initialised. If the same random seed is used, the current run is replicated.
Figure 7.3: Criteria selection and weight assignment

- **save a solution**: the user can save any non-dominated solution provided by the algorithm. The solution is saved in a text file, using the data format of the Duties Framework. Hence, the solutions can be easily visualised in the Duties Framework and in the GIST System.

- **insert a solution into the population**: Any solution that was previously obtained by manual or automatic procedures can be inserted in a population. The solution is coded into a chromosome and it replaces a randomly chosen chromosome in the current population.

### 7.4 The methodology used in the experiments

We have selected a set of real problem instances from three Portuguese mass transit companies that are currently using the GIST system. Two of the companies are large public companies, while the third one is a medium size company managed by a local municipality. Each company has its own set of operational rules and even the general legislation set by the central government is applied differently.

In Section 7.4.1, we describe how to preprocess the input data in order to avoid generating too much duties, without discarding any interesting duty. In Section 7.4.2 we describe the general methodology adopted in the experiments.
7.4.1 Preprocessing the input data

As described in Chapter 2.3, the BDSP requires information about the pieces-of-work, the set of driver duties rules and the set of overall rules. Since all companies selected for the tests are using the GIST system, all the information needed is stored in their GIST databases.

Although GenT is fully integrated with the GIST System, we have run all our tests using the Duties Framework. The Duties Framework (see Section 2.4.2) is a stand-alone application with powerful interactive and graphical tools, analogous to those existing in the GIST System and fully integrated with it. Figure 7-6 shows the process of exporting data from GIST to GenT.

In GIST, planners have previously introduced the required information, namely the active relief opportunities, the driver duties rules and the overall rules. This information is also saved in a text file (HGenIn.hztz) with a particular data format, which is imported from the Duties Framework. In this application we can manipulate the imported data, generate the set of feasible duties and graphically visualise the solutions (manual or automatically produced).

To generate the set of feasible duties an algorithm is used that builds all the duties that satisfy the constraints and pre-defined rules. All the previous data along with this set is then added to a text file (HGenGera.htz). In order to be fully operational, the GenT application simply has to read this file. Upon a user’s request, any solution produced by GenT can be stored in a text file similar to the others (HGenOut.htz). This file can be imported by the Duties Framework as well as by the GIST System.
Although this process is fast and simple, some precautions must be taken in order to guarantee the quality of the input data. In fact, the amount and quality of these data strongly influence the performance of the algorithm and the quality of the results. GenT has no limitations concerning the maximum number of feasible (generated) duties in the input file. In our experiments we have imposed an upper limit of 15000. This value was empirically found with the contribution of the planners' suggestions, aiming at including enough feasible duties while allowing the algorithm to be efficient and to produce interesting solutions in reasonable time. The number of generated duties was also defined by taking into account the following considerations:

- The number and duration of the pieces-of-work clearly influences the number of generated duties. Each piece-of-work is defined by the time between two active relief opportunities. In most of the real problems tested, we have used the active relief opportunities already defined by the company planners. However, in a few problems we have deactivated some relief opportunities in order to reduce the number of generated duties. This was done when those relief opportunities were very close to the first or to the last relief opportunities of a vehicle.

- The set of driver duties rules strongly influences the number and the quality of the generated duties and, usually the tighter the rules, the shorter the number of feasible duties. As discussed in Chapter 2.3, some rules are imposed by the legislation, while others result from the application of internal procedures. For example, the number of duty interruptions although not constrained by the legislation, should be as small as possible. In our experiments we have not changed any constraint imposed by the legislation, although we have slightly modified some internal rules. Hence, for example, when duty interruptions are not desirable by a company, we forbid or limit the maximum number of duty interruptions, as a way to control the number of generated duties.
Figure 7-6: GenT input data process

- The global rules and, in particular, those concerning the meal periods, also affect the number of feasible duties. In fact, the constraints associated to the lunch period are perhaps those that mostly influence the quality of a driver schedule. Clearly, this aspect should be taken into account by the companies for negotiation with unions. When the meal period (usually lunch) is very tight, it becomes sometimes quite difficult to generate feasible duties. Most companies are very rigid regarding this constraint, while a few allow some flexibility. In our experiments we take advantage of the flexibility shown by some of these companies, otherwise we strictly adhere to the rules.

Finally, when comparing our solutions with those currently implemented, we have noticed that companies do often use a considerable number of infeasible duties. Most of the times, these duties involve small violations to constraints related to the break or to the stretch length. In some cases, however, in order to spare one duty, companies introduce large violations in several duties. All the solutions produced by GenT are composed by feasible duties, in the sense that they are all generated according to the set of rules defined by the company. We have not made any manual adjustment to the solutions, since such processing would be highly dependent on each company procedures.
7.4.2 Adopted methodology

Using GenT to produce BDSP solutions is very different from applying a traditional blackbox optimisation algorithm. In fact, we can interact with the algorithm at any generation, adjusting parameters, saving solutions or introducing new solutions into the population. We can alternate between the Agg model and the ND model. We can add or remove criteria, restart the run or start a new one. For example, if the algorithm is taking too long to produce an acceptable solution, we can save the best one(s) produced so far, stop the execution and resume it the next day.

Our main goal was to provide a friendly and interactive tool, fully controlled by the user, that could be able to produce good solutions in reasonable (useful) time. A single manual solution can take days or weeks to obtain and test. One of the advantages of our approach is that several satisfactory solutions can be produced in a few minutes and hence different alternative scenarios can be evaluated. The GIST System and the Duties Framework are fundamental complementary tools since they allow an almost instantaneous graphical visualisation of solutions.

Criteria

In Table 7.2 we present the criteria available in the current version of GenT, related to the soft constraints and objective functions discussed in Chapter 5. The set of criteria initially selected depends on each particular company, which can save its particular settings.

<table>
<thead>
<tr>
<th>N.</th>
<th>Criterion Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Number of leftovers ($eval_1$)</td>
</tr>
<tr>
<td>2</td>
<td>Number of duties ($eval_2$)</td>
</tr>
<tr>
<td>3</td>
<td>Total duration of leftovers ($eval_3$)</td>
</tr>
<tr>
<td>4</td>
<td>Gap from an average duration of duties ($eval_4$)</td>
</tr>
<tr>
<td>5</td>
<td>Duties cost ($eval_5$)</td>
</tr>
<tr>
<td>6</td>
<td>Number of vehicle changes ($eval_6$)</td>
</tr>
<tr>
<td>7</td>
<td>Gap from an average duration for each duty type ($eval_7$)</td>
</tr>
<tr>
<td>8</td>
<td>Penalisation of duty types ($eval_8$)</td>
</tr>
<tr>
<td>9</td>
<td>Penalisation of minimum, maximum or average duration of leftovers ($eval_9$)</td>
</tr>
</tbody>
</table>

Table 7.2: Available Criteria

Parameters

Table 7.3 presents the range and the default values for the algorithm parameters. Although each company can define their own set of parameter values, they are advised to start with this set of values, that have been tuned through our experience with several problems from different companies. All parameters
are applicable to both Agg and the ND models, except parameter 3 that only refers to the ND model. Parameters 1 and 2 are general GA parameters (presented in Sections 6.4.2 and 6.4.5). Parameters 4 to 8 are related with the diversifications strategies discussed in Section 6.7. Parameters 9 to 11 are crossover operator rates (Section 6.4.3) and parameters 12 to 16 are mutation and repairing operator rates (Section 6.4.4).

<table>
<thead>
<tr>
<th>N.</th>
<th>Parameter name</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Population size</td>
<td>10</td>
<td>500</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>Generation gap</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>Rate of elements in the elite set</td>
<td>0</td>
<td>1</td>
<td>0.05</td>
</tr>
<tr>
<td>4</td>
<td>Rate of random individuals per generation</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Minimum distance</td>
<td>0</td>
<td>1</td>
<td>0.05</td>
</tr>
<tr>
<td>6</td>
<td>Temperature decrement rate</td>
<td>0</td>
<td>1</td>
<td>0.98</td>
</tr>
<tr>
<td>7</td>
<td>Generations with constant temperature</td>
<td>0</td>
<td>-</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>Minimum entropy</td>
<td>0</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>9</td>
<td>Union crossover rate</td>
<td>0</td>
<td>1</td>
<td>0.4</td>
</tr>
<tr>
<td>10</td>
<td>Two-Point crossover rate</td>
<td>0</td>
<td>1</td>
<td>0.4</td>
</tr>
<tr>
<td>11</td>
<td>Swap crossover rate</td>
<td>0</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>12</td>
<td>Improve mutation rate</td>
<td>0</td>
<td>1</td>
<td>0.01</td>
</tr>
<tr>
<td>13</td>
<td>Destroy mutation rate</td>
<td>0</td>
<td>1</td>
<td>0.01</td>
</tr>
<tr>
<td>14</td>
<td>Destroy rate</td>
<td>0</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>15</td>
<td>Reduce-Empty mutation rate</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>16</td>
<td>Reduce-Bus-Changes mutation rate</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 7.3: Default parameter values

Table 7.4 lists several parameter options together with their default values. Parameter 17 that allows the user to choose the Duties Selection procedure, has already been discussed in section 6.4.1. By default we have chosen to apply dynamic crossover rates (parameter 18) even if this option considerably slows the execution time (see Section 6.7). Existence of duplicates in the population (parameter 19) as discussed in 6.7 is allowed by default. The default method to select solutions for the elite set (parameter 20) was clustering, a method defined in Section 6.6.1. Finally, the default distance function used in the clustering procedure is the distance in the objective space (parameter 21).

General guidelines

Comprehensive experimentation has shown that the following guidelines should be followed when using GenT: understanding the BDSP of each company, choosing the appropriate criteria, controlling duty types and improving diversification.
Table 7.4: Default parameter options

- Understanding the BDSP of each company

Although the major features of the process are common to all companies, each has its own way of dealing with them. The trade-off between the number of duties and the number of leftovers, the average duration of the duties or the percentage of each duty type are handled differently by each company. In fact not all possible rules and internal procedures have been incorporated in the GIST System, and hence some time and effort must be spent in analyzing the current solutions and in understanding the way planners work, before tuning the system. Besides, the network topology and the dimension of the problems also differ from company to company.

- Choosing the appropriate criteria

The set of selected criteria can change during one single run, which will hopefully lead the algorithm to explore different areas of the search space. In our methodology some criteria are permanently selected, while others are added or removed depending on the solutions obtained. In the Agg model, we have to assign different weights to each criterion and if there are too many criteria involved, it becomes difficult to control the trade-off between criteria for the various solutions. In the ND model, the selection of too many criteria is likely to lead to a high number of non-dominated solutions and the algorithm may become too slow. Hence we have decided to start with a small number of criteria which is increased according to the quality and features of the solutions obtained. For example, we start by selecting the minimisation of the number of duties, and the penalisation to the duty types may be selected later, only if the obtained solutions do not contain the desired percentage of duty types.

- Controlling the duty types

As referred in Chapter 6 (Section 6.4), usually the set of generated duties is not uniformly distributed along the different duty types. Some duty types, related to more flexible rules, are naturally represented in a higher percentage. Hence, a purely random duty selection procedure might introduce a biased representation of the search space. We have therefore proposed two additional duty selection procedures in order to control this bias: the Maximum Percentage Procedure and the Priority Level Procedure.
When a duty type is much more represented in the set of generated duties than another, we should use one of these two procedures.

