Alma Mater Studiorum Università di Bologna

Dottorato di Ricerca in
Automatica e Ricerca Operativa

MAT/09

XXI Ciclo

Integrating Crew Scheduling and Rostering Problems

Victor Andres Vera Valdes

Il Coordinatore
Prof. Claudio Melchiorri

Il Relatore
Prof. Paolo Toth

Esame finale 2010

Logic will get you from A to B. Imagination will take you everywhere.
(A. Einstein)

Para Pamela, Renata y Benjamin.

Contents

List of Figures

List of Tables

Abstract

Crew scheduling and crew rostering are similar and related problems which can be solved by similar procedures. So far, the existing solution methods usually create a model for each one of these problems (scheduling and rostering), and when they are solved together in some cases an interaction between models is considered in order to obtain a better solution.

A single set covering model to solve simultaneously both problems is presented here, where the total quantity of drivers needed is directly considered and optimized. This integration allows to optimize all of the depots at the same time, while traditional approaches needed to work depot by depot, and also it allows to see and manage the relationship between scheduling and rostering, which was known in some degree but usually not easy to quantify as this model permits.

Recent research in the area of crew scheduling and rostering has stated that one of the current challenges to be achieved is to determine a schedule where crew fatigue, which depends mainly on the quality of the rosters created, is reduced. In this approach rosters are constructed in such way that stable working hours are used in every week of work, and a change to a different shift is done only using free days in between to make easier the adaptation to the new working hours.

Computational results for real-world-based instances are presented. Instances are geographically diverse to test the performance of the procedures and the model in different scenarios.

Chapter 1

Introduction

For any public transportation company drivers are an important part of the cost and therefore it is essential to achieve the best possible efficiency on their working schedules. Because of this, crew scheduling and rostering problems have been subject of research for a long time. For example, some works in crew scheduling date as back as 1960's (for example Wren [27] and Elias [11]), while crew rostering is a little bit newer in literature (for example Nicoletti [20]).

One of the basic characteristics of these problems is if they are solved for a schedule of services that is repeated every day or if it is for more sporadic services (not repeated every day but with other frequency or just not repeated at all). The case treated here is the first one that is also the case where most of the research has been done.

Considering both problems (scheduling and rostering), crew scheduling is the main topic studied in the literature, because it is where more of the cost reduction can be achieved and there are many well developed models and procedures to solve it. Crew rostering instead is more related to aspects like quality of life than to costs. Nevertheless, the evolution of the research has changed towards methods that integrate both problems in order to obtain a better general result, usually the reduction of the quantity of total drivers, and also because the reduction of the quantity of drivers is not a main concern anymore by itself (because there are many well developed procedures already), but other aspects more related to quality of life have become more and more important. To integrate both problems to improve the solution is widely accepted, but not particularly demonstrated. It is just obvious to

think that the solution of the crew scheduling problem influences the solution of the crew rostering problem because it is its input (biggest influence could not be found), but how to measure this influence is the big question. In almost all the cases this integration is some type of exchange of information between both problems, using some kind of measure about when there may be a better solution. A direct measure of the effect could be much better. One of the two main objectives of this work was to achieve a new integrated model to solve both problems simultaneously, which means to consider the effect of the rostering directly.

Even if these problems have been studied for a long time, there are some aspects barely studied yet. One of them is regularity of the working hours and resting periods. It is hard to find any approach that considers these elements and most of the research where regularity is included is related to airlines, because fatigue is a priority due to safety reasons and it comes mainly from non-regular working schedules. The problem is that the nature of the airlines services leads to models and solution procedures which are quite specific for every particular situation. This work tries to be also an answer to those needs in a more general way and it is its second main objective.

Because of the nature of the public transport it is quite hard to find any kind of regularity in it if we compare it with traditional works, where working hours are well established. Nevertheless it is absolutely important to consider it because regularity is a key element for having a good rest and working in the best way, not only because of safety reasons, but also because fatigue is an issue that affects many different aspects of a person's life.

Because of this, in this approach not only those elements were considered, but they were the basic construction blocks for everything. In fact, this whole approach was constructed backwards, starting from what was needed as result till to find how to do it with what is known. Moreover, the model obtained is the most traditional model used to solve these problems (a set covering problem model), but integrating both problems in just one model. Many procedures have been proposed in the literature, but most of them are centered in obtaining the best mathematical solution being blind about the quality in the terms that are a main issue here.

Differences in labor practices among modes of transport and companies usually constrain the possibilities for applying a model to a wider set of cases.

The approach presented here was developed trying to obtain a procedure as general as possible. All the issues that may change from one case to another (instances) were isolated in special functions, in order to, if needed, only change those functions without change the general procedure. Nonetheless there is in some degree a fixed structure because it is necessary to obtain the desired objectives.

Crew planning may be seen from the planning stage viewpoint, when the total number of crews and crews distributions among depots have to be determined yet, or from the operating stage when the number of crews by depot is known as input data. This approach considers the problem from the planning stage viewpoint.

The contents will be presented as follows: Chapter 2 gives the basic definitions and concepts that will be needed to fully understand the following chapters. Chapter 3 describes the crew scheduling and rostering problems giving also some information about the general scheme in which they are inserted. Chapter 4 shows a brief analysis of the literature mainly focused in showing how integration has been treated and why regularity is so important. Chapter 5 describes the model paying special attention in those elements that allow to have a single set covering problem model representing both problems. Chapter 6 explain the solution procedure used. Chapter 7 shows the results obtained and an analysis about many of their features. Conclusions and some ideas about future work are presented in Chapter 8.

Chapter 2

Concepts and Definitions

Scheduling and rostering terminology differs widely between countries, organizations or even within a single company. The main concepts and definitions as they will be used in this document are presented here. 'Buses' and 'drivers' will be used as terms to represent the ideas because this work was born in a long-distance buses context, but certainly the definitions do not change if they are applied to a different kind of transport.

1. Service: It is the total journey a bus does from the first station of its itinerary till the last one. It may have intermediate stops where passengers can get on and off the bus and where drivers may be changed too. Stops for passengers may not be the same points as those ones where a driver can be replaced. A service is composed of segments and has five attributes: service number, departure station, arrival station, departure time and arrival time. Eventually, a service may have associated some special equipment too, but that case will be no treated here.

2. Segment: It is a piece of a service that can be defined from a passenger′s perspective (places where a passenger may get on or off the bus) or from a driver′s perspective (where a driver change may be done). In this case, the second perspective is used. A segment may have some stops within it, but that is not relevant for the solution procedure if a driver may not be changed or if he cannot take a rest in that point. A segment has five attributes just as the whole service: service number, departure station, arrival station, departure time and arrival time.

3. Pairing: It is a set of segments performed by a driver that starts and finishes in the same depot. It is said that a pairing *belong to* a depot A if it starts and finishes there. There are some rules every pairing has to satisfy, like resting time after some driving.

4. Depot: It is a base to which a driver can be associated. A depot may have more than one station where buses arrive and depart. In this case, a pairing is considered valid if the driver returns to the same depot, but not necessarily to the same station where he started. Connection of different services within a depot can be used if they respect a minimal connection time needed for a driver to move from one station to the other.

5. Shift: It is a driver's work load for one day. It may be formed by one or more pairings, or even just a part of a pairing if it is the case of a two (or more) days pairing. In the literature it is also known as *duty*.

6. Roster: It is a work schedule for a particular driver for several days that is repeated in a cyclical way when it finishes. It is composed by pairings and shifts and it is also controlled by certain rules and regulations. Further on another element will be called roster too, but for now this is the idea that will be used.

7. Spell: A spell is a sequence of segments with no legal resting time due driving taken within its execution. A spell may include segments from different services, and it may contain pauses, but they are shorter than the pause needed to declare a resting time due driving as taken.

8. Time Block: Correspond to time periods of the day. For example, it is possible to say that the hours from the 7:00 till the 17:00 hours are the *morning time block*.

Chapter 3

Crew Scheduling and Rostering

3.1 Transportation Problems

It is possible to say that the main problems in public transport are four:

1. Timetable
2. Rolling Stock
3. Crew Scheduling
4. Crew Rostering

Even if the first two problems are not included in this approach, it necessary a brief view of them because they define the input for the crew scheduling and rostering problems and the future of the optimization is on reaching a better integration of the different stages of the total problem. To develop the timetable considering the rolling stock problem also gives better results, and it is evident that the timetable is a main driver to define the total quantity of drivers that will be needed for scheduling and rostering. Just as the integration of scheduling and rostering, this last kind of integration (timetable + rolling stock) has been barely studied so far.

The Timetable Problem

Behind a timetable there are several different issues related which reveals it complexity. Some of them depends on the type of transport (for example,

for trains the management of the access railways to the station have to be considered to define the feasibility of the solution) while others are more general (for example, it is always needed to study the demand and include aspects related to policies about frequency). Connection among services and safety reasons are also part of the data that need to be considered, and what to do with those empty transports that already finished their services has to be considered too.

Usually the process of obtaining a timetable begins with a definition of what is needed without considering the complete feasibility of the plan. Aspects such as the demand among couples of stations, distribution of the demand along a day and among days, estimations of costs and earnings, estimations of the quantity of machines and workers needed, existing agreements or contracts, marketing or commercial plans are considered. This ideal timetable has to be corrected considering feasibility, trying to let it as close as possible to the ideal one. In any case, the ideal timetable may be a set of timetables which all satisfy what is wanted.

Different approaches are needed if the complete system or at least one part is under design and a situation where the structure is basically fixed or there are constrained resources. One example of this last situation is when platforms in railways applications need to be assigned. Within this case it is possible to find different forms of the problem. For example, if platforms are required for more than one company it is important to consider issues about the competition among them.

There are also characteristics that are desirable in a timetable, where one of the most considered is the robustness. A timetable is robust if services are barely affected when an abnormality appears. To consider possible disruptions when the timetable is constructed permits to reduce the adverse effect of them and also to find a solution to return the system to the normality in an easier way. An example of a timetable constructed in this way is given in Odijka et al.[21].

A timetable not only indicate the services to be done, but also may indicate the technical requirements of every service. This technical requirements may be about the crew and other personal needed to perform the service, during its whole duration (like the crew) or only a part (like a mechanic before the departure), or may be about the machines needed, directly (for example minimal capacity by class in train applications) or indirectly (ma-

chines needed for example before the departure).

The Rolling Stock Problem

This problem is about having in the right place and in the right moment the machines (buses, trains) required by the timetable.

This problem is often found in train services because of the complexity of the possible combinations to satisfy a service, but in other areas we can also find similar situations (for example, buses of different capacity). This problem not only imply just satisfy some minimal requirements of every service, but also how to move the different units in order to have them where is needed in the right moment, including activities such as programed maintenance. In railways application, if a convoy (a set of a least one locomotive and passenger cars) has to be divided in a certain station, the problem also has to consider the availability of resources in that station.