Consider, for example, a company where the set of generated duties is composed by 5000 duties of type A, 1000 duties of type B and 50 of type C. If this company always prefers duty type A to duty types B and C, in a way that is well reflected by these figures, it is not necessary to use a duty selection procedure, since duties of type A will be randomly chosen more frequently. However, if the maximum percentage of duty type A is limited to a certain value, we should use the Maximum Percentage Procedures. If duty types B and C are preferred to duty type A, we should use the Priority Level Procedure. The percentage of duty types can also be controlled through the criterion that tries to minimise the penalisation assigned to each duty type. However, duty selection procedures impose a fundamental search constraint that can improve the performance and effectiveness of the search.

- Improving diversification

In Chapter 6 we have presented several diversification strategies (Section 6.7). All of these strategies can be tuned by the user, although for testing purposes we have generally adopted the default values referred in Section 6.7. However, if the solutions do not improve for a large number of generations, we can force the algorithm to reinitialise the population, by increasing the minimum distance threshold. When the distance between the best and the worst chromosomes in the current population attains this value, the population is reinitialised. On the other hand, if the number of reinitializations is too high, we can decrease the threshold. The effectiveness of the algorithm in automatically detecting premature convergence depends on each particular problem but this is an issue that clearly deserves further research.

Another parameter that can be dynamically controlled by the user is the size of the elite set. When the number of non-dominated solutions found in each generation is higher than the maximum size of the elite set, the clustering procedure chooses those solutions that will be kept in the elite set. In some cases, the number of clusters (maximum number of elements in the elite set) defined by default (5% of the population size) may not be enough to provide a good representation of the non-dominated solutions found so far, and some interesting solutions may be excluded. In this situation, we can increase the number of solutions in the elite set. On the other hand, when the solutions in the elite set are too much similar, we should decrease the number of solutions in the elite set, thus forcing the clustering procedure to join in the same cluster similar solutions.

An experienced user is likely to strongly interact with the algorithm during one run. He can change the selected criteria and increase or decrease the size of the elite set trying to obtain solutions with the desired features. This process obviously involves a certain degree of subjectivity making it difficult to report and interpret the results of experiments. Nevertheless, we now present two charts reporting to typical runs of each model (Agg and ND), in which there was no user interaction.
Figure 7-7 describes a run of the Agg model for a real problem with a passive user. The criteria selected were the number of duties (weight: 5), the number of leftovers (weight: 5), the duration of the leftovers (weight: 1) and the duties cost (weight: 1). The figure shows the evolution of the best solution of the population for each criterion during 140 generations, as well as the value of the aggregate objective function (Total eval). The duration of the leftovers is presented in a secondary time axis. We can observe that the best solution is not always improving in all criteria, since sometimes when the number of duties increases, the number of leftovers or their duration decreases. Only the duties cost is always (decreasing) during the run, but this criterion is directly affected by the number of duties and the number of leftovers. Naturally, the aggregate function is constantly improving with a behaviour very similar to that of the duties cost criterion.

![A typical run of the Agg model without user interaction](image)

Figure 7-7: A typical run of the Agg model without user interaction

Figure 7-8 shows a run for the same problem using the ND model and no user interaction. We have chosen the same criteria and the algorithm has run for 200 generations. In this case, instead of the best solution in the population, we have plotted the solutions in the elite set, whose maximum size is 10. We can see that as the algorithm evolves, the solutions in the elite set become more similar. In fact, in the last generations, the number of duties varies from 58 to 60 and the number of leftovers is very small.
The duration of the leftovers is the criterion that shows a higher variation along the various generations.

![A typical run of the ND model without user interaction](image)

Figure 7-8: A typical run of the ND model without user interaction

Finally, Table 7.5 shows a plot of the number and duration of the leftovers versus the number of duties for the solutions in the elite set at generation 200. This set is composed by three solutions with 58 duties and two leftovers, two solutions with 59 duties and one leftover, and a single solution with 60 duties and no leftovers. We can easily see that, as the number of duties decreases, the number and duration of the leftovers increases. These six solutions are presented to the user, who is asked to judge and choose. While some companies would choose the solution without leftovers, others would choose one of the solutions with less duties, depending on the duration of the leftovers.

<table>
<thead>
<tr>
<th>N. duties</th>
<th>N. leftovers</th>
<th>Dur. leftovers</th>
<th>Duties Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>0</td>
<td>0:00</td>
<td>64783</td>
</tr>
<tr>
<td>59</td>
<td>1</td>
<td>1:17</td>
<td>67263</td>
</tr>
<tr>
<td>59</td>
<td>1</td>
<td>2:08</td>
<td>66477</td>
</tr>
<tr>
<td>58</td>
<td>2</td>
<td>2:52</td>
<td>68780</td>
</tr>
<tr>
<td>58</td>
<td>2</td>
<td>3:26</td>
<td>68325</td>
</tr>
<tr>
<td>58</td>
<td>2</td>
<td>4:01</td>
<td>67928</td>
</tr>
</tbody>
</table>

Table 7.5: Final solutions in the elite set
The next three sections describe the bus driver scheduling process in three Portuguese transport companies and how GenT was used to efficiently produce solutions of good quality. The time needed to produce each solution depends on the size of the problem, but we have imposed two hours as the maximum time for GenT to produce one solution (in the Agg model) or a set of solutions (in the ND model). Since GenT is used interactively, we consider as execution time the real elapsed planning time, including reading the input data, choosing criteria, running the model, analysing and saving intermediate solutions, changing criteria or adjusting parameters.

### 7.5 Bus driver scheduling at Carris

Carris is a large public transport company operating in Lisbon with 882 buses and 1778 drivers. They have been using the GIST system since 1995. Moreover, Carris is now using a preliminary version of GenT in their regular planning process. It should be referred that some of the features currently available in the application have been suggested by their planning team. Unfortunately, the old manual solutions are not available anymore, thus preventing us from performing interesting comparisons. Currently, after one or more executions of the algorithm, the planners choose one of the solutions proposed by the system, eventually making some manual adjustments on it.

As noted in Chapter 2, although Carris is a large company, their BDSP instances are small, since they divide the operational planning process into several small problems that are handled independently. In fact each BDSP instance corresponds to a single route.

Carris allows two duty types: straight duties and split duties, whose features are summarised in Table 7.6. The lunch period lies between 10h00m and 15h00m and the dinner period lies between 18h00m and 22h30m. Only split duties must satisfy these constraints and one of the meals must lie inside the break. However, planners allow a small tolerance (1 to 3 minutes) to the meal duration (1 hour).

In a schedule, the average duration of the duties should be as close as possible to 8 hours. This is difficult to achieve, since straight duties are preferred to split duties and their maximum duration is 7h30m. Hence planners try to obtain straight duties with an average duration close to 7 hours, since an internal procedure imposes that an additional period (dummy time) of 30 minutes is added to all straight duties and split duties with an average duration of 8 hours. Straight duties are allowed to have a "small interruption" from 5 to 50 minutes, although this is a highly undesirable situation, since that interruption is paid time.

Most driver schedules in Carris include some uncovered work (leftovers). Leftovers are only allowed if they satisfy certain properties. A considerable part of the driver scheduling process in the company
is therefore devoted to controlling the generation of leftovers. Usually, leftovers coming from different schedules are handled together and used to build stretches of new duties. This process is a consequence of handling each line independently, which results in small problems for which it is more difficult to obtain solutions with no leftovers. It is performed in a subsequent phase, when all driver schedules are complete. Since the leftovers are used to build stretches of duties, their duration should not be smaller than 2 hours and their average duration should be close to 3h30m. When it is not possible to build new duties with the leftovers, they are assigned as extra time to the existing duties or, in the worst case, they are simply not performed.

Since leftovers cannot be assigned to a driver in a regular basis (in the rostering phase), certain precautions must be taken concerning some of their features. In particular, leftovers should not correspond to the first or to the last trips of a vehicle and they should not be in consecutive running boards in the same time period.

### 7.5.1 Computational results

Carris is already regularly using GenT as part of GIST. However, we have decided to make some more controlled experiments on a small group of lines to show how to combine different criteria in order to obtain alternative solutions. Since leftovers are used to build stretches of new duties, their duration should not be too short. The trade-off between the number of leftovers and the number of duties was analysed for each problem instance, considering the percentages of duty types involved (straight duties are preferred) and the average duration of the duties (as close as possible to 8 hours). The set of

<table>
<thead>
<tr>
<th>Rule description</th>
<th>Straight duties</th>
<th>Split duties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Number of stretches</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>Stretch length</td>
<td>6:00</td>
<td>7:30</td>
</tr>
<tr>
<td></td>
<td>2:00</td>
<td>5:00</td>
</tr>
<tr>
<td>Break length</td>
<td>1:00</td>
<td>1:00</td>
</tr>
<tr>
<td>Meal duration</td>
<td>6:00</td>
<td>7:30</td>
</tr>
<tr>
<td>Normal duty length</td>
<td>7:30</td>
<td>7:30</td>
</tr>
<tr>
<td>Driving time</td>
<td>8:00</td>
<td>9:00</td>
</tr>
<tr>
<td>Total working time</td>
<td>7:30</td>
<td>9:00</td>
</tr>
<tr>
<td>Total normal spreadover</td>
<td>6:00</td>
<td>7:30</td>
</tr>
<tr>
<td>Number of duty interruptions</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Length of duty interruptions</td>
<td>0:05</td>
<td>0:50</td>
</tr>
<tr>
<td>Number of vehicle changes</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Non-start duties periods(hh:mm)</td>
<td>8:20</td>
<td>12:00</td>
</tr>
<tr>
<td>Mealbreak validation</td>
<td>No</td>
<td>In the break</td>
</tr>
</tbody>
</table>

Table 7.6: Duty types for Carris
generated duties satisfies all the standard rules presented in Table 7.6.

For our experiments, we have arbitrarily chosen three problems instances, whose main characteristics are presented in Table 7.7. The table identifies the instance, shows the routes involved, the total number of vehicles, the total duration of the schedule (in hours), the number of nodes that are used as relief points, the number of active pieces-of-work and the average duration of these pieces-of-work (in hours).

<table>
<thead>
<tr>
<th>Problem</th>
<th>Route</th>
<th>Number of buses</th>
<th>Total duration</th>
<th>Number of relief points</th>
<th>Number of pieces-of-work</th>
<th>Average duration of pieces-of-work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carris-A</td>
<td>18</td>
<td>10</td>
<td>136:00</td>
<td>3</td>
<td>148</td>
<td>0:55</td>
</tr>
<tr>
<td>Carris-B</td>
<td>20</td>
<td>9</td>
<td>131:27</td>
<td>3</td>
<td>208</td>
<td>0:37</td>
</tr>
<tr>
<td>Carris-C</td>
<td>22</td>
<td>7</td>
<td>105:10</td>
<td>2</td>
<td>91</td>
<td>1:09</td>
</tr>
</tbody>
</table>

Table 7.7: Carris problem set

Table 7.8 presents a summary of four alternative solutions obtained for problem Carris-A, together with the current solution (this solution is denoted here by "Carris"). The first column identifies the solution, the second and third columns give the number of duties and leftovers in the solution, the fourth column gives the total duration of the leftovers. The fifth and sixth columns show the average duration of the leftovers and duties, respectively. For the average duration of the duties we added 30 minutes to the straight duties, thus reproducing the internal procedure adopted by Carris. The last columns show the number, percentage and average duration of duties for each duty type.

Solutions A.AG.1 and A.AG.2 were obtained using the Agg model in two different runs, while solutions A.ND.1 and A.ND.2 were produced by ND model in the same run. Solutions A.AG.1 and A.ND.1 are very similar to the current solution, with 2 leftovers and similar percentages of straight and split duties. We were trying to minimise the number of leftovers (criterion 1), the gap from the average duration of duties (criterion 4), the total cost of the duties (criterion 5) and the penalisation assigned to the duty types (criterion 8). The average duration of duties was set to 8 hours and a high penalisation was assigned to split duties.

Solutions A.AG.2 and A.ND.2 were obtained with a different set of criteria. In this case, we have also tried to minimise the number of leftovers and an average duration of the duties close to 8 hours, but we have replaced criteria 5 and 8 by the minimisation of the number of duties (criterion 2) and the minimisation of the total duration of the leftovers (criterion 3). The algorithm has reduced the number of leftovers to 0 but the number of split duties has increased. In fact, the solution produced by the Agg model has one more duty than the others. The right trade-off between the number of duties and leftovers and the percentages of duty types is not at all clear, and a detailed analysis is therefore left to the planners.
Table 7.8: Solutions for problem Carris-A

In Table 7.9 we present the solutions obtained for problem Carris-B. The solutions produced by the Agg model have managed to reduce the number and duration of the leftovers, but the number of duties has increased. However, the number of split duties has not changed. Solution B.ND.1 is very similar to the current solution in Carris, while in solution B.ND.2 the algorithm has managed to replace two leftovers by a feasible duty. The leftovers are not too short and the number of straight duties is larger than the number of split duties, as a result of having selected criterion 8. For getting solution B.ND.3 we have not penalised the duty types and we have tried to minimise the total duration of the leftovers and. In fact, this solution contains two more (split) duties than Carris current solution, but it has no leftovers.

Table 7.9: Solutions for problem Carris-B

The solutions obtained for this problem are also represented in the chart of Figure 7-9 which shows the trade-off between the number of duties and leftovers and the percentage of duty types.

Finally, Table 7.10 presents the solutions obtained for problem Carris-C. Although solution C.AG.1, obtained by the Agg model, is not very interesting, when compared with the solutions produced by the ND model, we have decided to present it for comparison purposes. Solution C.ND.1 has less 3 leftovers, by adding one single duty while maintaining the proportion of straight and split duties. Solutions C.ND.2
7.5. BUS DRIVER SCHEDULING AT CARRIS

Figure 7-9: Solutions obtained for problem Carris-B

and C.ND.3 are very similar to each other but, while solution C.ND.3 has one leftover less than solution C.ND.2, the average duration of the leftovers (4h53m) may be considered too high. Once again, the final analysis of the trade-off between these two solutions is left to the planners.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Carris</td>
<td>13</td>
<td>4</td>
<td>10:50</td>
<td>2:42</td>
<td>7:38</td>
<td>8</td>
<td>61.5</td>
<td>6:58</td>
<td>5</td>
<td>38.5</td>
<td>7:55</td>
</tr>
<tr>
<td>C.AG.1</td>
<td>14</td>
<td>2</td>
<td>4:12</td>
<td>2:06</td>
<td>7:27</td>
<td>7</td>
<td>50.0</td>
<td>6:45</td>
<td>7</td>
<td>50.0</td>
<td>7:40</td>
</tr>
<tr>
<td>C.ND.1</td>
<td>14</td>
<td>1</td>
<td>4:40</td>
<td>4:40</td>
<td>7:24</td>
<td>8</td>
<td>57.1</td>
<td>6:48</td>
<td>6</td>
<td>42.9</td>
<td>7:34</td>
</tr>
<tr>
<td>C.ND.2</td>
<td>13</td>
<td>3</td>
<td>9:50</td>
<td>3:16</td>
<td>7:36</td>
<td>7</td>
<td>53.8</td>
<td>6:51</td>
<td>6</td>
<td>46.2</td>
<td>7:54</td>
</tr>
<tr>
<td>C.ND.3</td>
<td>13</td>
<td>2</td>
<td>9:46</td>
<td>4:53</td>
<td>7:34</td>
<td>7</td>
<td>53.8</td>
<td>6:50</td>
<td>6</td>
<td>46.2</td>
<td>7:51</td>
</tr>
</tbody>
</table>

Table 7.10: Solutions for problem Carris-C

7.5.2 Concluding remarks

We have used the algorithm with its default parameters, except for the following situations:

1. When the number of non-dominated solutions in a generation becomes too large, the default clustering procedure may exclude some interesting solutions. Then, for some generations, we increase the number of elements in the elite set in order to obtain a more representative sample of the solutions space. When the algorithm stabilises in a small number of non-dominated solutions we decrease again the number of elements in the elite set.
2. When the solutions do not improve for a large number of generations, the algorithm has probably failed to detect premature convergence. Then we have adjusted the parameters that improve diversification.

Straight duties can have a small duty interruption, although planners try to reduce their number as much as possible. Typically, Carris driver schedules contain one or two straight duties with interruptions. In order to control the number of duty interruptions we have tried the following different strategies:

1. In the generation phase, we may assign a high penalisation to each duty interruption thus increasing the cost of these duties. When using this strategy, we must additionally consider the criterion that minimises the duties costs (criterion 5). However, when the cost function has several components, it is difficult to choose the appropriate penalty values in order to obtain the desired result.

2. Since a duty interruption usually involves a vehicle change, as an alternative, we can consider the criterion that minimises the number of vehicle changes (criterion 6). The problem with this strategy is that it affects all duties with vehicle changes and not only the duties with interruptions.

3. In the generation phase, we can define two different types of straight duties: one that allows interruptions and another that does not allow interruptions. A high penalisation must be assigned to the straight duties with interruptions and the criterion that penalises duty types (criterion 8) must be selected. We have used this strategy with success, since no solution obtained in this way for Carris had straight duties with interruptions.

At Carris there are two important constraints concerning leftovers: they must neither include nor the first or the last trip of a vehicle, and they must not occur in consecutive vehicles in the same period of the day. The first constraint is taken into account by the algorithm, since the planner may assign a penalty factor to the leftovers in the beginning or in the end of a vehicle schedule. The second constraint, however, is not yet considered by the algorithm. Hence, all solutions with leftovers must be analysed in order to check if there are violations of this constraint. Here we have only presented solutions that satisfy this constraint.

Analysing the several solutions generated for each problem, we can observe that, as expected, reducing the number of leftovers usually increases the number of duties. Moreover, for each particular instance, the larger the number of duties, the lower the average duration of duties.

In all solutions, the average duration of the leftovers was never smaller than 2 hours neither larger than 5 hours. This seems to be reasonable to the planners.

We have applied both the Agg and the ND models, but we have not switched models in the same run. Each solution provided by the Agg model was obtained in a different run, while the ND model
solutions were usually obtained in the same run. Each solution was obtained in less than 15 minutes and some of them in only a few minutes. In particular, the Agg model has proved to be considerably faster than the ND model. We have not made manual adjustments to the solutions, and all duties satisfy the whole set of constraints and operational rules, although some of the current solutions in Carris contain several unfeasible duties that were produced after small manual adjustments. Carris instances are very small problems, but several hours would be needed to produce them manually.

The main difficulties we faced were related to the proportion of straight and split duties and with the duration of the leftovers. For all problems we found several alternative solutions, thus confirming the multiobjective nature of the problem.

Finally, the difficulty in finding solutions with no leftovers is related to the small size of the problems and to the specific rules imposed by Carris. They believe that, in larger schedules, it would be considerably more difficult to obtain a duties average duration close to 8 hours.

7.6 Bus driver scheduling at STCP

STCP is a large public transport company operating in the metropolitan region of Porto with 549 buses and 1214 drivers. They have been using the GIST system since 1995. Currently, the company strongly uses GIST graphical capabilities to support manual bus driver scheduling. At STCP each problem is the result of interlining three or more lines, creating medium size instances. Planners state that producing a reasonable solution from scratch can take one or two days.

The company allows four duty types: straight duties, night duties, split duties and long break duties, whose rules are summarised in Table 7.11. Night duties are very similar to straight duties, occuring in the night period (from 20h00m to 8h00m). Straight duties are always preferred to split duties and long break duties are highly undesirable. This duty type is usually used to cover peak periods and its long spreadover make it very unpopular for the drivers.

Leftovers are compiled in a report that is circulated among the drivers and proposed as extra work. However, STCP planners avoid as much as possible to assign extra work to drivers for two main reasons: the costs involved and the extra required negotiation with drivers and unions. Hence, one of the main goals of the company is to completely avoid leftovers from driver schedules although, in order to achieve this goal, planners often violate several other rules.

Only the lunch period is defined, lying between 10h00m and 15h00m and the minimum time for a meal is one hour. For straight and split duties, lunch may be scheduled before or after the work. Split duties can also consider the lunch period as part of the break. Night and long break duties do not have any constraints concerning mealbreaks. Although several other rules are often broken by the planners
in their manual scheduling, mealbreak constraints are always satisfied.

<table>
<thead>
<tr>
<th>Rule description</th>
<th>Straight duties</th>
<th>Night duties</th>
<th>Split duties</th>
<th>Long break duties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>Stretch length</td>
<td>5:00</td>
<td>7:00</td>
<td>5:30</td>
<td>7:00</td>
</tr>
<tr>
<td>Break length</td>
<td>1:00</td>
<td>1:00</td>
<td>1:00</td>
<td>2:00</td>
</tr>
<tr>
<td>Meal duration</td>
<td>1:00</td>
<td>0:00</td>
<td>1:00</td>
<td>0:00</td>
</tr>
<tr>
<td>Normal duty length</td>
<td>5:00</td>
<td>7:00</td>
<td>5:30</td>
<td>7:00</td>
</tr>
<tr>
<td>Driving time</td>
<td>7:00</td>
<td>7:00</td>
<td>7:00</td>
<td>7:00</td>
</tr>
<tr>
<td>Total working time</td>
<td>7:00</td>
<td>7:00</td>
<td>7:00</td>
<td>7:00</td>
</tr>
<tr>
<td>Total normal spreadover</td>
<td>5:00</td>
<td>7:00</td>
<td>5:00</td>
<td>7:00</td>
</tr>
<tr>
<td>Number of duty interruptions</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Number of vehicle changes</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Non-finish duties periods</td>
<td>23:00</td>
<td>4:15</td>
<td>0:00</td>
<td>23:00</td>
</tr>
<tr>
<td>Non-start duties periods</td>
<td>4:00</td>
<td>16:00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mealbreak validation</td>
<td>Lunch only</td>
<td>No</td>
<td>Lunch only</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 7.11: Duty types for STCP

At STCP, the average duration of duties is computed by duty type. Straight and split duties together should average 6h40m, while long break and night duties should average 6h30m. This rule is not very rigid, but it reflects an internal procedure related to the rostering process. In fact, straight and split duties are handled together in the same roster, while long break and night duties belong to separate rosters. At STCP, drivers have a single day break by week and each rotation group includes several weeks. This explains the shorter average duration of the duties when compared with Carris.

In conclusion, STCP tries to simultaneously minimise the number of duties, to eliminate leftovers (or, when it is not possible, to minimise their duration) and to satisfy the pre-defined average durations. To deal with these conflicting objectives, a multiobjective approach seems to be the most appropriate.