Normally a timetable is changed a couple of times during a year and then a solution for the rolling stock problem is done over this horizon. Some examples of this problem in the literature are [5] and [18].

3.2 Crew Scheduling and Rostering Problems

Crew scheduling and crew rostering are considered separated but related problems because there was not any model easy to apply that could permit consider them as one. As will be shown there is in the literature just one model that optimize both things at the same time, and a second one that is near to this idea of integration.

Both problems can be described as the construction of work programs for every driver or crew, but each one has a different scope. Usually crew scheduling is understood as the problem that tries to find a set of *shifts* or *duties* of *minimal cost* that covers all the services. This set of shifts forms the short term working schedule or daily working schedule. The middle term working schedule is the problem solved in the crew rostering problem, and it is what defines the final quantity of drivers. However, the set of shifts obtained in the crew scheduling problem also influences the final quantity of drivers, so there has to be a good balance between them. What is understood

as *minimal cost* depends on each case. Usually the reduction of the quantity of drivers is the objective searched.

As definition of the crew scheduling problem it is possible to consider the following, given by Freling et al. [15]:

> *Given a set of tasks with fixed starting and ending times and locations, and given a set of rules and criteria, find the minimum cost set of duties such that each task is included in a duty and all rules are satisfied*

In this case the term duty is used as synonym of shift, and task is synonym of segment. Some definitions or approaches consider this problem as the *minimum cost set of pairings* which is not exactly equivalent. In this section it will make no difference which definition was used, because what is relevant are the methods used to solve it, but this approach consider the definition that use the concept *shifts* as the valid one.

The main input for the crew scheduling problem and as consequence for the crew rostering problem is the timetable of services, which has associated a certain set of units (buses, trains) for each service that may influence the crew scheduling and rostering in case that not any driver may be assigned to any service, for example, because there are severals models of machines and a particular driver does not know how to drive all of them, which will not be considered in this approach.

In more theoretical terms and as Caprara et al. [1] show, both problems can be represented in a graph $G = (V, A)$ that have nodes $j \in V$ for each item (segment or pairing) and arcs $(i, j) \in A$ if and only if element j can be sequenced after i in a feasible manner. Using this, both problems can be formulated as finding a minimal-cost set of circuits or paths of G covering each node at least once (Caprara et al. [1] say *covering each node once*, but it depends on the model used).

They also indicate that the graph can be acyclic if it is related to a urban transport system, and it is not acyclic if there are services at any hour of the day, as this approach considers. In the crew scheduling case a dummy node d is included for each depot to represent it and arcs (d, j) and (j, d) are included if a segment j can be used as first or last element of a pairing starting or ending in this depot. Then a pairing is a circuit that has to begin

and end in the node representing a depot. In the crew rostering problem the graph is not acyclic and no dummy nodes are needed because each depot has its own graph (under a traditional point of view where pairings are selected before create the rosters).

Usually, these two problems (crew scheduling and crew rostering) are represented using a set covering problem (or a set partitioning problem) formulation which is based on the idea of generating pairings (in crew scheduling) or rosters (in crew rostering) beforehand, because to create a linear or integer model to define which pairings has to be constructed has not been done in an efficient manner. Then the existing solutions procedures are based on the idea of generating pairings or rosters using a secondary procedure and select the best subset of them during the optimization procedure. A pricing procedure in some kind of column generation approach is usually used to generate more pairings or rosters to improve a current solution, till no improvement can be obtained.

A representation using the set partition formulation (presented by Caprara et al. [1]) is as follows: Let $\mathcal{C} = \{C_1, ..., C_n\}$ represent the collection of all simple circuits of G corresponding to a feasible pairing or roster, with $n = |\mathcal{C}|$ and each circuit C_j having a cost c_j. I_j is the set of nodes covered by the circuit j. A binary variable x_j is defined for every circuit C_j and takes the value 1 if the circuit is included in the solution and 0 if not. Then, the following model can be defined:

$$min \sum_{j=1}^{n} c_j x_j \tag{3.1}$$

s.t.

$$\sum_{j:v \in I_j} x_j = 1, \qquad v \in V \backslash D \tag{3.2}$$

$$\sum_{j \in S} x_j \geqslant |S| - 1, \qquad S \in \mathcal{S} \tag{3.3}$$

$$x_j \in \{0, 1\}, \qquad j = 1, ..., n \tag{3.4}$$

$\mathcal{S}$ represent the set of all inclusion-minimal sets $S \in \{1, ..., n\}$ with the characteristic that no feasible solution contains all circuits C_j for $j \in S$. Constraints (3.2) indicates that every node has to be covered once, and constraints (3.3) are the *side constraints* that control issues related to a particular base. For example, if the problem has a defined quantity of drivers on every depot and the solution has to respect this, then using side constraints would be possible to do it. Side constraints may also control other issues, like the maximum number of night rests outside the depot or the quantity of two-days pairings chosen.

It is important to notice that the model minimize the quantity of pairings and/or rosters, but this not mean the minimization of drivers. The relationship between the two problems is a case by case situation, specially because the construction of the rosters usually is done using some kind of structure, which is particular to every application. To use, for example, a 'week of work' as a unitary base to construct rosters is a common practice and this may define a non linear relationship between the two problems. For example, in Caprara et al. [1] a 'week' of 6 days is considered, and every solution of the crew rostering problem is constructed over this block, and then, the solution is always a multiple of 6, which gives to the problem a certain non-linear characteristic. Then, the minimization of the shifts in the crew scheduling problem cannot assure the best rostering solution and neither the best general solution. It is just intuitively close enough to be useful.

If the cost of overcovering (assign two crews to a single node) is meaningless, it is possible to change constraints (3.2) for:

$$\sum_{j:v\in I_j} x_j \leqslant 1, \qquad v \in V\backslash D \tag{3.5}$$

which are the constraints for the set covering problem formulation. The objective function in this case may need a variation, because in case of overcovering the cost c_j of one of the two circuits that cover the same segment may not be the same anymore, because that segment can be performed only by one crew and costs are defined considering that the crew performs the whole circuit and not considering the possibility of sharing a segment with another crew. Certainly it could be possible to create also pairings that consider the possibility of overcovering costing differently according what segments in the circuit are considered overcovered and performed by other crew, but this is clearly not practical because of the quantity of possible combinations (only to manage the quantity of possible pairings in the traditional way may be overwhelming). Usually the variation of the cost of the circuits because of overcovering is not considered in the optimization stage when costs represent something more than the pure presence of the pairing.

To allow or not allow overcovering of the segments depend on each situation. Usually when it is a case of airlines crew scheduling/rostering a set partitioning problem is preferred because the high cost involved in having pilots not working but traveling. For trains and buses the set covering formulation is more used, because usually is cheaper to have for some segments extra drivers. How much overcover should be permitted is a relative issue that depends on its cost. If having a driver as a passenger involves no cost (any kind of cost) overcovering is not an issue, but if there is a cost, it has to be balanced with the saving it produces, which is an issue, as far as the author knows, that has not been solved clearly yet. Certainly just the presence of a driver traveling and not working is in some sense a cost, but it is not always easy to establish a value for it (cost and benefits).

The effect that overcovering has in the final quantity of drivers is not clear neither. It is accepted that some overcovering is needed to minimize the quantity of drivers, and it is also accepted that too much overcovering should produce to have more drivers than what is needed, but how much has to be accepted or where (which segments or considering what structure) is not clear.

If overcovering is not allowed and not covering a node is a possible solu-

tion, then (3.2) could be replaced by:

$$\sum_{j:v\in I_j} x_j + u_v = 1, \qquad v \in V\backslash D \tag{3.6}$$

where u_v is a binary variable that takes the value 1 if there is no circuit chosen to cover that node, and 0 if a circuit do cover the node. In this case it is also usual to consider a cost for not covering a node. The objective function in that case would be:

$$min \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{|V\backslash D|} p_i u_i \tag{3.7}$$

In theory all the possible sequences of pairings (or rosters) have to be created beforehand, but even for medium size problems the quantity of possible sequences is so big that is not practical to create them all to apply the model, and certainly most of them are not even feasible considering the rules that govern how drivers have to work. It is for this reason that column generation procedures are usually used to solve this problem, creating just as many pairings or rosters as needed. Moreover, thinking in terms of the quality of pairings (or rosters) to create all the possible combinations is not the best approach at all, unless there is a tight control on the optimization procedure in order to prefer those of *better quality*. What it is consider as *better quality* is a subjective matter that depends on each implementation. For example, Ernst et al. [12] indicate that *the quality of a roster is measured in terms of its operating cost and how well it satisfies the personal preferences of the staff. The various objectives in the latter category are usually referred to as quality of life objectives.* The criteria applied here will be explained further on.

3.2.1 The Rules

Rules depends on every particular case and may come from national laws and may be also a company's specific rules. It is also possible to have hard rules that always have to be respected and soft rules which can be broken in some cases under certain conditions. Some of these rules are valid for pairings or shifts and other for rosters, depending on what those rules pretend to

control (daily work a longer period). In any case there are some of them quite common to find (with different values but representing the same idea). These common rules are:

1. Continuous Driving Time: This basic rule is about how many minutes a driver can drive (or work) without having a rest. Because the quantity of resting minutes usually depends on the quantity of minutes driven, it is possible to have some pauses within this "continuous driving time" but they are not long enough to be considered as resting periods due driving. If the time considered for the pause is the driving time or the working time (they may differ for a specific case) depends on the particular rules every situation has. For example, some kind of preparation before a trip can be considered as working time, but for the pauses it only may count the real driving time.

2. Resting Time After a Driving Period: After every driving period a rest must be assigned. How long this resting time is usually depends on how many minutes the driving period had. For example, a usual form is to say that the rest will be a certain percentage of the minutes driven. Only after this resting time has been taken a new driving period may be assigned.

3. Weekend Days: To consider some kind of weekend is usually necessary. It is also frequent to work with a 'week of work' that last a certain quantity of days and weekend days have a direct relationship with it. Sometimes the "weekend days" (that are not precisely a Saturday and Sunday) are included within the week of work, but sometimes is like this approach where a weekend is a bridge between one week of work and the next one. The quantity of weekend days usually is defined according to some rules related to the working time performed during the week of work. For example, it may depended on the quantity of worked hours or on the hours of the day it was performed.

4. Return to the Depot: Drivers must return to their depot on weekends, and also if possible every day. In Europe it is a normal rule to return them every day and limit the quantity of days they have to rest outside their depot, but in other countries this is not true.

5. Maximum Quantity of Driving Hours by Day: For safety reasons the daily quantity of minutes a driver can work or drive is limited.

6. Maximum Quantity of Total Hours by Day: In other words it is the maximum quantity of minutes a daily shift may have.

7. Maximum Monthly Working or Total Hours: A basic rule on every country is about how many hours a person can work during a month as maximum. It may be the case that drivers are subject to a different quantity, lower than the normal one due safety reasons.