### 7.6.1 Computational results

We have arbitrarily chosen three problem instances from STCP, whose main features are presented in Table 7.12. We have always used the original set of pieces-of-work and the set of rules defined by the planners in the GIST system. Since at STCP the minimisation of the number of leftovers is a very important objective, we have always tried to obtain solutions with no leftovers. However, when we obtained a solution with one or two leftovers but with a number of duties considerably smaller than that of the current solution, we have chosen to include it in the set of alternative solutions. In fact, it
may be interesting for the company to analyse if the reduction in the number of drivers compensates the existence of a few extra hours.

Concerning the duty types, we have assigned the highest priority and the lowest penalisation to straight duties, and the lowest priority and the highest penalisation to long break duties. The purpose was to obtain solutions with a maximum number of straight duties and a minimum number of long break duties.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Routes</th>
<th>Number of buses</th>
<th>Total duration</th>
<th>Number of relief points</th>
<th>Number of pieces-of-work</th>
<th>Average duration of pieces-of-work</th>
</tr>
</thead>
<tbody>
<tr>
<td>STCP-A</td>
<td>7-92-95</td>
<td>24</td>
<td>384:29</td>
<td>5</td>
<td>218</td>
<td>1:45</td>
</tr>
<tr>
<td>STCP-C</td>
<td>31-36-55-89</td>
<td>20</td>
<td>238:23</td>
<td>3</td>
<td>126</td>
<td>1:53</td>
</tr>
</tbody>
</table>

Table 7.12: STCP problem set

The actual time needed to interactively produce each solution with the Agg model was at most 15 minutes. For the ND model, the maximum time needed to obtain a set of non-dominated solutions was around 30 minutes.

Table 7.13 presents a summary of several alternative solutions obtained for problem STCP-A as well as the company's current solution. The second and third columns present the number of duties and leftovers in the solution, and the fourth column gives the total duration of the leftovers. In the fifth column we present the global average duration of straight and split duties. The last columns show the number, percentage and average duration of duties for each duty type.

Solution A.Ag.1 has the same number of duties as the current solution and no leftovers. It does not contain any long break duty, although it contains less straight duties than the current solution. However, the average durations of straight and split duties is closer to the target value. This solution was obtained with the Agg model and we have mainly used the criteria 2, 3 and 8, with a higher weight assigned to criterion 3, thus minimising the duration of leftovers.

Solution A.ND.1 is very similar to the current solution although it does not contain any long break duty. Solution A.ND.2 has one duty less than the current solution and no leftovers, but the number of split duties is higher. These two solutions have been obtained in the same run of the ND model, with the same set of criteria that was used in the Agg model. We have assigned a very high penalisation to long break duties in order to exclude these duties from the solutions.

Solutions A.ND.3 and A.ND.4 were also obtained in a single run of the ND model but, in this case we have selected criteria 2, 3 and 4, to obtain solutions with the minimum number of duties and leftovers and an average duration close to the desired. Criterion 8, has been added occasionally, trying to reduce
the number of long break duties. With solution A.ND.3 we have managed to reduce the number of duties by two but, on the other hand, three long break duties have been added. Finally, solution A.ND.4 has three duties less, but it contains one leftover (with 2 hours), which may make it non feasible in practice. Besides, this solution contains five long break duties (two more than the current solution).

In all solutions the average duration of straight and split duties is closer to the desired value (6:40m) than the current solution, although the number of straight duties is always smaller.

<table>
<thead>
<tr>
<th>Solution</th>
<th>N. duties</th>
<th>N. leftov</th>
<th>Dur leftov</th>
<th>Avdr</th>
<th>StrSpl</th>
<th>Straight duties</th>
<th>Split duties</th>
<th>LB duties</th>
<th>Night duties</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.STCP</td>
<td>60</td>
<td>0</td>
<td>0:00</td>
<td>6:28</td>
<td>40</td>
<td>66.7</td>
<td>6:14</td>
<td>10</td>
<td>16.7</td>
</tr>
<tr>
<td>A.AG.1</td>
<td>60</td>
<td>0</td>
<td>0:00</td>
<td>6:40</td>
<td>30</td>
<td>50.0</td>
<td>6:19</td>
<td>22</td>
<td>36.7</td>
</tr>
<tr>
<td>A.ND.1</td>
<td>60</td>
<td>0</td>
<td>0:00</td>
<td>6:31</td>
<td>37</td>
<td>61.7</td>
<td>6:17</td>
<td>15</td>
<td>25.0</td>
</tr>
<tr>
<td>A.ND.2</td>
<td>59</td>
<td>0</td>
<td>0:00</td>
<td>6:40</td>
<td>32</td>
<td>54.2</td>
<td>6:22</td>
<td>19</td>
<td>32.2</td>
</tr>
<tr>
<td>A.ND.3</td>
<td>58</td>
<td>0</td>
<td>0:00</td>
<td>6:41</td>
<td>33</td>
<td>56.9</td>
<td>6:24</td>
<td>15</td>
<td>25.9</td>
</tr>
<tr>
<td>A.ND.4</td>
<td>57</td>
<td>1</td>
<td>2:00</td>
<td>6:48</td>
<td>28</td>
<td>49.1</td>
<td>6:26</td>
<td>17</td>
<td>29.8</td>
</tr>
</tbody>
</table>

Table 7.13: Solutions for STACP-A problem

The trade-off between the solutions provided by the ND model can be clearly understood through Figure 7-10 that shows, for each solution, the number of duties of each type. The solutions are sorted by decreasing order of number of duties. The trend lines show that as the number of duties decreases, we have more split and long break duties and less straight duties. The reason is that split and long break duties can have larger durations than straight duties and hence less duties are needed. Although the average duration for each duty type is respected, the best compromise is to be decided by the planners.

Table 7.14 presents the solutions proposed for problem STCP-B. Although the solution provided by the Agg model (B.AG.1) contains one more duty than the current solution, we have included it for comparison purposes. In fact, it was difficult to choose the right criterion weights to reduce the number of leftovers without adding duties.

Solutions B.ND.1, B.ND.2 and B.ND.3 were obtained through the ND model and we have mainly used criterion 2 to minimise the number of duties, criterion 3 to minimise the duration of the leftovers and criterion 8 with a high penalisation for long break duties. Solution B.ND.1 is very similar to the current STCP solution, while solution B.ND.2 has one duty less and more split duties. The average duration of duties is however closer to the target value. In solution B.ND.3 we have managed to reduce the number of duties by two but one leftover has been added.

In solutions B.ND.4 and B.ND.5, we have not used criterion 8 and, as a consequence, a few long break duties has been added. Our purpose was to reduce the number of duties without imposing any
restrictions to the duty types. The best solution found has 50 duties (three less than the current solution) and two small leftovers.

Figure 7-10: Number of duty types in ND solutions for problem STCP-A

<table>
<thead>
<tr>
<th>Solution</th>
<th>N. duties</th>
<th>N. leftov.</th>
<th>Dur leftov</th>
<th>Avdr</th>
<th>Straight duties</th>
<th>Split duties</th>
<th>Long break duties</th>
<th>Night duties</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.STCP</td>
<td>53</td>
<td>0</td>
<td>0:00</td>
<td>6:25</td>
<td>35</td>
<td>66,0</td>
<td>6:13</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>18,9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7:07</td>
<td>3,8</td>
<td>6:52</td>
<td>6.25</td>
</tr>
<tr>
<td>B.AG.1</td>
<td>54</td>
<td>0</td>
<td>0:00</td>
<td>6:23</td>
<td>30</td>
<td>55,6</td>
<td>6:11</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>29,6</td>
<td>6:46</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6:54</td>
<td>3,7</td>
<td>6:52</td>
<td>6.20</td>
</tr>
<tr>
<td>B.ND.1</td>
<td>53</td>
<td>0</td>
<td>0:00</td>
<td>6:23</td>
<td>35</td>
<td>67,9</td>
<td>6:12</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>18,9</td>
<td>7:05</td>
<td>2.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7:05</td>
<td>3,8</td>
<td>6:52</td>
<td>4.9</td>
</tr>
<tr>
<td>B.ND.2</td>
<td>52</td>
<td>0</td>
<td>0:00</td>
<td>6:31</td>
<td>30</td>
<td>57,7</td>
<td>6:15</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>28,8</td>
<td>7:05</td>
<td>3.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7:05</td>
<td>3,8</td>
<td>6:52</td>
<td>5.31</td>
</tr>
<tr>
<td>B.ND.3</td>
<td>51</td>
<td>2</td>
<td>2:43</td>
<td>6:35</td>
<td>25</td>
<td>49,0</td>
<td>6:10</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>37,2</td>
<td>7:10</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7:10</td>
<td>3,9</td>
<td>6:52</td>
<td>9.8</td>
</tr>
<tr>
<td>B.ND.4</td>
<td>51</td>
<td>0</td>
<td>0:00</td>
<td>6:38</td>
<td>21</td>
<td>41,2</td>
<td>6:17</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>27,4</td>
<td>7:10</td>
<td>10.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7:10</td>
<td>19.6</td>
<td>6:48</td>
<td>11.8</td>
</tr>
<tr>
<td>B.ND.5</td>
<td>50</td>
<td>2</td>
<td>3:05</td>
<td>6:44</td>
<td>19</td>
<td>38,0</td>
<td>6:20</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>28,0</td>
<td>7:17</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7:17</td>
<td>22.0</td>
<td>6:49</td>
<td>12.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6:20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.14: Solutions for problem STCP-B

Figure 7-11 shows the variation of the distribution of the duty types as the number of duties in the solutions decreases. We can see that a reduction of only three duties (from 53 in the current solution to 50 in B.ND.5) implies a large variation in the distribution of the different duty types.

Table 7.15 shows the alternative solutions for problem STCP-C, the smaller one in the STCP problem set in terms of number of duties. The Agg model has produced one solution (C.AG.1) with the same number of duties as the current solution, but it includes 5 long break duties. The ND model has also provided one solution (C.ND.1) with the same number of duties and no leftovers, but without long
break duties. For this solution we have used criteria 2, 3 and 8. Next, in order to focus the search in the reduction of the number of duties, we have periodically unselected and selected criterion 8. Using this procedure we have found three solutions with less duties, but with more long break duties.

<table>
<thead>
<tr>
<th>Solution</th>
<th>N. duties</th>
<th>N. leftov.</th>
<th>Dur leftov.</th>
<th>Avdr StrSpl</th>
<th>Straight duties</th>
<th>Split duties</th>
<th>Long break duties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N.</td>
<td>%</td>
<td>Avdr</td>
<td>N.</td>
<td>%</td>
<td>Avdr</td>
<td>N.</td>
</tr>
<tr>
<td>C.STCP</td>
<td></td>
<td>0</td>
<td>6:29</td>
<td>24</td>
<td>64,9</td>
<td>6:11</td>
<td>13</td>
</tr>
<tr>
<td>C.ND.4</td>
<td>34</td>
<td>2</td>
<td>3:30</td>
<td>6:52</td>
<td>14</td>
<td>41,2</td>
<td>6:15</td>
</tr>
<tr>
<td></td>
<td>37</td>
<td>0</td>
<td>0:00</td>
<td>6:23</td>
<td>23</td>
<td>62,2</td>
<td>6:04</td>
</tr>
<tr>
<td></td>
<td>37</td>
<td>0</td>
<td>0:00</td>
<td>6:28</td>
<td>25</td>
<td>57,5</td>
<td>6:04</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>0</td>
<td>0:00</td>
<td>6:39</td>
<td>19</td>
<td>52,8</td>
<td>6:10</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>0</td>
<td>0:00</td>
<td>6:46</td>
<td>16</td>
<td>45,7</td>
<td>6:13</td>
</tr>
</tbody>
</table>

Table 7.15: Solutions for problem STCP-C

Figure 7-12 shows the variation in the number of each duty type as the number of duties in the solutions decreases. Solution C.ND.4 has more split duties than straight duties, as well as 4 long break duties, but the number of duties has been reduced to 34 (instead of 37 in the current solution).

### 7.6.2 Concluding remarks

Bus driver scheduling at STCP is very different from the procedure at Carris, presenting other type of difficulties and challenges. To avoid leftovers is of high priority. In all analysed problems, our approach
has proved to produce solutions with no leftovers. The main difficulty was to reduce the number of duties without introducing many long break duties. Our procedure was based on interactively activating or deactivating the criterion that penalises the duty types. Naturally, due to the rules defined, the number of duties could only be reduced when the number of split duties increases. However, the average duration of straight and split duties is always close to the expectations of planners.

The Agg model has produced solutions that are comparable to the current solutions in only a few generations. In fact all these solutions have been obtained in less than 15 minutes. However, this model was not able to produce interesting alternative solutions. The ND model is slower (the average real time needed to obtain each of the presented solutions was 30 minutes) but it has provided several alternative solutions, although some of them contain a few minutes of uncovered work.

One important feature of STCP problems is that meal break validation must be performed on straight and split duties. The mealbreak period and the minimum meal duration must be strictly respected.

7.7 Bus driver scheduling in SMTUC

SMTUC is a medium size transport company owned and managed by the municipality of Coimbra and operates with 130 buses and 295 drivers. SMTUC has started to use the GIST System in 2000 and, currently, all driver schedules are produced with the support of GIST. For a given planning period, the bus driver scheduling problem instance involves all route schedules together for a particular day type (namely, weekdays, Saturdays and Sundays). Hence, the planning team periodically solves three bus
drivers problems, which may take several weeks to conclude.

Bus driver scheduling at SMTUC involves three duty types: straight duties, split duties and long break duties, whose rules are summarised in Table 7.16. The average duration of all duties in a schedule must be as close as possible to 7h00m. The rules imposed to the duty types are considerably more flexible than those of Carris or STCP. First, the minimum duration of the stretches and the minimum duration of the duties are not constrained to any specific value and, second, two duty interruptions are allowed on all three duty types. However, it is clear that planners wish to minimise the number of duty interruptions and to maximise the length of stretches and duties. The large number of buses involved and the flexibility of the rules hugely increases the number of feasible duties but the problem size is still manageable by GenT, since the pieces-of-work defined by the planner are considerably long and few (see Table 7.17).

At SMTUC both split and long break duties are preferred to straight duties, split duties being the most desirable. Leftovers may occur and planners assign them as extra work to normal drivers or as normal work to extra drivers.

The lunch period begins at 10h00m and ends at 16h00m while the dinner period is from 16h00m to 24h00m. Split and long break duties must have one meal (of, at least, one hour) during the break, while straight duties must include time for a meal of half an hour before or after the driving time.

<table>
<thead>
<tr>
<th>Rule description</th>
<th>Straight duties</th>
<th>Split duties</th>
<th>Long break duties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Number of stretches</td>
<td>Min Max</td>
<td>Min Max</td>
<td>Min Max</td>
</tr>
<tr>
<td>Stretch length</td>
<td>0:00 7:00</td>
<td>0:00 5:00</td>
<td>0:00 5:00</td>
</tr>
<tr>
<td>Break length</td>
<td>0:30 1:00</td>
<td>1:00 2:00</td>
<td>2:00 10:00</td>
</tr>
<tr>
<td>Meal duration</td>
<td>5:00 1:00</td>
<td>8:00 0:00</td>
<td>7:00 0:00</td>
</tr>
<tr>
<td>Normal duty length</td>
<td>5:00 1:00</td>
<td>5:00</td>
<td></td>
</tr>
<tr>
<td>Driving time</td>
<td>7:00 8:00</td>
<td>0:00 16:00</td>
<td></td>
</tr>
<tr>
<td>Total working time</td>
<td>0:00 7:00</td>
<td>0:00 0:00</td>
<td>0:00 2:00</td>
</tr>
<tr>
<td>Total normal spreadover</td>
<td>2 0</td>
<td>2 0</td>
<td></td>
</tr>
<tr>
<td>Number of duty interruptions</td>
<td>0 2</td>
<td>0 2</td>
<td></td>
</tr>
<tr>
<td>Length of duty interruptions</td>
<td>0:10 0:59</td>
<td>0:10 0:59</td>
<td>0:10 0:59</td>
</tr>
<tr>
<td>Number of vehicle changes</td>
<td>0 2</td>
<td>0 2</td>
<td></td>
</tr>
<tr>
<td>Mealbreak validation</td>
<td>Yes</td>
<td>In the break</td>
<td>In the break</td>
</tr>
</tbody>
</table>

Table 7.16: Duty types for SMTUC

The main objectives of the company consist in minimising the number of duties and leftovers, while keeping an average duration of duties close to 7h00m, and the percentage of split duties as high as possible.
7.7.1 Computational results

As referred, for each planning period, SMTUC has basically three driver schedules, one for each day type. We have selected the largest planning period (school time) and the associated driver scheduling problems (for each day type), which are presented in Table 7.17. Each problem contains the trips of all lines in operation. Note that on Sundays (problem SMTUC-A) the offer is substantially smaller when compared with weekdays (the number of buses goes from 113 to 25).

<table>
<thead>
<tr>
<th>Problem</th>
<th>Line</th>
<th>Number of buses</th>
<th>Total duration</th>
<th>Number of relief points</th>
<th>Number of pieces-of-work</th>
<th>Average duration of pieces-of-work</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMTUC-A</td>
<td>All-Sundays</td>
<td>25</td>
<td>304:51</td>
<td>5</td>
<td>113</td>
<td>2:41</td>
</tr>
<tr>
<td>SMTUC-B</td>
<td>All-Saturdays</td>
<td>61</td>
<td>642:58</td>
<td>9</td>
<td>215</td>
<td>2:59</td>
</tr>
<tr>
<td>SMTUC-C</td>
<td>All-Weekdays</td>
<td>113</td>
<td>1389:00</td>
<td>13</td>
<td>445</td>
<td>3:07</td>
</tr>
</tbody>
</table>

Table 7.17: SMTUC problem set

In order to control the number of feasible generated duties, we have reduced to one both the maximum number of duty interruptions and the number of vehicle changes. In all the problems we have kept the original set of pieces-of-work, since their average duration is considerably large.

The minimisation of the number of duties and the minimisation of the number of leftovers were the main criteria used. Nevertheless, we have also taken into account the preference for split duties, although this criterion was selected and unselected along the process, according to the percentage of the different duty types in the intermediate solutions.

The maximum actual time needed to interactively produce each solution with the Agg model, was 1 hour while, for the ND model, we have let the algorithm run for 2 hours, due to the large size of the problems.

In Table 7.18, we present the results obtained for problem SMTUC-A (Sundays). Since straight duties are undesirable and the current solution contains only one duty of this type, we have tried to obtain solutions with no straight duties, by imposing a minimum percentage of 100% to split duties. Solution A.AG.1 has one duty and one leftover less than the current solution and it was obtained by minimising the number of duties (criterion 2), the number of leftovers (criterion 3) and the gap from the desired average duration of duties (criterion 4), which was set to 7 hours. The ND model has produced three different solutions that can be analysed in Figure 7-13.
Table 7.18: Solutions for problem SMTUC-A

Although solution A.ND.1 contains the same number of duties as the current solution (47) it has 3 leftovers less. Solution A.ND.3 has two duties less and the same number of leftovers. All solutions obtained by GenT have an average duration closer to 7 hours than the current solution.

Figure 7-13: Distribution of duty types in ND solutions for problem SMTUC-A

In Table 7.19, we present several alternative solutions to problem SMTUC-B. The current solution is very similar to the solution obtained with the Agg model, although solution B.AG.1 contains two leftovers less and a slightly smaller percentage of split duties. With solutions B.ND.1 to B.ND.4, obtained with the ND model, we have tried to minimise the number of duties and leftovers, while keeping a percentage of split duties similar to the current solution. We have managed to reduce the number of duties as well as the number and duration of the leftovers, without significantly increasing the percentage of straight duties. The last three solutions in Table 7.19 (B.ND.5 to B.ND.7) were also obtained by the ND model,
but here we have mainly tried to minimise the number of duties and leftovers. We have managed to obtain a solution with no leftovers, but the percentage of straight duties has increased from 22,11% in the current solution, to 31,58% in solution B.ND.7. Moreover, it should be noted that the average duration of the duties in all solutions is very close to 7 hours, although the average duration of straight duties is considerably smaller than in the current solution (7h23m). We believe that this is mainly due to the smaller number of duty interruptions in GenT solutions, since the interruption time (although it does not correspond to driving time) is added to the duty length.

<table>
<thead>
<tr>
<th>Solution</th>
<th>N. duties</th>
<th>N. lefov.</th>
<th>Dur lefov.</th>
<th>Avdur duties</th>
<th>N. Split duties</th>
<th>N. Straight duties</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.SMTCU</td>
<td>95</td>
<td>7</td>
<td>9:55</td>
<td>7:08</td>
<td>74 77,89</td>
<td>21 22,11</td>
</tr>
<tr>
<td>B.AG.1</td>
<td>95</td>
<td>5</td>
<td>5:40</td>
<td>6:59</td>
<td>71 74,70</td>
<td>24 25,26</td>
</tr>
<tr>
<td>B.ND.1</td>
<td>93</td>
<td>2</td>
<td>7:35</td>
<td>7:06</td>
<td>73 78,49</td>
<td>20 21,51</td>
</tr>
<tr>
<td>B.ND.2</td>
<td>94</td>
<td>3</td>
<td>6:10</td>
<td>7:04</td>
<td>73 77,66</td>
<td>21 22,34</td>
</tr>
<tr>
<td>B.ND.3</td>
<td>95</td>
<td>2</td>
<td>1:20</td>
<td>7:03</td>
<td>71 74,74</td>
<td>24 35,26</td>
</tr>
<tr>
<td>B.ND.4</td>
<td>92</td>
<td>5</td>
<td>8:10</td>
<td>7:10</td>
<td>69 75,00</td>
<td>23 25,00</td>
</tr>
<tr>
<td>B.ND.5</td>
<td>94</td>
<td>1</td>
<td>2:30</td>
<td>7:06</td>
<td>66 70,21</td>
<td>28 29,79</td>
</tr>
<tr>
<td>B.ND.6</td>
<td>95</td>
<td>1</td>
<td>3:50</td>
<td>6:59</td>
<td>66 69,47</td>
<td>29 30,53</td>
</tr>
<tr>
<td>B.ND.7</td>
<td>95</td>
<td>0</td>
<td>0:00</td>
<td>7:02</td>
<td>65 68,42</td>
<td>30 31,58</td>
</tr>
</tbody>
</table>

Table 7.19: Solutions for problem SMTUC-B

The set of alternative solutions is shown in Figure 7-14, where we can see the distribution of the duty types for each solution. It can be noted that the first four ND solutions are very similar to the current solution (B.SMTCU), concerning the distribution of duty types. The last three solutions show a significant increase in the number of straight duties along with a decrease of the number of leftovers.