8. Minimum Daily Rest: There has to be a minimal quantity of consecutive resting minutes every day to allow sleeping time.

9. Lunch Pauses: Usually some time needed for eating is considered. It may be also fused with the resting time due driving time.

The nature of the rules has a deep effect on the efficiency that is possible to obtain to create shifts or pairings. This means that is of extreme importance to develop a procedure to create shifts or pairings that could be not only fast but also easily implemented and modified.

3.3 Crew Scheduling and Rostering in the Literature

It is not the intention of this section to make an extensive review of the literature but to show in a rough way the diversity possible to find in it plus give some ideas about the procedures used to solve them. To have a wider and deeper idea about the procedures used any of the several reviews that have been done on these topics lately, not only on transportation areas, would be a valuable guide. Some recent examples of these documents are are Ernst et al. [13] (a bibliography) and [14] (where not only transport is considered but also other areas and there is a special focus on rostering) , Kohl and Karisch [17] (for airlines rostering), and an old example can be founded in Raff [22]. Several conferences and dedicated books have been done too. Some examples are [6], [7], [10] and [26].

Crew scheduling is the problem that has been mainly studied in the literature because it is said that is where most of the cost can be reduced, and there are many situations where the rostering is not so relevant. For example, in urban applications usually only crew scheduling is considered. If distances are long or there are night services, then rostering becomes a more important issue.

Some of the oldest approaches of crew scheduling can be traced as back as 1960's (for example, Wren [27] and Elias [11]), but due the scarce technological development, even if it was possible to formulate these problems in mathematical programming terms, to solve them it was only possible to apply heuristics, many times based on the same manual methods used then. Only in the late 70's was possible to use mathematical programming formulations to solve them, but always with some heuristic help and within limited sizes.

Since then, many different methods has been used to solve the problem and many of them applied together, being the set covering problem formulation (that can be traced as back as 1970's like in Rubin [23]) the most popular when buses or trains are the subject of study. This representation has been used not only in exact methods, but also in metaheuristics. In that last case, what it is normally used is a procedure to solve the set covering problem model and not a special development for the crew scheduling problem.

The set partitioning problem has also been used, specially when is an airlines problem because of the high cost of having a overcovering situation. In this case it is allowed leave uncovered some segments because special crews are used to cover them.

To show the diversity of approaches used it is possible to cite the following implementations:

Caprara et al. [2] used the set covering problem formulation to solve it, proposing a method where constraint logic programming and Lagrangian relaxations were used.

Silva [8] worked in a bus driver scheduling problem that tried to minimize the quantity of shifts for a day. He used the set covering problem in a approach that considered constraint logical programming (CLP) within a column generation approach. CLP is used to construct the new columns within an iterative procedure.

An example where metaheuristics (tabu search and genetic algorithms) were used to solve a crew scheduling problem with multiple objective functions is given in Loureno et al. [19].

Cavique et al. [4] used two different kinds of tabu search procedures to solve the problem in an application for the Lisbon Underground. In this case the objective was to find the minimal quantity of shifts that could cover all the services given some contractual rules.

A interactive multi-objective genetic algorithm was used by Pinho de Sousa et al. [9] to solve the crew scheduling problem.

One of the few cases where a wide selection of instances was used is in Wren et al. [28] where is presented the TRACS II system, which is constructed and used by diverse companies in different countries and was designed precisely to obtain a more general system.

Crew rostering seems to be a newer problem in the literature. It can be traced as back as the 1970's (for example Nicoletti [20]).

Most of the papers that only consider crew rostering are related to airlines, because usually services are not so frequent and also many of them are so long that at the end to create a program for a single day does not have much sense. But rostering in this conditions is not the rostering that matter for this approach. The most interesting part of the literature for the purpose of this work will be presented in the next chapter, when integration will be analyzed.

Chapter 4

Integration and Regularity

4.1 Integration so far

Before talk about integration, it is necessary to talk about how these two problems could be solved without it in just one procedure. A decomposition approach was usually considered, dividing the problem in the following stages:

1. Pairing Generation
2. Crew Scheduling
3. Crew Rostering

Nowadays these stages are defined as interactive modules that may be applied dynamically in order to improve the solution, but in old procedures these three stages used to be applied in sequence.

How to do the optimization of these problems in sequence was more or less like this: First, there is a generation of a limited quantity of feasible pairings because to generate all of them is not only overwhelming but not practical at all, because of the low quantity that is finally used. The main issue was (and is even today) what kind of criteria could be applied to generate precisely the set of pairings that could give the smallest quantity of drivers in the following step. Once this set was generated, the second step was to select the best subset of them using usually a set covering formulation in order to minimize the quantity of pairings that covers all the segments. This was done because it was reasonable to expect that the minimization of

the pairings could give a final low quantity of drivers and overcovering was also controlled because to have too many segments overcovered could lead to more drivers than needed. Nevertheless a minimal quantity of pairings and minimal overcovering are not synonyms of minimal quantity of drivers because there is no a direct relation between this objective and those characteristics of the solution but they depend on the particular sequencing done. It is the sequencing that can be done the one that dictates if that set can o cannot achieve the lowest quantity of drivers. Once the solution for the set covering was ready, the next step was to solve the crew rostering problem. From the selected set of pairings several rosters had to be constructed, sequencing the pairings in feasible ways. A couple of pairings can be sequenced immediately (considering the resting time needed) or using a daily break between them. A long chain of pairings sequenced becomes a roster. All these sequences of pairings (rosters) can be represented just as pairings were represented, using columns in a set covering problem formulation that will decide the final subset of rosters and then gives the minimal quantity of drivers given the limitations considered.

Using this old idea is possible that within a roster two pairings are sequenced in a same day, forming a shift. The natural question that arises is which is the difference with the current approach if even these approaches may consider the creation of shifts. The difference with the approach presented here is that in this old fashion the selection of the pairings is done before analyze if they can be sequenced in an efficient manner. To work with shifts and not pairings from the beginning permits to control the effect of the sequencing the whole time.

More or less 10 years ago integration became almost compulsory to consider if both problems were solved at the same time, but not many times this was done.

A good example of this was given by Caprara et al. in two different papers ([1] and [3]). These authors in [1] applied the same three stages mention before to solve both problems in a sequential fashion, but because of the limitation of this approach (the selection of a set of pairings without consider its possible effects in the rostering phase) the next step was try to integrate the stages in order to achieve a further improvement. This was done in [3] where pairings are generated previously and the best solution is searched guided by some bounds that give some information about the

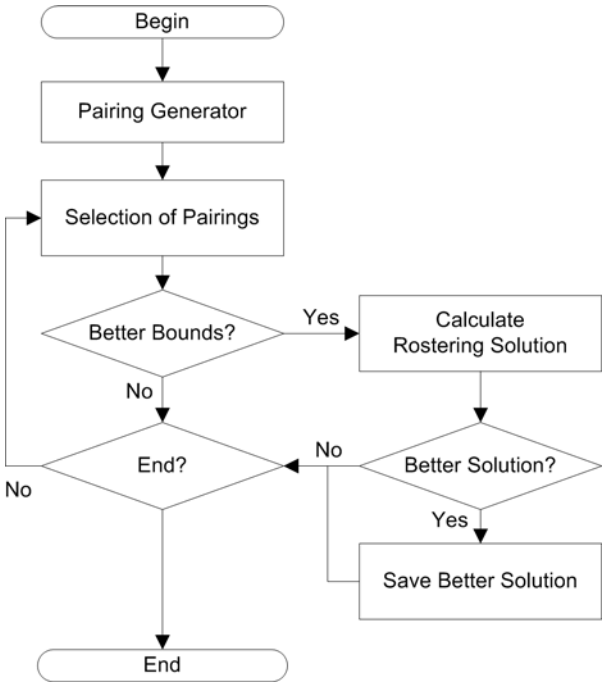

Figure 4.1: Optimization idea Caprara et al. [2001]

chances of obtain a better rostering solution. Their idea was to guide the selection of pairings according to the objective of the rostering phase. They kept the stages indicated before, but the approach works as shown in figure 4.1.

This proposal considers a modification of the cost of the pairing during the optimization procedure, in order to represent better their effect in the general result (minimization of the quantity of drivers).

This case is not integration as intended in this study, but it recognizes and works with the relationship between scheduling and rostering. They indicate as future work the possibility of creating pairings in a dynamic procedure according to what is obtained during the procedure. The approach presented here works with this dynamic generation, plus the possibility of knowing for sure if a set of shifts can improve the general solution.

Another case that was presented as a integrated approach was the one given by Freling et al. [15] where rosters are constructed directly from the segments, which is not a real integration of both problems, but an elimination of the crew scheduling step. They use a Branch and Bound technique with column generation and tested several instances, but they only applied

the 'integrated' approach to a particular case of a airline, where three relatively small instances were tested (196, 244 and 521 segments each one). Results in terms of quality were in some degree better than the not integrated approach, but processing times were much higher (the biggest instance took more than 26 hours to finish).

As far as the author knows, there is just one single case of a single model that optimize at the same time crew scheduling and rostering problems. Ernst et al. [12] present a case in a railway context (National Rail Australia), where a general model (not a set covering or partitioning problem model) was developed. They define the crew scheduling problem as *the construction of duties in such a way that the timetable is covered adequately* where a duty is an element like a pairing (closed) usually long enough to be a shift (a daily assignation). The Australian railway network has very sparse nodes and trips are usually quite long, producing that a driver could not return to his home every day, but after a few. Because of this they may assign to a driver pairings that not always belong to his depot and the solution search is for one complete week.

The objective function is not about quantity of drivers but it is the sum of three concepts: (1) Cost of the selected set of pairings ; (2) Cost of overcovering/undercovering and (3) cost of the *crew complements*. *Crew complements* are sets of pairings or shifts that last one week (which is the planning horizon of the model) and rosters are created assigning these units to the real crews. From a different point of view, each *crew complement* represent a crew in a depot for one week.

In the set of constraints workload for every crew is controlled and these constraints plus the consideration of the overcovering/undercovering is the way this model controls and optimize the rostering. After optimization is done, the minimal cost is founded, but the specific rosters are not constructed. Ernst et al. [12] propose two methods to create cyclic rosters or non-cyclic rosters respectively. These procedures do not change the final quantity of crews because it is defined from the quantity of *crew complements* which is fixed. In terms of results the authors indicate that three instances where tested, having from 242 segments till 1309 segments. The solution had in most of the cases a weekly working load between 2300 and 2700 minutes and the solution procedure never took more than 3 minutes in solving the problem. They also tested instances created to analyze the

performance of the algorithm, all of them with near 1500 segments. Solution times where higher in these cases, but never more than 7 minutes and never less than 2 minutes.

4.2 Regularity in a Regular Work

The construction of the model and the finding of the how to melt both problems was made from the 'end to the beginning', because first was defined how rosters should look and from that idea the research was try to find how to introduce this ideal result in a solution procedure. Existing procedures that consider issues about regularity are usually constructed in a traditional manner, and the solution is corrected using 'penalty functions'. The problem of using penalty functions is that the idea behind the optimization is to obtain a solution as always and correct it according to what it is wanted. An approach that considers these objectives as part of its structure should give better results.