Problem SMTUC-C is the largest real problem that we have tested. In order to control the number of generated duties, we have imposed a minimum duration of 1h30m to each stretch and a minimum duration of 5h00m to each duty. Besides, since the current solution contains only 8 straight duties, we have not allowed the generation of this duty type.

Table 7.20 presents the set of alternative solutions obtained for this problem. In solutions C.AG.1 and C.ND.1 we have tried to minimise the number of duties and leftovers and, simultaneously keep a high percentage of split duties. In fact, solution C.ND.1 has no leftovers and two duties less than the current solution, whith the same percentage of split duties. The other solutions (C.ND.2 to C.ND.7) correspond to several solutions obtained in the same run, by trying to minimise the number of duties while keeping their average duration close to 7h00m. The penalisation assigned to the duty types was
set up interactively, in order to control the increment of long break duties. In fact, we have managed to produce a solution with less five duties (210) and no leftovers, and several solutions with a higher reduction on the number of duties but with some uncovered work.

As shown in Figure 7-15, reducing the number of duties mainly affects the number of split duties, which also decreases. Except for solution C.ND.1, where the penalisation of the long break duties has played an important role, the remaining solutions show a small variation on the number of long break duties.
7.7.2 Concluding remarks

The SMTUC problem set has allowed us to test our approach for large instances that include a complete planning period. Due to the large size of the original problems, we have been forced to impose certain values for some rules, in order to control the number of generated duties. Our approach has proved to be able to significantly reduce the number of duties (particularly in the largest problem) as well as the number of leftovers, while keeping an average duration of duties close to the desired value.

Although SMTUC is a medium size company, their bus driver scheduling problems are larger than those of Carris and STCP. Moreover, these problems are changed very often, in order to incorporate minor changes in the offer, with implications in the driver schedules. Manual driver scheduling in problems of this dimension is clearly a very difficult task. GenT is able to provide several interesting alternative solutions that can be analysed, improved or manipulated manually by the planners.

7.8 Summary

The main goal of our multiobjective GA approach was to provide mass transit companies with a friendly, interactive and flexible software tool able to quickly produce good BDSP solutions, and thus benefit from the planners' know-how and experience.

For testing purposes, we have selected three Portuguese companies with problems of different sizes and complexities. As already stated in Chapter 2, the larger companies have smaller problems. In fact, while Carris has a different driver scheduling problem for each route, STCP works with groups of three
or more routes, and in SMTUC the driver scheduling problems include all routes of the network for each
day type.

For each company, we have started by analysing the particular bus driver scheduling process. Then
we have identified their main objectives and soft constraints, and we have applied GenT in a simulated
real environment. Taking into account planner expectations, we have imposed the limit of two hours,
in real elapsed time, to obtain a final set of non-dominated solutions. This limit was seldom reached,
except for SMTUC problems, that are considerably larger than the others.

We must emphasise that we have not chosen a "best" solution for any problem. Rather, we have
suggested a set of alternative solutions, leaving the final choice to the company planners. The "right"
decision depends on several aspects that are beyond our knowledge, such as management strategies,
internal operational policies or union agreements. Nevertheless we have always presented one solution
that is as close as possible to the one currently available.

We have run both the Aggregate (Agg) model and the Non-Dominated (ND) model for all problems.
The main conclusion is that, although the Agg model is faster, the ND model is more effective in
producing alternative solutions. Moreover, the ND model is easier to parametrise. In fact, in the
Agg model, the user has to choose the weights to assign to each criterion according to its relative
importance. Although this choice is implicit in the planner's mind, its translation into criterion weights
is not easy. On the contrary, in the ND model, he/she just selects criteria and at the end, he/she
analyses the non-dominated solutions obtained. The chosen solution reflects his/her implicit weighting
function. Naturally, in either situation, some time and iterations are needed to adjust the criteria for
each company. The interactive nature of GenT allows the users to tune the parameters and criteria
while the algorithm is running.

GenT has proved to be a valuable tool in producing good quality BDSP solutions. For all tested
problem instances we have managed to produce a solution similar to the one currently available and
several others that represent interesting compromises between the objectives stated by the planners. As
expected, we can conclude that reducing the number of duties increases the number of undesirable
features in the solution, but the real trade-off between the several objectives involved can now be effectively
measured and analysed. Moreover, the impact of changes in the duty rules can now be studied and
analysed in order to support future decisions at the tactical and strategical levels.
Chapter 8

Conclusions and Future Work

This thesis has addressed the bus driver scheduling problem as a critical component of the operational planning process in mass transit companies. A particular emphasis was given to the case of Portuguese companies. A review of the most successful decision support systems for the operational planning has also been provided. The GIST system, that is being used by the major Portuguese mass transit companies, has been described in more detail.

The main currently available methods for driver scheduling have been described. Most of them are based on single objective optimisation based on the minimisation of a cost function and several important objectives and soft constraints are not usually taken into account. We have proposed a new perspective on modelling the BDSP that is closer to the real problem and to the planners work. We have identified and discussed several different objectives and side constraints used by mass transit companies in their bus driver scheduling process. In order to tackle these simultaneous and conflicting objectives, a multiobjective approach based on genetic algorithms has been proposed.

We have developed two different GA models, namely the Aggregate (Agg) model and the Non-Dominated (ND) model. Both models incorporate the same set of criteria and a common set of operators. While the Agg model groups all criteria in a single objective function and provides a single solution, the ND model uses the concept of Pareto dominance providing a set of alternative trade-off solutions. In this model an elite set of non-dominated solutions is maintained and clustering procedures have been used to reduce it. Dynamic crossover and mutation rates have been applied and several intensification and diversification techniques have been implemented. All the parameters of the algorithm can be fully controlled by the user.

The Agg model was tested on benchmarking set partitioning problems and has produced satisfactory results. Moreover, both models have been integrated in an interactive application, GenT, that has been
embedded in the GIST system. We have tested our approach in a set of real problems from three Portuguese mass transit companies. The ND model has achieved better results than the Agg model in providing a set of competitive alternative solutions for all problems tested.

8.1 Main achievements of this thesis

Bus driver scheduling is an important process in the management of a mass transit company, usually involving the work of several planners. As a result of this process, important savings can be made with no deterioration of the service quality. On the other hand, the quality of the driver duties is currently an important aspect that must also be taken into account. Any advances in methods that support the decision making process of the planners can therefore have a significant impact. Thus, in this context we view the major contributions of our work as follows:

1. The research has provided a new and valuable insight into the complex nature of the bus driver scheduling problem. Modelling the bus driver scheduling problem as an optimisation problem based on the minimisation of the number of duties (or the corresponding costs) has been the most common approach throughout the years. However, duty costs are not always easy to define and a driver schedule is not only composed by duties. In many companies, it is not possible to assign the complete bus work to drivers and often driver schedules include some blocks of uncovered work (leftovers). Furthermore, the minimisation of the number of duties is not always the most important objective. In fact, the quality of a driver scheduling solution depends not only on the individual quality of each duty, but also on several other features that involve the whole solution. Often, an optimal solution concerning duty costs cannot be implemented in a real context since it does not take into account some particular requirements for the whole solution. Decision making in bus driver scheduling is clearly a multiobjective process, reflected in our perspective for modelling the real problem.

2. A novel GA approach to the BDSP has been developed and tested. Several GA components and knowledge-based crossover, mutation and repairing operators have been designed. The Agg model has been tested on standard benchmarking problems and has produced satisfactory results. The tests performed on real problems have shown that while the Agg model produces solutions comparable to the current ones, the ND model is more effective in the selection of a subset of varied non-dominated solutions. We may conclude that the ND model represents a high valuable contribution to support the driver scheduling planning process.

3. The GA models were integrated in an advanced interactive tool (GenT) which can be used either
as a stand-alone application or integrated in a decision support system. GenT can be used in different contexts. At the operational level, GenT represents a valuable tool to quickly produce alternative solutions. At the tactical level it can be used to simulate different operating scenarios and evaluate the impact of changes in driver rules. Furthermore, GenT can be also applied by R&D teams to support the research and testing of new techniques as new genetic operators and new criteria can be easily incorporated.

4. Our general multiobjective approach can be extended to other scheduling problems involving several different objectives. In fact, train or airline driver scheduling problems are natural extensions of our problem, but we believe that the approach could be also applied to other scheduling problems, such as vehicle scheduling or rostering. Naturally, different crossover and mutation operators should be devised and a different set of criteria should be found, but in principle the general structure of the approach could be easily adapted.

8.2 Future research

As it is natural, a research work such the one described in this thesis is never fully closed. Several issues do clearly deserve further research. In the following points we briefly present some of those issues:

1. Our approach was based on a generate and select procedure, on which each solution is composed by a subset of duties selected from a large set of feasible duties that were previously generated. The procedure that generates this large set of duties can be exhaustive or it can be based on a set of rules that exclude some potentially uninteresting duties. We have used an exhaustive procedure that has been developed by the GIST development team. However, it could be interesting to develop a new duty generator routine based on "intelligent" rules. In such a procedure the number of duties to generate could be parametrised by the user. If no limit is imposed, we have an exhaustive procedure while a tight value will impose the application of certain rules that would only select a representative subset of the feasible duties. The set of rules would be also parametrised by the user. Furthermore, this new procedure would also allow the user to add new driver scheduling rules. In fact we have observed that the set of rules is always being updated and incremented.

2. The set of feasible duties is generated once before the algorithm execution and it never changes. If the user wants to add another duty type to the set of feasible duties, or to slightly change a single parameter of a duty type, the whole process must be started from scratch. We think it would be interesting to integrate the generation process in the algorithm. Hence, at any moment the user could add or remove duties to the set of generated duties. The process would be fully controlled by
the user who would say if he wishes to keep the duties that had already been generated or remove them. He could also parametrise the number of new duties of each duty type to be added to the new set. In a certain sense, this new approach would be similar to column generation, since the generation of duties occurs when the algorithm is running.

3. An alternative approach could be to integrate the genetic algorithms with fuzzy logic. In this approach the concept of feasible duty would be fuzzified, in the sense that a duty is not just feasible or infeasible. Instead, feasibility would be viewed as a fuzzy criterion with duties that are more feasible than others. This approach seems very natural, since planners implicitly apply this concept. For example, a constraint violation of one or two minutes is not critical, but if it is of ten minutes the duty is probably clearly infeasible. Further research on this topic would be interesting.

4. It would be interesting to test the modifications that we have made to the SPEA algorithm in standard benchmarking problems for multiobjective optimisation. In fact, the application of some new concepts, such those of population density and dominance sharing can be interesting alternatives to standard fitness sharing.

5. Our approach could be extended to vehicle scheduling and also to an integrated approach to vehicle and driver scheduling. These problems also involve multiple criteria, although they are usually formulated as single objective optimisation problems. However, our experience with several Portuguese mass transit companies has shown that certain objectives and soft constraints are not being taken into account by these approaches. Identifying the main criteria involved, designing a new coding scheme and developing new genetic crossover and mutation operators would be the main topics of research in this area.
Appendix A

Performance of crossover operators

The performance of a genetic crossover operator is difficult to define and to measure. From our point of view, it can be defined as the contribution of the operator for an effective search of the solutions space. The difficulty of evaluating this contribution has lead us to implement different performance measures that hopefully contribute to a better understanding of the behavior of such operators. We have used SPSS 11.0 for Windows to perform statistical tests and compute the influence of different factors in the performance of the crossover operators that we have implemented, namely Union, Two-Point and Swap operators, which have been described in Chapter 6.4.3.