There are many policies and practices about how to construct shifts and rosters for situations where workers are needed all the time in different areas (not only transport). An example that comes from a Chilean refinery (formerly called Petrox S.A.) is in the table 4.1.

Table 4.1: Shift work performed in a regular scheme

Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Shift 08:00 – 16:00	A	A	A	A	A	A	B	B	B	B	B	B	C	C	C	C	C	C	D	D	D	D	D	D
Shift 16:00 – 00:00	C	D	D	D	D	D	D	A	A	A	A	A	A	B	B	B	B	B	B	C	C	C	C	C
Shift 00:00 – 08:00	B	B	B	C	C	C	C	C	C	D	D	D	D	D	D	A	A	A	A	A	A	B	B	B
Free Day	D	C	C	B	B	B	A	D	D	C	C	C	B	A	A	D	D	D	C	B	B	A	A	A

Each letter represents a crew and every day there is one crew covering one of the three 8-hours shifts, plus a crew having a free day which is represented in the row 'free day'. Every crew works 6 days and then have a 'weekend' which long depends on the shift performed during the week (from one to three days).

Certainly a schedule like this is too rigid if it is applied to a transportation context, because there are services to be done at different times of the day and there is no way to fit everything into strict blocks, but it would be ideal to have something similar to this for achieving regularity in a non regular kind of work.

It is known that on a crew scheduling and rostering problems there are millions of possible pairings and usually solution procedures do not consider the possibility of creating regular sleeping hours, when the quantity of pairings may give the idea that is possible depending on how they are chosen. In this approach it will be shown how this can be done.

4.3 Regularity in a Non Regular Work

The importance of regularity may be easily understood by intuition, just because any person has the experience that a little change in the rest pattern or the working pattern has a great significance in how that person feels. This issue has been studied formally and here some of those results are mentioned.

Smith et al. [24] indicate that roster design requires the consideration of three simultaneous goals. Quoting:

- *Good ergonomic design to maximize employee health and safety in and outside the workplace.*
- *Integration of work hours with social and family life, and*
- *Securing flexibility and efficiency in meeting production and service criteria.*

Nevertheless they also indicate that these goals are barely considered in the real design because financial benefits are taken as the main and almost only criterion. They also indicate (from a further reference) which are the best practices to design a roster. Quoting:

1. *No more than three consecutive night shifts.*
2. *At least 10 hours between consecutive shifts and at least a 48-hour break between the end of the last night shift and the start of the first day shift.*
3. *At least one full weekend and some part weekends off each month.*
4. *Individual work hours limits to avoid excess hours work and fatigue.*

5. *No second jobs.*
6. *Travel to or from work (or post-shift sleep facility) no greater than 1 hour.*
7. *Shift swaps allowed as long as within work hours limits and involving no quick night-day changeovers.*
8. *At least two and preferably three short breaks during the shift for meals and recreation.*
9. *Medical screening of shiftworkers.*

Further on, when the model will be detailed, will be more clear how this approach satisfy some of these point, but here it will be mention lightly. It would be recommendable to read again this section after read how the model is constructed and solved, to clarify in a better way the ideas. The first condition is not reached for this approach because all of the shifts are performed for a week of work. The second condition depends on the parameters used, but as it is shown here, it is respected on one day shifts. When there is an external rest (the driver do not sleep on his depot) it may be a little shorter, being at least 8-hours long. The third condition depends on how rosters are constructed, but the design of this approach considers the assignation of full weekends (two days) an even more in many cases, so it should be accomplish in most of the cases. The fourth condition is controlled by the rules applied (maximum continuous driving time). The fifth and sixth conditions belong to a different decision area. The eighth is satisfy because if a worker changes from one shift to another (which in fact is not considered here, but the model has the flexibility of considering it if needed) it should be done to a shift with the same quantity of weekend days, so it should be to a 'closer' shift in terms of working hours. The ninth condition is controlled by the same regulation applied and depend on how many spells a shift has. The last condition is not part of the scope of this approach.

Smith et al. [24] consider to work in shifts as a stressor that have three main sources of stress: sleep and fatigue (chronic and acute), circadian rhythmicity and family and social life. They also indicate that shifts that rotate faster are preferred to those ones with a slow rotation, but it depends on the long of the daily shift. More work contain and longer the shift are the

basic elements to require a fast rotation to avoid stress. Some examples of circadian rhythms are heart rate, motor activity rate and core body temperature. Some of these circadian rhythms are fundamental for alertness which certainly is a main safety issue for a driver (see Weir [25]). The quantity of sleep and also the quality and timing of sleep are important to mitigate fatigue.

Smith et al. [24] indicate the following characteristics of a shift as the main issues that have an impact on sleep and fatigue:

- Shift Start and Ending Times: A shift that starts too early usually induce lost sleep, because most of the people do not start sleep earlier because of this and they do not have a good rest because of the fear of oversleeping. Also, the shift start and ending times establish how easily a person can arrive to work or to his home, adding another stressing component.

- Direction of Rotation: The direction of rotation is the order in which shifts are sequenced to create a roster. For example a sequence Day - Afternoon - Night is a direction of rotation. Particularly this one is called *forward rotation* and it is preferred because its sleep-wake patterns permits a better adaptation to the new working hours.

- Shift Length: Fatigue is clearly influenced by a shifts' length and the time of the day where is performed.

- Number of Consecutive Night Shifts: Smith et al [24] indicate that daytime sleep length is shorter than night sleep length which induce lost sleep during a night shift period and when a series of night shifts begin (the first day) is when more sleep is lost. Then, in case of needing a night shift, in theory the best practice is to make as few changes to night shifts as possible to avoid that 'first day' and also try to keep them short (in terms of quantity of days repeated) to avoid the lost of sleep produce during the day.

- Number of Consecutive Day Shifts: Even if in terms of lost sleep is the shift that has less of it, the quantity of consecutive days has to be an adequate quantity and no longer. In other words, it is about the long of the week of work.

- The Time Off Between Consecutive Shifts and Days Off Between Different Shifts: This time is required not only for sleeping, but also for resting and recreation.
- The Number of Working Hours per Week: More hours induce a lower quality of the work performance.
- The Regularity of the Roster: Regular rosters permit to predict and plan sleep hours and others activities.
- The number of shift crews: It is referred to works where shifts has a common length. Four shifts are the usual quantity and five a quantity that allow flexibility for other activities.

The current approach considers these characteristics in the following way:

- Shift start and ending times: It is the procedure that chooses the shifts to keep and these times comes from there. The only possible way of control them would be to control the optimization procedure in order to choose the inclusion of a shift according to this characteristic when severals shifts may be included because in terms of optimization they do not make a difference. This kind of control was not considered.
- Direction of Rotation: For optimization purposes, it does not matter the direction of rotation, and then the forward direction (Morning - Afternoon - Night) is followed when possible, depending on which shifts are finally included in one roster.
- Shift Length: The maximum long is controlled as a parameter, but it is hard to obtain a good solution when shifts are limited to be short because the natural irregularity founded in a transport context.
- Number of Consecutive Night Shifts: All shifts are performed for the same quantity of days.
- Number of Consecutive Day Shifts: The same comment as the last point is valid. In any case the long of the week of work was establish based on a typical length.

- The Time off Between Consecutive Shifts and Days off Between Different Shifts: The set of rules that defines how many weekends days correspond to a each shift selected is fundamental for the optimization procedure and a key element for the solution founded. The rules applied and presented here take care of this point as a main issue aiming to a better adaptation to the new working hours.

- The Number of Work Hours per Week: The quantity of hours each week depend on the shift performed. There is no control about this issue more than a maximum daily quantity set as a parameter, which in many cases comes from legislation.

- The Regularity of the Roster: Rosters are constructed in such a way that planning is perfectly possible to make. Only mayor disruptions could change this.

- The Number of Shift Crews: It is not relevant for this approach because the quantity of shifts crews is defined by the quantity of shifts included in the roster, but it is important to mention that the construction of the rosters is made based on the idea of a 4-shifts structure (3 shifts covering 8 hours plus a crew having a free day).

Regularity is seldom find in literature as a main issue. One example is Klabjan et al.[16], where an airline crew scheduling problem is solved over a weekly horizon and regularity is search over that period, calling regularity the possibility of repeating shifts during that week. The timetable is weekly, but with many daily flights that have exceptions on weekends. This approach did not considered regularity as a basic construction block of the solution as how is done here.

Chapter 5

The Model

The model and the key elements that allow to have a single set covering problem model optimizing crew scheduling and rostering at the same time will be explained here.

5.1 Key Elements

Two key elements allow to have a single model optimizing scheduling and rostering at the same time. One is how rosters are constructed. The other is to work with shifts instead of pairings. Both things allow to consider directly the quantity of drivers in the objective function.

5.1.1 Working With Shifts

To optimize the quantity of drivers using a direct representation of them in the objective function is necessary to work with shifts and not only pairings. If everything is measured in terms of shifts the quantity of drivers is directly considered because a shift is the work a driver has to cover one specific day. Many traditional approaches work with pairings maybe because it is the first block of construction for a feasible program, but pairings do not represent directly the quantity of drivers. One pairing represents one driver, but we cannot say that a certain quantity of pairings represent the same quantity of drivers, because it may be possible to sequence those pairings in a feasible shift and reduce the final quantity of drivers, which could be smaller than the quantity of pairings.

Moreover a same quantity of pairings may represent different quantity of drivers.

5.1.2 How to Construct a Roster

Before explain how to construct a roster, it is necessary to explain how the long of a weekend after a week of work is defined.

This approach considers the weekends as bridges between one week of work and the following one and not as an integral part of the week as most approaches do. Because of this after a week of work some days have to be assigned as weekends. How many days will be assigned depends on every particular situation, because usually every company or nation has some rules rewarding this issue. Here the following set of rules were considered:

- One day is assigned to all shifts as a base.
- One extra day is assigned to all shifts that begin in hours considered 'not desirable' (too early or too late). The typical limits considered are from 18:00 (included) till 07:00 (excluded).
- One extra day is assigned to all shifts that have working hours during the night. The limits used are from 22:00 to 05:00.

Certainly other rules could be used to define how many weekends days correspond to every shift. This is not particular important because the optimization procedure does not change because of this definition (just a module has to change).

Considering the rules stated, there are three possible lengths of the weekend (1, 2 or 3 days) which will be vital for the optimization process.

Now it will be described how a single roster is constructed for a single driver and how it can be related to others drivers' rosters to form a closed set of shifts/drivers. It will be shown how each one of these sets (and as consequence all the rosters that belong to a depot) can be represented using the associated 'weekend resting days'. Finally it will be shown how this fact allows to have a formula that considers the extra quantity of drivers needed due the necessity of creating rosters and giving weekly resting times.

To construct a roster a basic rule is used: During a week of work a driver has to perform only one shift and a change to a different shift can be done

only using free days (called weekends) in between. If two-days shifts are also allowed this rule needs to be changed a little, but in essence it will be the same idea.

A 'weekend' here can be any day or set of days during the week. It only represents a resting period before a change of shift for a driver. It is not fixed to a particular day of the week because in this way they change during the execution of the roster in a long term and then it is fair for every driver.

To show how a roster is constructed an example will be shown.

If three shifts have to be assigned:

- X: A shift with 1 weekend day.
- Y: A shift with 2 weekend days.
- Z: A shift with 3 weekend days.

A roster for a driver A can be constructed in the following way. First, driver A perform shift X for a week of work and when the week finishes a weekend resting period is to be assigned; in this case one day:

Day	1	2	3	4	5	6	7
Shift X	A	A	A	A	A	A	
Rest							A

After this weekend another week of work can be assigned to driver A. In this case the shift Y is assigned for one week and after it its weekend resting period (two days):

Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Shift X	A	A	A	A	A	A									
Shift Y								A	A	A	A	A	A		
Rest							A							A	A

After this second weekend the third shift is assigned and its weekend:

Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Shift X	A	A	A	A	A	A																		
Shift Y								A	A	A	A	A	A											
Shift Z																A	A	A	A	A	A			
Rest							A							A	A							A	A	A

After finishing with the third shift (and its resting time) this driver can be assigned to the first shift again, having in this way a 24-days roster for him. Now the question is who performs these shifts the rest of these days when driver A is not performing them. Just as it was done with driver A it is possible to make the same assignations for a second driver B, starting from the seventh day, when driver A leaves shift X.

Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Shift X	A	A	A	A	A	A	B	B	B	B	B	B												
Shift Y								A	A	A	A	A	A	B	B	B	B	B	B					
Shift Z	B	B	B													A	A	A	A	A	A	B	B	B
Rest				B	B	B	A						B	A	A					B	B	A	A	A

It can be seen that the day 24th is followed by the day 1. Driver B makes the same roster than driver A, but not the same days.

To complete the assignation and cover all shifts every day we need two more drivers, C and D as shown in the following table:

Table 5.1: Final assignation - Four drivers covering 3 shifts in 24 days

Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Shift X	A	A	A	A	A	A	B	B	B	B	B	B	C	C	C	C	C	C	D	D	D	D	D	D
Shift Y	C	D	D	D	D	D	D	A	A	A	A	A	A	B	B	B	B	B	B	C	C	C	C	C
Shift Z	B	B	B	C	C	C	C	C	C	D	D	D	D	D	D	A	A	A	A	A	A	B	B	B
Rest	D	C	C	B	B	B	A	D	D	C	C	C	B	A	A	D	D	D	C	B	B	A	A	A

This procedure can be always done if just one rule is respected: *The sum of the weekend resting days of the shifts assigned to a driver has to be equal to the long of the week of work.*

For convenience this set of shifts and drivers will be called from now on a roster, even if it includes several rosters (the same one for every driver). A roster for a particular driver will be called *single-roster*.

Two more examples are shown to make clearer this point.

Example 1:

- R: Shift with 1 weekend day.
- S: Shift with 1 weekend day.
- T: Shift with 2 weekend days.
- U: Shift with 2 weekend days.

The resulting roster is:

Table 5.2: Example 1: Final Roster

Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Shift R	A	A	A	A	A	A	B	B	B	B	B	B	C	C	C	C	C	C	D	D	D	D	D	D	E	E	E	E	E	E
Shift S	D	E	E	E	E	E	E	A	A	A	A	A	A	B	B	B	B	B	B	C	C	C	C	C	C	D	D	D	D	D
Shift T	C	C	D	D	D	D	D	D	E	E	E	E	E	E	A	A	A	A	A	A	B	B	B	B	B	B	C	C	C	C
Shift U	B	B	B	B	C	C	C	C	C	C	D	D	D	D	D	D	E	E	E	E	E	E	A	A	A	A	A	A	B	B
Rest	E	D	C	C	B	B	A	E	D	D	C	C	B	A	E	E	D	D	C	B	A	A	E	E	D	C	B	B	A	A

Example 2:

- M: Shift with 1 weekend day.
- N: Shift with 1 weekend day.
- L: Shift with 1 weekend day.
- P: Shift with 3 weekend days.

The resulting roster is the following one:

Table 5.3: Roster 2: Final Roster

Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Shift M	A	A	A	A	A	A	B	B	B	B	B	B	C	C	C	C	C	C	D	D	D	D	D	D	E	E	E	E	E	E
Shift N	D	E	E	E	E	E	E	A	A	A	A	A	A	B	B	B	B	B	B	C	C	C	C	C	C	D	D	D	D	D
Shift L	C	C	D	D	D	D	D	D	E	E	E	E	E	E	A	A	A	A	A	A	B	B	B	B	B	B	C	C	C	C
Shift P	B	B	B	C	C	C	C	C	C	D	D	D	D	D	D	E	E	E	E	E	E	A	A	A	A	A	A	B	B	B
Rest	E	D	C	B	B	B	A	E	D	C	C	C	B	A	E	D	D	D	C	B	A	E	E	E	D	C	B	A	A	A

A 6-days week of work has been defined by convenience (this long has a key function in the optimization procedure), but this rule is valid considering any possible long of the week of work.

It will be shown that in the optimization procedure it is possible to obtain rosters that do not reach the 6-days adding their respective weekend days. This is not a problem because there two possible solutions as it will be shown.

Following the rule announced rosters result with different lengths. The long of the roster depend on the quantity of shifts included. Because of every shift is performed for a week of work and the total quantity of resting days is equal to the week of work too the following formula gives the long of the roster in days:

$$roster\_long = (n\_shifts + 1) \times long\_week \tag{5.1}$$

where:

- $roster\_long$: Quantity of days the roster last.
- n_shifts: Number of shifts the roster has.
- $long\_week$: Length of the week of work in days.

Every roster represents one extra driver in the system because of the necessity of giving weekend rests (the extra row shown in the tables under the name 'Rest' is the representation of that extra driver). In fact it is possible to create rosters with more than one 'row' of resting assignments and these rosters could cover more shifts, but that case will be not considered here.

Then, for a particular roster it is possible to state the following:

$$q\_drivers = n\_shifts + 1 \tag{5.2}$$

where $q\_drivers$ is the quantity of drivers the roster needs.

If the first six days of any of the tables shown here are taken it can be seen that during that period every driver takes free days and every resting weekend considered correspond to a different shift.

Considering the first example shown the first six days of the 'Rest' row contain the following information:

Table 5.4: Roster represented by shifts' weekend days

Day	1	2	3	4	5	6
Rest	D	C	C	B	B	B

In this example, D is the rest that comes from the shift X, C are the resting days that comes from the shift Y, and B are the days that comes from the shift Z.

Moreover, if we take any subset of consecutive six days (because six is the long of our working week) the same situation will be present: all of the shifts will be represented through their weekend resting days (independent of the drivers included), and this will be true always because of the structure used to create rosters.

This idea can be used to represents drivers in the objective function.

Let's consider a complete solution for the problem. It will be form by a set of shifts that can be converted into subsets if they are categorized by the depot to which they belong to. If one of these subsets is taken there will be a pool of shifts, each one having a certain quantity of weekend resting days associated according to their working hours or any other characteristic that defines how long the weekend has to be. This fact can be used to represent the same shifts using only their weekend rests and then there will be a pool

of resting days that needs to be associated to a roster. All of the weekend days for a particular shift have to be put together always.

Thinking grossly it is possible to say that if the total quantity of resting days of a depot is divided by the long of the week of work (because the long of the week is also the maximum quantity of resting days a roster can have) it is possible to have a first idea about how many rosters need to be constructed (the minimal quantity).

Previously it was said that to choose a week of work 6-days long was a convenient decision. One reason is it is just a normal quantity of days to work before having a weekend, but there is another reason that is fundamental for this application. From the rules explained before, it is known that there are three values that can be assign as weekend rest: 1, 2, or 3 days. Now let's think that every roster is a container that can receive 6 resting-days and because every 6-days subset of weekend days represents one more driver in the system, the objective is to minimize the quantity of containers used. That problem is the well know bin packing problem (every roster is a container and there are elements of size 1, 2, or 3 to assign), but this situation is a special case of the bin packing problem, fortunately easier to solve. Moreover, it is not not necessary even to solve it to know the result (the quantity of rosters that will be needed) and this is fundamental for the procedure, because it is possible to optimize the quantity of rosters without construct them.

Then, for a particular depot i, the quantity of rosters that will be needed is:

$$Q\_rd_i = \left\lceil \sum_{j=1}^{S_i} \frac{rd_{ij}}{long\_week} \right\rceil \tag{5.3}$$

where:

- S_i : Quantity of shifts of the depot i.
- rd_{ij} : Quantity of resting days associated to the shift j of the depot i.
- $long\_week$: Long of the week of work.

and for the system, the total quantity of rosters is just the sum of this quantity for every depot.

$$Q\_rd = \sum_{i=1}^{D} \left\lceil \sum_{j=1}^{S_i} \frac{rd_{ij}}{long\_week} \right\rceil \tag{5.4}$$

where:

- D : Quantity of depots in the System.

In other words, this last equation represents the extra quantity of drivers the system needs to rotate its drivers and give them free days as weekends.

From equation 5.1.2 it is possible to see that longer the week of work, fewer drivers will be needed to rotate. The only problem of this is that the special case of the bin packing problem that allows to optimize without construct the rosters would not be possible anymore, but rosters have to be constructed to know really how many they are and the real quantity of drivers of the system. In that sense, what would be possible (and maybe better) is to optimize using a week of 6 days as it is presented here, and when there are less efficient depots in terms of the definition of the quantity of drivers for rostering (when the sum of the weekend days of the shifts is not an exact multiple of the week) those extra days could be spread on different rosters, making those weeks longer and reducing in one driver the requirements of that depot.

When rosters are constructed this way, the monthly working hours are variable each month because the schedule is not constructed following a 30-days target. Nevertheless it is possible calculate an average value for every roster. For simplicity it will be called 'monthly hours', even if it is not exact value every month:

$$monthly\_hours_j = \frac{reference\_month \times long\_week \times \sum_{i=1}^{S_j} wt_i}{roster\_long \times 60} \tag{5.5}$$

Where:

- $reference\_month$: 30 days
- wt_i: Daily working time of the shift i
- S_j: To indicate the quantity of shifts included in the roster j

- *roster_long*: As indicated in the equation 5.1
- 60: To change the value from minutes to hours.