Several questions can be posed concerning the performance of crossover operators. Some of these questions are:

1. How to measure the performance of a crossover operator? How are the different performance measures related to each other?

2. Which factors influence the performance of a crossover operator? Is it affected by any characteristic of each particular instance? In this case, which are these characteristics and how do they affect the performance of the operator?

3. The Union crossover operator can be used with a different number of parents. Does the number of parents affect the performance of this operator?

4. How the different operators can be compared to each other? Can we say that they perform differently? In this case, can we say that one is better than another?

In Section A.2.1, we describe how the sample data set was obtained and the performance measures that we have used to test the performance of the crossover operators. In Section A.3, we use sev-
eral statistical techniques to test the influence of different factors on the performance of the crossover operators.

In these experiments, we have used the GA based on the Non-Dominated (ND) model and default values for all the parameters. In the generation of the initial population duplicates were not allowed, but in the replacement of the old individuals duplication can occur. When premature convergence is detected, the entire population is renovated (it is generated a new population), but the external set is preserved. The duty types have not been considered for the selection of the eligible duties.

A.1 Measuring the performance of the operators

To measure the contribution of a crossover operator for an effective search of the solutions space, we compare the child produced by the operator with its parents and also with the current population. We have implemented three different performance measures that are computed at each run of the algorithm: fitpar and fitmed relate the child with the current population and domin relate the child with its parents.

Let $cr\_op \in \{Union, Two\_Point, Swap\}$ be a crossover operator and $n_t(cr\_op)$ be the number of times that it is applied at generation $t$.

$C_t$ is a chromosome of a population $P_t$ and $Parents(C_t)$ is the set of parents of $C_t$.

The $fitpar_t(cr\_op)$ performance measure is defined as the percentage of children produced by $cr\_op$ with a fitness larger than the fitness of one of the parents (see expression A.1).

$$fitpar_t(cr\_op) = \frac{|\{C_t : \exists parent \in Parents(C_t) : fit(C_t) > fit(parent)\}|}{n_t(cr\_op)} \quad (A.1)$$

$fitmed_t(cr\_op)$ measures the percentage of children produced by $cr\_op$ with a fitness larger than the average fitness of the current population (see expression A.2).

$$fitmed_t(cr\_op) = \left\lfloor \frac{\left\{ C_t : fit(C_t) > \frac{1}{\text{pop}_t} \sum_{C_t \in P_t} fit(C_t) \right\}}{n_t(cr\_op)} \right\rfloor \quad (A.2)$$

$domin_t(cr\_op)$ corresponds to the percentage of children produced by $cr\_op$ that dominate, in the Pareto sense, one of the parents (see expression A.3).

$$domin_t(cr\_op) = \frac{|\{C_t : \exists parent \in Parents(C_t) : F(C) \leq F(parent)\}|}{n_t(cr\_op)} \quad (A.3)$$

These three measures correspond to different views of the concept of performance of a crossover operator.
A.2 Union crossover operator

The Union crossover operator can use two or more of parents. Increasing the number of parents also increases the execution time of the algorithm. In this section we show that the number of parents does not affect the performance of the operator and, based on this conclusion, we have implemented the Union Crossover operator using the minimum number of parents (two parents).

A.2.1 Experimental design

To obtain the sample data set we arbitrarily have chosen 10 problems from different bus companies and for each run we have used the ND genetic algorithm with the default parameters for 100 iterations. For each problem, the Union Crossover operator was applied with two, three and four parents and for each case the algorithm has run 20 times. Each case of the sample data set includes information about the problem, the number of parents used by the operator and the values of the three performance measures for one run of the GA. Hence, the sample data set is composed by 600 cases (10 problems × 20 runs × 3 instances of number of parents).

In Table A.1, we present the correlation coefficients between these performance measures. We can see that all three of them have a significant correlation, but while $fitpar$ and $domin$ are positively correlated, $fitmed$ has negative correlation coefficients with the other two performance measures. This means that, when $fitpar$ and $domin$ are high, $fitmed$ is low and, in fact, we can admit that it is more difficult for a chromosome to be better than its parents when the average fitness of the population is high. The mean value for $fitmed$ shows that 73% of the children have a fitness higher the average fitness of the population, while the mean values for the other performance measures show that for almost 40% of the cases, the children produced are "better" than their parents. These values confirm that the parents are above average individuals and shows that this operator effectively searches the solutions space.

Furthermore, we have investigated the internal consistency of the three measures, to see if they contribute consistently to the concept of "performance". For this purpose, we have performed a Reliability Analysis using the Alpha (Cronbach) model to determine the extent to which these measures are related to each other. This method provides an overall index of the repeatability or internal consistency of the scale as a whole. We obtained a reliability estimates (Alpha) of 0.7576, which is considered a good indicator of the reliability of the performance measures.

Since all performance measures are highly correlated and contribute to the same concept, we have used the principal components method in order to reduce the number of variables in the data set. This method finds linear combinations of variables (components) that accounts for as much variation in the original variables as possible and these components can be used to replace the original variables. The
application of this method to the three performance measures returned a single component, denoted by \textit{pmd\_parents} that will be further used as the dependent variable instead of the three measures, since this variable resumes our performance concept.

<table>
<thead>
<tr>
<th></th>
<th>\textsc{fitpar}</th>
<th>\textsc{fitmed}</th>
<th>\textsc{domin}</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textsc{fitpar} Mean</td>
<td>0.40</td>
<td>Pearson Correlation</td>
<td>1</td>
</tr>
<tr>
<td>Std.Dev.</td>
<td>0.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>\textsc{fitmed} Mean</td>
<td>0.73</td>
<td>Pearson Correlation</td>
<td>-0.714</td>
</tr>
<tr>
<td>Std.Dev.</td>
<td>0.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>\textsc{domin} Mean</td>
<td>0.36</td>
<td>Pearson Correlation</td>
<td>0.942</td>
</tr>
<tr>
<td>Std.Dev.</td>
<td>0.03</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.1: Correlation between performance measures

A.2.2 Influence of problem density

Usually, the complexity of a BDSP instance is related with its dimension (size of the problem matrix), in particular with the number of generated duties. Remember that, in the mathematical formulation of the BDSP, the problem matrix is a 0-1 matrix where the columns represent the generated duties and the rows represent the pieces-of-work. Each entry of this matrix has a 1 if the duty of that column covers the piece-of-work of that row, and 0 otherwise. The density of a problem is given by the rate of 1's in the problem matrix.

We have chosen ten problem instances from different companies, with different sizes (from 2000 to 15000 columns) and different complexities. Due to their particular structures (topology and rules for duties generation) all these problems have very low densities (from 0.02 to 0.1). However, we have noticed that the density of the problem constitutes a factor with a significative influence on the performance of the operators [33]. Hence, we have categorised the problems according to three density categories. In Figure A-1, we present the case summaries for each performance measure according to the density categories. Figure A-2 helps us to observe the influence of the problem density in the three performance measures. Each boxplot chart represents the median and quartiles for a single performance measure and three categories of density.

We can see that, in fact, the density of the problems influences all three performance measures of the operator. \textit{fitpar} and \textit{fitdom} decrease when the problem density increases, which was somehow expected, since the complexity of the problem increases with density. In fact, the solutions space is bigger and more complex when the density is high than when the density is low (the number of duties that potentially can cover each piece-of-work is smaller when the density is low). \textit{fitmed} slightly increases with the density of the problems.
A.2. UNION CROSSOVER OPERATOR

<table>
<thead>
<tr>
<th>Categories for density</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>FITPAR</td>
<td>0.42</td>
<td>0.01</td>
<td>178</td>
</tr>
<tr>
<td></td>
<td>0.41</td>
<td>0.02</td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>0.38</td>
<td>0.02</td>
<td>178</td>
</tr>
<tr>
<td></td>
<td>0.40</td>
<td>0.02</td>
<td>600</td>
</tr>
<tr>
<td>FITMED</td>
<td>0.72</td>
<td>0.01</td>
<td>178</td>
</tr>
<tr>
<td></td>
<td>0.73</td>
<td>0.01</td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>0.73</td>
<td>0.01</td>
<td>178</td>
</tr>
<tr>
<td></td>
<td>0.73</td>
<td>0.01</td>
<td>600</td>
</tr>
<tr>
<td>DOMIN</td>
<td>0.38</td>
<td>0.01</td>
<td>178</td>
</tr>
<tr>
<td></td>
<td>0.36</td>
<td>0.02</td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>0.33</td>
<td>0.02</td>
<td>178</td>
</tr>
<tr>
<td></td>
<td>0.36</td>
<td>0.03</td>
<td>600</td>
</tr>
</tbody>
</table>

Figure A-1: Case summaries for the performance measures

Figure A-2: Boxplots of the three performance measures for categories of problem densities

A.2.3 Influence of the number of parents

Now, we intend to test if the number of parents affects significantly the performance of the Union crossover operator. Since we have concluded that the problem density influences the performance, we have performed a univariate analysis with two factors in order to measure the influence of the number of parents and of the problem density on the performance of the Union operator. Figure A-3 presents two plots with the variation of the pmu_nparents with the number of parents and the categories for density. In fact, we can observe that the performance worsens when the density increases, for all number of parents but there is no clear connection between the performance and the number of parents within each category of density.

We have used the SPSS software to perform an univariate analysis with two factors, whose results are presented in Table A-4. The dependent variable is pmu_nparents and the factors are the number of
Figure A-3: Boxplot and mean plot for the pmd_nparents variable

parents and the categories for density (a random factor). The null hypothesis is that these factors do not affect the performance. The statistical test confirms that, for a significance level of 0.05, the number of parents does not affect the performance (since $\text{Sig} = 0.319$, the null hypothesis is not rejected) but it is clearly influenced by density (since $\text{Sig} = 0.000$, the null hypothesis is rejected). However, both factors together do not justify a variation on the performance of the Union crossover operator (as $\text{Sig} = 0.118$, the null hypothesis is not rejected).

Tests of Between-Subjects Effects

<table>
<thead>
<tr>
<th>Source</th>
<th>Type III Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPARENTS</td>
<td>Hypothesis</td>
<td>2</td>
<td>1.476</td>
<td>1.535</td>
<td>.319</td>
</tr>
<tr>
<td>NDENS3</td>
<td>Hypothesis</td>
<td>2</td>
<td>141.683</td>
<td>146.7</td>
<td>.000</td>
</tr>
<tr>
<td>NPARENTS * NDENS3</td>
<td>Hypothesis</td>
<td>4</td>
<td>1.847</td>
<td>1.118</td>
<td></td>
</tr>
</tbody>
</table>

Figure A-4: Test for the effect of the number of parents and density on the performance (significant under 5%)

A.2.4 Conclusions

In this section we tested the influence of the number of parents in the Union crossover operator on its performance. We have performed a Reliability Analysis to the three performance measures defined in
Section A.1 and they were replaced by a single performance measure named \textit{pmd\_npa}rents after the application of the Principal Components method. We have noticed that the problem density constitutes a factor with influence on the performance of the operator and we have identified three categories for the density. Then, we have performed a univariate analysis with two factors (number of parents and categories for density) to test how they affect the performance of the Union crossover operator. We have concluded that both factors together do not affect significantly the performance.