5.2 The Model

Considering what has been described so far, the following model is obtained:

$$min \sum_{i=1}^{D} \sum_{j=1}^{S_i} c_{ij}\ x_{ij} + \sum_{i=1}^{D} \left\lceil \sum_{j=1}^{S_i} \frac{rd_{ij}\ x_{ij}}{long\_week} \right\rceil \tag{5.6}$$

Subject to

$$\sum_{i=1}^{D} \sum_{j=1}^{S_i} a_{ij} x_{ij} \geqslant 1 \qquad for\ k = 1, ..., M \tag{5.7}$$

$$x_{ij} \in \{0, 1\} \tag{5.8}$$

where:

- x_{ij}: Binary variable to represent 'shift' j of depot i.
- c_{ij}: Scheduling coefficient, with value 1 for 1-day shifts and 2 for two-days shifts.
- D : Quantity of depots in the system.
- S_i : Quantity of shifts of the depot i.
- rd_{ij} : Quantity of weekend resting days associated to the shift j of the depot i.
- $long\_week$: Long of the week of work.
- a_{ij} : Technical coefficients to represents every shift as a column.
- M: Quantity of segments.

To describe the variable it is used the word 'shift', but it may represent two shifts when it is a two days shift (there is a resting time outside the depot longer than the minimal daily rest), but because those shifts has only sense treated together, what is correct is call them just shift, as the rest of them.

The first part of the objective function represent the quantity of drivers needed to cover the work load, and it would be possible to say that is the crew scheduling part. The second part of the objective function represents the quantity of drivers needed to create rosters, and then it is the crew rostering part.

For optimization purposes here is considered a relaxation of this model as follows:

$$min \sum_{i=1}^{D}\sum_{j=1}^{S_i} c_{ij}\ x_{ij} + \sum_{i=1}^{D}\sum_{j=1}^{S_i} \frac{rd_{ij}\ x_{ij}}{long\_week} \tag{5.9}$$

Subject to

$$\sum_{i=1}^{D}\sum_{j=1}^{S_i} a_{ij}x_{ij} \geqslant 1 \qquad for\ k = 1, ..., M \tag{5.10}$$

$$x_{ij} \in 0, 1 \tag{5.11}$$

This relaxation is called the integer relaxation, because only the condition of the rounding for the rosters is changed. This model can relaxed even more if the integrality conditions are dropped. In that case the equation 5.11 becomes:

$$0 \leq x_{ij} \leq 1 \tag{5.12}$$

This one will be called the linear relaxation.

In both relaxations, coefficients may be reduced to a single value that represents how many drivers the inclusion of that shifts means, and it will be a fractional value, because there is an integer part that comes from working part of the shift in the crew scheduling problem, plus an *extra* that comes from the effect of the rotation (the weekends every shift has to include).

This coefficient is calculated as follows:

$$C_{ij} = c_{ij} + \frac{rd_{ij}}{long\_week} \tag{5.13}$$

Two examples of how coefficients are defined are given:

Example 1:

A one day shift is considered in this example. The first part of the objective function is simple, because for every shift it is necessary a 1 to

count the presence of the driver for one day. The second part, related to the rostering, needs to be calculated but it is not complex. In fact, it is just the quantity of weekend days divided by the long of the week of work. Then, if a shift with two weekend days is taken, the coefficient would be:

$$Coefficient = Scheduling\,Effect + Rostering\,Effect = 1 + \frac{2}{6} = 1\frac{1}{3}$$

Example 2:

This example is about a two-days shift. In this case, every day is calculated by itself and the results are added to have a single coefficient, because the variable represent both shifts. For example, if on day 1 the shift has 2 weekend days and in day 2 the shift has 1 weekend day, it would be:

$$Coefficient\,day\,one = Scheduling\,Effect + Rostering\,Effect = 1 + \frac{2}{6} = 1\frac{1}{3}$$

$$Coefficient\,day\,two = Scheduling\,Effect + Rostering\,Effect = 1 + \frac{1}{6} = 1\frac{1}{6}$$

$$Coefficient = 1\frac{1}{3} + 1\frac{1}{6} = 2\frac{1}{2}$$

As it can be seen from the model, all of the depots are included and as consequence optimized. Most of the traditional approaches consider the rostering optimization phase depot by depot. This has two reasons.

First, because the optimization of the scheduling problem is not integrated with the rostering part in those approaches, a solution for the crew scheduling part defines a set of pairings (of shifts) for every depot, and the rostering optimization has to be done with those shifts and no others. The other reason is that, as some authors indicate (for example Caprara et al. [3]), the traditional procedure followed for most of the companies is to solve the rostering within a regional scope. To integrate all of the depots should give a better general result, because the effect that every depot produces over the others is implicitly considered.

The other important characteristic is the non linearity of the term that represent the rostering part. This is not only an effect of this particular

model, but a characteristic that can be founded in other approaches in some way or another, but not explicit and measured as it can be seen here. Citing again to Caprara et al. [3] in the rostering phase they work with 6-days working weeks, but they include the weekends inside them. Every roster created is obtained from the addition of these weeks, and then, every solution is a multiple of 6, which produce the same non linear effect presented here. To have the possibility of a direct measure of this effect is the tool this approach can give and other approaches do not have to control in a better way an integrated optimization.

Chapter 6

Solution Procedure

The solution procedure is designed considering a basic idea: to gradually sacrifice desirable characteristics of the solution in order to improve the value of the objective function in terms of its linear relaxation and also searching for the best integer solution. The aim is to include in the solution as many *good-quality* shifts as possible. In some sense may be seen as a decisional procedure, because it is possible to improve a current solution taking decisions about what may be sacrificed or changed, but instead of asking on every iteration what will be sacrificed next, the decision is included in the procedure and controlled by parameters. Because of this design it could be interesting to implement it precisely in decisional scheme, allowing to the user to include or exclude shifts or spells with certain characteristics or constructed in certain ways, or to change parameters gradually or by areas, but such implementation is out of the scope of this work.

It is known by researchers that the crew rostering solution depends on the crew scheduling output, and as a result the general solution depends on it. The point is that the crew rostering solution always is done taking as a base a crew scheduling solution and the best solution founded is within these limits, not knowing if it is the best general solution. One simple reason for this is that both problems are represent by independent models, and then, it is impossible to say that the second model has reached its best solution when it is limited by the first one, where there are so many alternative solutions.

Moreover, the real objective of the problem is not linear because the rules of the rostering make it that way, specially the condition of working with weeks of works. Existing approaches of rostering when services are

performed daily usually have to consider some kind of structure to assign rosters, which is usually a week of work, and rosters are constructed adding these 'units'. Then, because of this traditional and logical condition, rosters construction has not a direct (or 'linear') relationship with the crew scheduling problem. Because of this, traditional approaches has to work with linear relaxations of the problem to improve the solution, which also is considered here, trying to identify when a better solution for the whole system is available. One of the best approaches that had this kind of measure (not direct) was the one proposed by Caprara et al. [3], already mentioned, when some bounds were used in order to detect when a set of pairings *could* lead to better solution. In this case the difference comes from the possibility of detect directly and clearly when a solution is better for the whole system.

This procedure proposed is helped by a column generation scheme, because new shifts are created to improve the solution, but instead of using traditional mechanisms that cannot control quality as it was intended here, a constructive procedure is used. New columns will be accepted considering their reduced cost, but they will be constructed within certain limits in order to maintain the characteristics desired.

6.1 Networks

To create pairings, spells and shifts it is always necessary to create networks to represent the units from where they are constructed (for example, segments are the units to construct shifts and they are represented in a segments-network). Here these ideas are considered, but treated in a different way, because rosters are constructed as it was presented before (and a network of shifts to construct rosters is not necessary) and because quality is a main issue and the networks created have to permit to work easily in that sense.

Segments are represented (from a theoretical perspective) in a network as nodes and there are arcs between those that can be sequenced because the stations (or depots) of arrival/departure are the same. Usually the segments network is constructed in order to create pairings or shifts, but in this case there is a intermediate product before create shifts: spells. These spells will be the elements in a new network to create finally shifts.

Because quality is a main objective of this approach, spells are created

having quality considerations. There are two main characteristics that defines the quality of a spell. The first one is how many services are included in the spell. Because a spell is by definition a set of segments that has no legal resting time inside it, the presence of more than one service means wasted time and also a bigger chance of causing a disruption if the connection among services is not done within the time window available. Then, a spell with fewer changes is preferred to a spell with more changes of services. Because of the problem of possible disruptions, the second characteristic that defines quality for a spell in this approach is how probable is to produce a disruption when a change of service is needed. The time window to make the change of service can be considered better if it is not 'too tight' to make the connection and it is not 'too long' because it would mean wasted time. This characteristic is controlled by parameters (they will be described further on) and the procedure is designed in order to sacrifice the quality of these connections in order to improve the solution if desired. In any case, according to the design of the procedure, this decision is taken only in more advances stages of the optimization to include spells with these 'worse' characteristics in the smallest quantity as possible.

The first iteration of the procedure and the initial solution are done using only spells with no change of service. The following iterations include spells with changes of services which maximum is controlled by a parameter.

Figure 6.1 shows the general idea of the optimization procedure and in the box 'Preprocessing' it is included the creation of the first set of spells and shifts. Section 6.5.1 shows an example of the spells created in this stage. New spells (with service changes) are constructed in the box 'New spells' of the same figure.

The spells created (in the preprocessing stage and the new ones created during the optimization stage) are included in a network to obtain shifts. It is possible to think in every spell just as a segment, with the difference that an original segment is covered by several of these spells. Every spell is represented as a node and connections between nodes exists if they can be sequenced in a feasible shift. This means that the ending time to consider is not the time when the driving period finishes, but when the legal rest that correspond to the spell is finished. Because of this every path in this network is a possible shift. The only pending issue is to close them in order to make the return to the depot of every driver, which can be done the same

day or the following one if two-days shifts are permitted.

Traditional procedures construct shifts from segments directly which has two disadvantages. The first one is to control feasibility after each segment is added. In this approach feasibility is also controlled, but because of the use of spells the possibilities tested are fewer. The second disadvantage is that in a traditional approach the quality is not a main issue unless it is forced, but the structures used are not the ideal ones to induce such characteristics. Instead, using a spells network as an intermediate step allow to control in a much better way the resulting quality.

In terms of the code the implementation of these ideas was done using several related sorted arrays instead of traditional representations. For example, the segments network is in fact made of two lists by depot (arrivals and departures) sorted by arrival time (ending time of the segment) and departure time (beginning time of the segment). Different stations of a same depot are not split, but it is controlled in the feasibility check if an eventual transference time between stations is satisfied or not. A parallel information that is kept is the most immediate connections for every arrival (next segment) and departure (previous segment). If an arrival is from a intermediate segment of a service the closest connection will be at least the next segment in the service. Because both lists are sorted by time, only knowing the first element all the possible connections are immediately available. It is only necessary to go forward or backwards depending on what it is wanted to create more connections and constructs spells. The same idea was used to represent the spell network, but using as ending time the time when a driver is free to take another spell. Certainly to create these lists need some time, but it is a great help when it is needed to construct shifts or pairings with good quality and the final performance of the whole procedure and the results obtained justify this idea.

6.2 Modular Design

One of the issues of most of the approaches proposed in the literature is that they are developed for particular instances. This approach was developed trying to make it as general as possible.

The fix part of this approach comes from its philosophy. For example, to use weekends as bridges between weeks of work and to construct the rosters

as it was shown cannot be changed because it is part of the basic definition of the model and procedure. In that sense what this approach also suggest is a management idea about how to perform the work. Usually what is controlled by laws, regulations and agreements do not control the general structure used to construct shifts, weeks, weekends, or rosters but give some general ideas and an average behavior. For example, in the Chilean law for drivers, there is an average of at least one free day every week (every seven days) that has to be given as weekend, but this not mean that every week a day has to be taken. It only means that in the long term the average has to be at least one day every week.

To make this application independent of the particular mode of transport used it was necessary to restrict some issues to specific modules. For example, feasibility rules are depending on the mode of transport and the company, and even more, even within a company it may depend on the geographical region where the work is performed. Then, it is necessary to have a module that control the feasibility, because if needed, changes are made inside it without changing the general solution procedure.

The main modules considered are:

- Feasibility Control: All rules indicated previously that concern particular pairings and shifts are controlled here. If quantities differ from one application to another, then it is necessary only to modify the values within this module. Some additional rules may be included without great problem. Feasibility issues that involve several shifts are not controlled through this module (For example, a limit in the quantity of two-days pairings in the solution or for a particular depot).
- Spell Construction: Spells are constructed in this module. If a different philosophy of construction is needed, only this module need to be changed.
- Shift Construction: Shifts and pairings are constructed in this module using the spells constructed by the *Spell Construction Module*. Just as the case of the spell construction, if a different philosophy of construction is needed, only this module need to be changed.

From these modules one the fix features can be seen. The procedure

works with spells as a intermediate step to create shifts, and it cannot be changes in the current version. If a direct construction of the shifts from the segments is wanted, it necessary to make a major change in the implementation.

Some other modules were considered at certain point but not included finally. From these ideas, the one that may be important and implement in future versions could be a preprocessing of the network, because many times to consider the original segments has no mayor advantage, but some disadvantages because increases the size of the problem almost unnecessarily. To make a reduction of the segments and create pseudo–segments (set of segments that for the procedure will be treated as one normal segment) may help to speed up the procedure. It may possible to return to the original segments if necessary in more advanced stages. This idea was considered but finally no used in this approach precisely in order to work with bigger instances and analyze the performance under those conditions. In any case, there is a parameter that controls how short spells can be, which produce almost the same effect that this preprocessing could do.

6.3 Rules

In chapter 3 some typical rules were announced. Here are presented the rules as they were applied in the solution procedure which are based in the values founded in Chile, from where this study started.

1. Continuous Driving Time: A 5-hours limit was used. It may seem a long period specially for European regulations, but because of the geography it makes some sense on that situation.

2. Resting Time After a Driving Period: The rule was set as a percentage of the driving time accumulated before the pause. A 40% was set as the typical value, also from the Chilean legislation.

$$resting\_minutes = working\_minutes \times 0,4 \tag{6.1}$$

3. Weekend Resting Time: The quantity of weekend days was defined according to the set of rules defined in the section 5.1.2.

4. Return to the Depot: A driver has to return to its depot after at most two shifts. This means that shifts that return to the depot the same day and couples of shifts linked by a daily rest outside the depot of the driver, and returning to the depot after the second day are the only two forms allowed. In this second case, the only special treatment is that both shifts have to be included (or excluded) in the solution at the same time. Both shifts form at least one pairing and it is represented in the model as a single column. The coefficient will indicate the two-days property.

5. Maximum Quantity of Driving Hours by Day: A high daily limit was establish (12 hours, which is the Chilean value) which at the end is never reached, because of the other rules do not permit it, specially the maximum length of a shift.

6. Maximum Quantity of Total Hours by Day: It is about the length a shift can have. A limit value of 14 hours was used as the basic value.

7. Minimum Daily Rest: 8 continuous hours were considered as the minimum daily rest. This is specially important for two-days pairings, because for single day pairings thanks to other rules and the structure used a driver does not receive ever such a quantity, but more.

Other aspects usually considered but not controlled in this application were:

- Maximum Monthly Working or Total Hours: To test the maximum efficiency this procedure could reach, the monthly hours were not limited, but studied. In any case, the monthly hours were equalized among rosters of every depot in a post-optimization step.

- Lunch Pauses: They were not directly considered, but fused with resting times due driving. To include explicitly lunch pauses it only necessary to add the condition to the feasibility module and maybe in the shift construction modules which at most could make unfeasible some shifts.

6.4 Parameters

The following parameters are considered:

- C_Max: Maximum number of changes a spell can have. This not mean that in the final solution there will be spells using this quantity of changes, buy there is the chance of it. The real selection will depend on if they are necessary for improving the objective function or not and the procedure is designed to include those with 'too many' changes in the minimal possible quantity. The construction of the spells is made using a reference segment to start, and every segments has its own count about the maximum level used so far.

- $per\_tc\_min\_seg$: Minimal percentage to control the connection within a spell when changes of service are used. When a spell is constructed and a connection with a segment that do not belong to the same service is included it is necessary to consider a minimal connection time to cover situations like if the previous segment finishes after the time scheduled or it is necessary some time to take the connection. per_tc_min is a percentage of the resting time due driving accumulated so far in the spell. For example, if the accumulated resting time is 1 *hour*, and $per\_tc\_min = 20\%$, then all connections under $per\_tc\_min \times 1h = 20\% \times 1h = 12$ *minutes* will be ignored because the time is too short to make the change of service without risking a disruption in the schedule of services. The basic value used was 40%. It was defined over the resting time and not the accumulated working time because a spell cannot have a legal rest inside, and then the control has to be done keeping the times within the window that defines the resting time that will no be taken.

- $per\_tc\_max\_seg$: Maximal percentage to control the connection within a spell when changes of service are used. The intention behind this parameter is to control how much will be waited for a connection without taking the resting time accumulated. For example, if $per\_tc\_min = 70\%$ and the accumulated resting time is 1 *hour* then if a connection with another service takes more than $per\_tc\_max \times 1h = 70\% \times 1h = 42$ *minutes* it will be ignored because there is too much wasted time and it would be better to take the resting time and after that follow

with the shift. The basic value used was 90%. It was defined over the accumulated resting time because of the same reason indicated previously in *per_tc_min_seg*.

- *per_tc_min_spell*: Similar to *per_tc_min_seg*, but it controls connection between spells to construct shifts. The value used was 0%. It means that when the resting period due driving of a spell is finished, the driver can be inmeadiatly assigned to another spell.

- *per_tc_max_spell*: Like the previous parameter, this one is similar to *per_tc_max_seg*, but it is also referred to spells. Different values were used on each instance, because some of them did not improve for higher values and then the lowest value is preferred one to avoid wasted time.

- *time_trigger*: On every iteration of the optimization procedure new spells (several) are created using as reference a segment and it is time consuming, specially because these spells are used to construct shifts, which is also time consuming and most of them are not going to be useful to improve the solution (segments with more chances are treated first based on it dual value). Because of this, a time limit to create new spells is used and this parameter is that time limit. The basic value used was 5 seconds.

- *min_spell_time*: Spells are conditioned to have a minimal time in order to avoid shifts too atomized, which can also induce disruptions. A minimal long of 40 minutes (driving) was considered.

6.5 Procedure

The idea of the main optimization procedure is to work with the linear relaxation of the problem (model 5.9 - 5.10 but with the variables as 5.12) creating on each iteration new shifts that can improve the linear objective, and stopping when a criteria is satisfied. On every iteration there is a checking about the best integer solution that can be obtained from the current linear solution, because as it will be shown, the best linear solution (at the end of the procedure) does not produce necessarily the best solution for the system. This effect comes from something quite well known in the crew

scheduling and rostering problems and it is the existence of several alternative optimal solutions, and because the non-linear nature of the rostering problem. For a same value of the relaxed objective function, and the integer solution too, there are different results for the real system. In some sense, this was the objective searched in the rostering optimization phase of traditional procedures within the limitation of the solution founded for the crew scheduling problem.

After this optimization cycle two post-optimization procedures are applied. The first one is aimed to search a better general solution, testing some columns changes. This not change the value of the linear relaxation, but it may change the value of the real objective function. The second post-optimization procedure construct the rosters and also try to equalize for every depot the quantity of average monthly hours of every roster, in order to make them fair for every driver.

The complete general idea of the procedure can be seen in figure 6.1.