In conclusion, increasing the number of parents does not influence the performance of the Union crossover operator and, based on this result, we have chosen to use the minimum number of parents (two parents) in the implementation of this operator.

A.3 Comparing the crossover operators

In this section we investigate the relationships between the performance of the three crossover operators. Since they are based on different levels of problem knowledge (see Section 6.4.3), we want to test if they perform differently and if the performance increases when we add information to the operators. In section A.3.1, we describe the procedure that we have used to obtain the sample data set. In Section A.3.2, we discuss the influence of the problem density on the performance of the operators and in Section A.3.3 we compare the performance of the three crossover operators.

A.3.1 Experimental design

To obtain the sample data set for this experiment, we have chosen 30 instances from different bus companies and for each run we have used the ND model with the default parameters for 100 iterations. For each instance the GA has run 10 times and, for each run we compute the three performance measures for each crossover operator. The sample data set is, then, composed by 900 cases (30 problems \(\times\) 10 runs \(\times\) 3 crossover operators). In fact, we have reduced this number to 887 cases, because we have extracted the extreme values, that could introduce some bias in the analysis.

In Table A-5, we present the case summaries for this data test. The right-most column shows the summaries for the three performance measures within each operator and the last three lines correspond to the totals for each performance measure. We can see that, globally, the performance values are very similar to those presented in Table A.1 for the Union crossover operator (mean values near 0.4 for \textit{fitpar} and \textit{domin} and near 0.7 for \textit{fitmed}). Once again, we have observed that \textit{fitmed} performs differently than the other two performance measures. While \textit{fitpar} and \textit{domin} present the worst values for the Union operator and the best values for the Swap operator, \textit{fitmed} presents the best values for the Union operator and the worst for the Two-Point operator.
Figure A-5: Case summaries for the performance of the operators

We have also performed a Reliability Analysis using the Alpha (Cronbach) model to determine the internal consistency of the three measures and we obtained a reliability estimates (Alpha) of 0.7489, which is considered a good indicator of the reliability of the performance measures. Similarly to the previous case, we have used the Principal Components method to test if we could reduce the number of variables in the data set. The application of this method to the three performance measures has extracted a single component, that we denoted by pmd_opr that will be used as the dependent variable in the statistical analysis.

A.3.2 Influence of the density of the problems

The sizes of the instances used in this data vary from 400 to almost 20000 generated duties (columns) and all of the problems have low density (from 0,02 to 0,2). We have verified that the size of the problem does not affect the performance of the operators, yet the density does. We have divided the cases in the data set into three categories according to their density and, as we can see in Table A-5 and in Figure A-6, the global performance of fitpar and domin clearly degrades when the density increases. However, concerning to fitted, it is not affected by the density of the problems (see the central boxplot chart in Figure A-6).

We can conclude that the ability of the crossover operators to produce children better than their parents deteriorates when the density increases, but their ability to produce children better than the average is not affected by density and is kept at a high rate (near 70%).

A.3.3 Performance of the crossover operators

Analysing separately each crossover operator, we find substantial differences on their performance. In Figure A-7 we present the boxplots for each crossover operator within each performance measure. We
observe that the Swap operator has the best performance for fitpar and domin, while the Union operator is the worst for the same performance measures. Concerning fitmed, we can see that Union operator performs better than the other two. These values also show that more than 70% of children produced by Union operator are better than the average of the population, but less than 40% of them are better than their parents. None of the other two operators produces the same rate of above average solutions, although they both produce more than 65% of such solutions. On the other hand, the Two-Point and the Swap operators produce a higher rate of children better than their parents.

Figure A-8, shows how the performance measures for the three operators vary with the density of the problems. We can observe that, similarly to the situation described in section A.2, fitpar and domin degrade with the increase of the density, for all operators, although it is more visible for the Two-Point and Swap operators than for the Union operator. However, the differences between the performance of the three operators attenuate when the density increases. In fact, for the problems with the lowest density (the first category of density), the performance of Swap operator is much higher than that of the
Union operator (for fitpar and domin). For problems with higher density (the third category of density), the differences between the operators are visibly smaller. However, fitmed is not very sensible to the variation in the density of the problems.

![Boxplots for each performance measure according to categories of density](image)

**Figure A-8:** Boxplots for each performance measure according to categories of density

These conclusions are confirmed when we replace the three performance measures (fitpar, fitmed and domin) by pmd_oper, the performance measure obtained through the Principal Components method (see Figure A-9).

In order to test if the density of the problems and the crossover operators significantly justify the variations in the performance, we have performed an analysis of variance using the ANOVA model with two factors (crossover operators and categories of density), which results are presented in Figure A-10. In fact, for a significance level of 5%, we conclude that the performance of the operators (summarised in the pmd_oper) variable is influenced by each of the factors separately and also by both of them conjointly.

### A.3.4 Conclusions

In this section we have compared the performance of the three crossover operators. After a Reliability Analysis, we have applied the Principal Components method which has extracted a single performance measure named pmd_oper. We have concluded that the problem density is a factor with a considerable influence in the performance of all the operators, degrading when the density increases, but the differences between them diminish for the problems with the higher density. However, we must notice that all problems in the data set can be considered of low density, so these conclusions cannot be extrapolated for high density problems.

The univariate analysis with two factors (the crossover operators and the categories for density) have shown that these two factors affect the performance, either considered isolated or together.
A.3. COMPARING THE CROSSOVER OPERATORS

Figure A-9: Boxplot and mean plot for the pmd_oper variable

Tests of Between-Subjects Effects

<table>
<thead>
<tr>
<th>Source</th>
<th>Type III Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPER</td>
<td>Hypothesis</td>
<td>82,785</td>
<td>2</td>
<td>41,392</td>
<td>9,412</td>
</tr>
<tr>
<td>NDENS</td>
<td>Hypothesis</td>
<td>155,204</td>
<td>2</td>
<td>77,602</td>
<td>17,630</td>
</tr>
<tr>
<td>OPER * NDENS</td>
<td>Hypothesis</td>
<td>17,629</td>
<td>4</td>
<td>4,407</td>
<td>13,486</td>
</tr>
</tbody>
</table>

Figure A-10: Test for the effect of the operators and density on performance (significant under 5%)

Finally, we can state all crossover operators that we have implemented have proved to be effective in improving the quality of the solutions. Two-Point and Swap operators consistently outperform the Union operator in producing children better than their parents, while the Union operator has proved to be quite effective in producing above average solutions. We think that all three operators play important and complementary roles in the search process, being the Two-Point and Swap operators more effective in intensification stages and the Union operator crucial in slowing down selection pressure.
Bibliography


Index

allele, 122
Average linkage method, 164

BDSP, 45
feasible solution, 95
solution, 95
BERTA, 42
Boltzmann
selection, 173
Boltzmann Tournament Replacement, 173
Boltzmann trial, 173
branch-and-bound, 51, 53
break
length, 25
building blocks, 122, 142
BUSMAN II, 41

CARMEN, 55
centroid, 164
chromosome, 80, 122
cluster, 164
clustering, 86, 161, 164
coding scheme
duties based, 124, 127
pieces-of-work based, 124, 128
column generation, 52, 55
COMPACS, 41, 47
constraint
hard, 90, 101
hard covering, 90, 102
hard partitioning, 90, 101
soft, 90, 104
Constraint method, 79
constraint programming, 57
convex set, 77
Crew
Pairing Problem, 62
Planning Problem, 60, 61
Rostering, 62
crew roster, 61
Crew-Opt, 54, 56
Crews, 42
crossover, 81, 138
performance, 175
Swap, 145, 178, 225, 233
swap, 144
Two-Point, 178, 225, 233
two-point, 141
Union, 178, 225, 231
union, 139, 227
CRU-SCHED, 53
day period, 13
dead running, 14
deceptive problem, 172
decision space, 76
decision support system, 3, 32
decision vector, 76
distance
  in decision space, 163
  in objective space, 163
split, 181
diversification, 146, 155, 172
dominance sharing, 169
driver scheduling, 15, 29
duplicates, 138, 174
Duties Framework, 37, 188, 192
duty, 20, 94
  break, 23
  feasible, 22, 29, 94
  interruption, 23
length, 25
long split, 24
mealbreak, 23
night, 25
shift, 23
split, 24
straight, 24, 96
stretch, 23
two-split, 24
eligible duty, 134
elitism, 84, 154
entropy, 175, 180
  rate, 175
  threshold, 176
epistasis, 123
Evolutionary strategies, 69
EXPRESS, 54
external set, 86, 161

fitness, 80
fitness assignment, 81

fitness sharing, 83
fuzzy sets, 70
gene, 122
generate-and-select, 52, 55, 65, 72
generation gap, 154
genetic algorithms, 64, 80
  hybrid, 64
  multiobjective, 6, 68, 83

genetic drift, 83
GenT, 188
GIST, 3, 32, 33, 52, 68, 188, 191
goal programming, 78
GOAL Systems, 42
GRASP, 72
hard rules, 22
HASTUS, 39, 52, 54, 56
HOT II, 40, 47
IMPACS, 41, 52
implicit enumeration, 51
implicit parallelism, 122
integer linear programming, 49, 51
leftover, 5, 30, 53, 68, 92, 94
  problem, 95
line, 14
linear ranking, 157
  lexical, 158, 167
meal duration, 25
metaheuristics, 63
MICROBUS, 41
MTRAM, 41, 59, 61
mutation, 81, 146
  destroy, 147
INDEX

destroying, 175
Improve, 149
Reduce-Bus-Changes, 153
Reduce-Empty, 152
repairing, 175

niche radius, 83
node, 13
non-dominated
set, 163
solution, 163
NP-hard, 51
NPGA, 85
NSGA, 86

objective space, 76
objective vector, 76
offer structure, 97
operator
repair, 132
overall rules, 28
overcover, 53, 65, 69, 70, 73, 90, 92, 101, 102, 130, 132, 134

pairing, 61
Pareto
dominance, 76, 163
optimal front, 76
optimal set, 76
optimal solution, 76
path, 14
piece-of-work, 21
PinGuim, 48
population, 80
population density, 167, 168
population heuristics, 64

population replacement, 82
binary tournament, 154
Boltzmann tournament, 155
generation, 154
steady state, 154
process
operational planning, 2, 12
rostering, 15
relaxed set partitioning, 93
relief opportunity, 21, 95
relief point, 20, 95
Rotation Group, 99
rotation group, 99
route, 14
RUCUS II, 46
running board, 14
schema, 122
length, 122
order, 122
Schema Theorem, 122
segment, 13
selection, 82
Binary tournament, 86, 172
binary tournament, 160
Boltzmann, 160, 172
lexicographic, 166
Roulette wheel, 159, 172
Tournament, 160
Sequential optimisation, 80
set covering, 50
set partitioning, 50, 92
sharing
fitness, 83, 85
genotypic, 83
phenotypic, 83
Simulated Annealing, 155, 160, 173
soft rules, 22
solution
    non-dominated, 76
SPEA, 86, 160, 161, 167
spell, 23
split duty, 96
spreadover, 96
strength, 86
stretch
    length, 25
Tabu Search, 69, 71
timetable, 14
total sharing, 170
total spreadover, 25, 27
total work spreadover, 27
total working time, 27
TRACS, 47
TRACS II, 41, 52, 56
TRAPEZE, 42
twins, 161, 174
Two-Point, 231
unfitness function, 132

VEGA, 84
vehicle and crew scheduling, 60
vehicle scheduling, 14

walking time, 23
weighted sum, 78
weighting based approach, 84