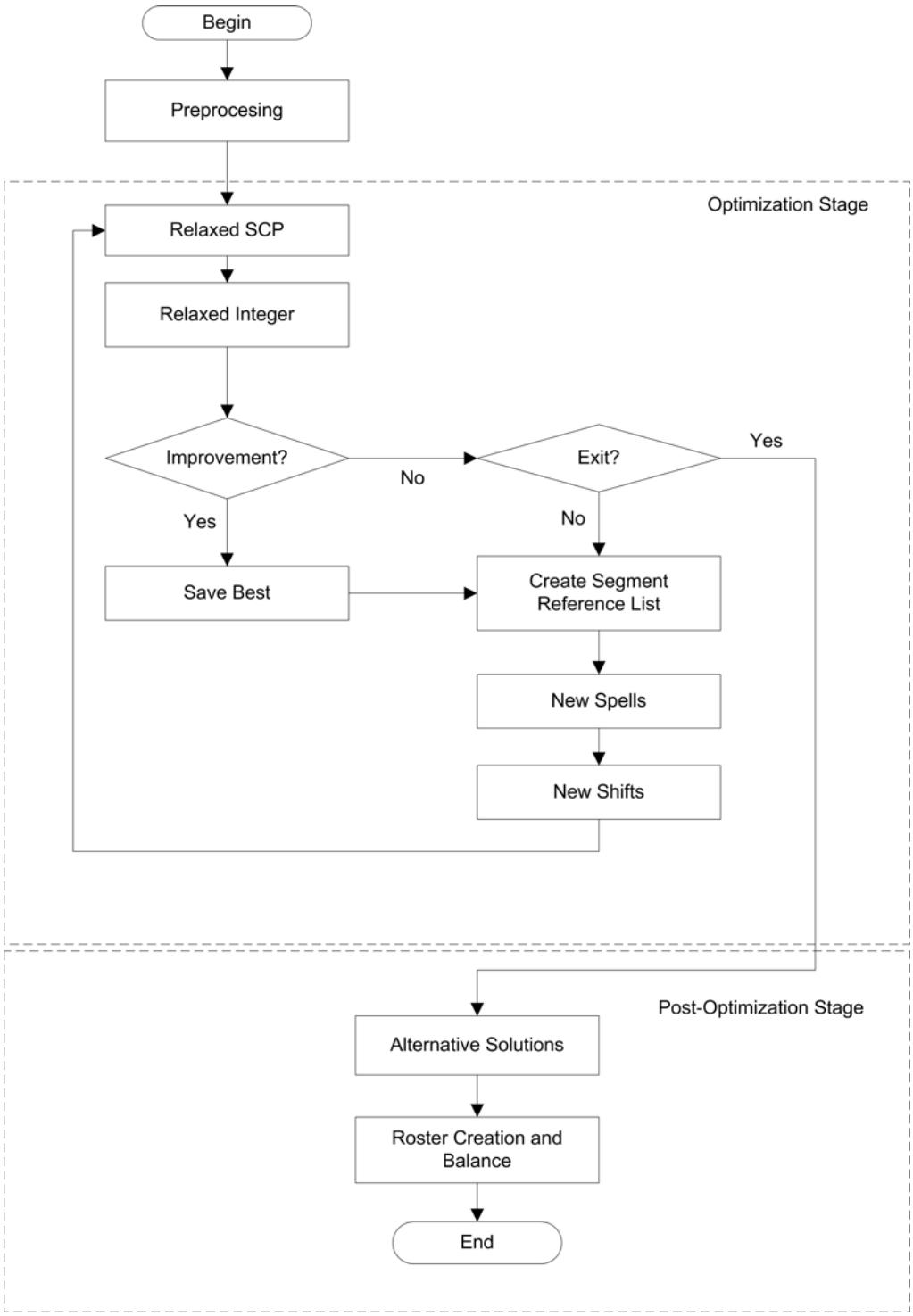

Figure 6.1: Optimization idea

6.5.1 Preprocesing

This stage has as objective to prepare the information that will be used during the following stages, mainly creating the basic set of spells and the basic set of shifts. The preparation of the arrays and information needed takes some time, but it is helpful for obtaining shifts, pairings, and spells of good quality and it will speed up other parts of the procedure, such as the checking of feasibility.

The construction of the spells is done in the following way: Because at the beginning no changes in service are allowed, what is done is to construct all possible spells using only segments of every service, considering all possible feasible lengths. For example, if a service is as follows:

Table 6.1: Example 1: A service with 4 segments

Segment	Begin	End	Length (Minutes)
1	08:00	08:45	45
2	08:45	09:30	45
3	09:30	11:30	120
4	11:30	12:25	55

Then it is possible to create this set of spells (considering as limit 5 hours – 300 minutes – of continuous driving time):

Table 6.2: Example 1: Possible spells

Spell	Segments Included	Length (Minutes)
1	1	45
2	1–2	90
3	1–2–3	210
4	1–2–3–4	265
5	2	45
6	2–3	165
7	2–3–4	220
8	3	120
9	3–4	175
10	4	55

If the time limit indicated changes, also the possible spells that can be constructed change. For example, considering a time limit of 4 hours of continuous driving time spell 4 would not be feasible.

In case that segments are longer a situation like this is founded. Considering a 5 hours limit in the continuous driving time and the following service:

The possible spells are:

Table 6.3: Example 2: A service with longer segments

Segment	Begin	End	Length (Minutes)
1	10:00	11:30	90
2	11:30	13:10	100
3	13:10	15:10	120
4	15:10	17:10	120

Table 6.4: Example 2: Possible rosters

Spell	Segments Included	Length (Minutes)
1	1	90
1	1–2	190
1	2	100
1	2–3	220
1	3	120
1	3–4	240
1	4	120

In this case, combinations such as 1–2–3 or 2–3–4 are not possible because they are too long (too many continuous driving minutes) and not feasible.

From these examples it can be seen that a same segment may be covered by different spells.

This set of spells will form the initial network to construct shifts, and this network is updated every time new spells are created.

Shifts are constructed in a similar way, but starting from a reference spell (every spell is taken as a reference for the initial set of shifts). In the spells network, starting from the reference segment, several parallel paths are constructed within the limits established by the parameters *per_tc_min_spell* and *per_tc_max_spell*, plus the limits indicated by the rules (length of the shift mainly). When one of these paths becomes a pairing it is saved and more spells are added in order to create longer shifts if possible. If from a reference segment no feasible shift could be obtained all the paths constructed so far are used to construct two-days shifts, inserting a daily rest and then adding more spells. This is not done with all the reference spells to limit the quantity of two-days shifts, because from the quality of life perspective, it is not good for drivers to have too many daily rests outside their depots, and because there are so many possibilities of two days pairings that the final quantity of columns would be too big and most of them useless.

From a reference spell, more spells are added before, after and before and after it, producing all the possible combinations. It may seems a huge task, but because the work is done with spells and not segments it is not

so time consuming at all, even for big instances. Moreover, because spells with no changes are used in this stage and they are the spells with best quality, to spend time trying to create all the feasible combinations with these characteristics is not a waste of time at all. The ideal situation is to use in the final solution as many of these shifts as possible because they are constructed with the best set of spells. Traditional approaches usually cares nothing about issues like this one, when they are fundamental on reality.

It is clear that this procedure to create shifts is not efficient if it is compared with traditional approaches, even more if in consecutive iterations the size of the spell network increases, but it is important to keep in mind two things:

One, traditional approaches are focused in obtaining quantity and not quality. Feasibility is the only goal when shifts are created, and solution methods usually are not designed neither to prefer good shifts over bad ones (quality is always a subjective matter in any case). Apply that idea in an approach that searches for quality as a main goal would be just non sense and because quantity is not a goal here, but quality, a less efficient procedure has more sense if it allows to create the shifts as they are desired if computational times are within reasonable limits.

The second reason is related to the quantity of possibilities tested among all the feasible shifts that could be created. One basic characteristic of these problems is it is possible to create a huge quantity of feasible shifts, but most of them will not even considered during the solution procedure. This issue has always been seen as a problem, but considering the objective searched here is much more an opportunity. It would be reasonable to expect good results using a 'quite restricted' set of shifts instead of a huge quantity (the idea of 'restricted' on these problems in any case could be mean thousands of columns, but with respect to the total, they are a restricted set) but controlling those shifts created. In that sense, this approach tries to make a deep search on the possible shifts with the best characteristics and not consider those that are feasible but with features not wanted. The only two limitations that need to be in mind are that this kind of focused search would only useful if the processing times are contained within reasonable levels. The second limitation is that it is true that a more 'free' procedure to create shifts could lead to improve the solution, because it would explore in areas that a limited approach would not investigate and that may be fundamental

to obtain the best mathematical solution, but it would be against the very essence of the objective of this approach.

6.5.2 Initial Solution

What is called the initial solution is the first solution obtained from the linear relaxation, which main objective is to obtain the values of the dual variables to create more shifts and improve the linear solution. There is also an initial solution for the integer relaxation of the problem which is also a solution for the system (the original model). The difference between this last two is that the integer relaxation cannot say how many drivers the solution has, but it defines the columns to be included. The original model is needed for that (its objective function).

6.5.3 Optimization step

Figure 6.1 shows the optimization idea inside the section labeled *Optimization Stage*.

The main construction blocks and their functions are:

Relaxed SCP

It obtains the solution of the relaxed model (equations 5.9, 5.10 and 5.12). In the first iteration it uses the columns created in the preprocessing stage, and during the successive iterations it uses the columns of the solution of the previous iteration plus the new shifts created. This step has to check if in case that two-days shifts are used, the conditions needed to create twin-rosters are satisfy. If they are not satisfy the two-days shift is retired and the problem re-optimized.

Relaxed Integer

From the linear solution obtained an integer version of the set covering model is used to define an integer solution. This solution is evaluated in the original objective function in order to detect if it better than the current best solution. In the first iteration the integer solution is set as the best solution. This best solution may be improved more in a post-optimization stage, when alternative solutions are searched. This is possible because at this stage of the procedure the solution founded is based on the linear improvement, and

because of the nature of the rostering it may not be the actual best solution. It may be seen as a reference where to search some more.

This step also has to check if in case that two-days shifts are used, the conditions needed to create twin-rosters are satisfy, because from the linear relaxation to obtain the integer solution those conditions may have been lost because of the chosen shifts.

Control Exit

Severals conditions to stop the iterations were tested, but finally only two remain and results are shown taking as reference that stopping criteria. The first criteria is to allow only a few iterations without improve the value of the linear objective function. The quantity of iterations is controlled by the parameter *limit_same_solution* and was set to 3 iterations. A normal stop criteria (to stop when it is not possible to obtain new columns with negative reduced cost) was not used because of the way the construction of the new spells is done. As it will be shown, every time a segment is used as reference to construct new spells one more level of changes is used, and to detect columns using a restricted shortest paths approach as usual has the problem that not only the general rules of the system have to be respected (working time, resting times, and so on) but also there are 'rules' or limits that are defined and controlled segment by segment, and apply such rules to a restricted shortest path method would be almost insane. At most, a restricted shortest path approach to control the stopping of the linear optimization could say that there are no more columns to be processed, but if it indicates that there are more columns only considering the general rules may not be true because new spells and shifts are constructed considering more restrictions that are not included when those paths are calculated. The second criteria is when all segments have been used to construct spells with the maximum quantity of changes allowed and no more shifts can be constructed, which usually is more possible to find in small instances that are processed and solved quite quickly.

Create Segment Reference List

To create spells a segment needs to be given as reference, but because to create spells and then shifts starting from every possible segment is just

too time consuming (and not useful at all), segments has to be classified in order to prefer as reference those that have more chances of participate in a shift that improves the linear solution. Then segments are sorted by reduced cost and processed in that order. The order is change to a random approach after a iteration when no improvement in the solution was possible. If it achieves an improvement, it returns to the sorted approach. If not, it continues testing randomly.

New Spells

A set of new spells is created using the reference list given by the previous step. This step is controlled by time (parameter $time\_trigger$) and then not all of the segments are tested. In the first iteration every segment has been already used as reference to construct spells with no changes in the service (in the preprocessing stage). When the procedure arrives to this point, for all the segments that are processed during the time given it is used one more level than the last one used. For example in the first iteration all segments processed are used to construct spells with one change, and in the second iteration if one of these segments already processed with one change is considered again, it is used to construct spells with two changes. If it was not considered in the previous iteration, it will be used to construct spells with one change only.

If a segment has been processed already with the maximum quantity of changes allowed, it is ignored.

In this way the procedure in a same iteration may produce spells with different quantity of changes as a way to improve the solution as much as possible but also controlling the quality. After create the spells the spell network is updated before create shifts.

New Shifts

Every spell created in the previous step is used to create new shifts. The procedure is the same described before.

Because the spell network was updated, the new shift could be constructed using not only the new spells but any spell created so far if they respect the limits defined to construct shifts. Because of the accumulation of spells the network becomes more dense, and more paths have to be tested,

but the processing time is balanced because usually there are not too many spells to test.

6.6 Post-optimization: Alternative Solutions

This step tries to improve the best solution founded in the optimization step changing some of the shifts (eliminating them) and re-optimizing using some columns that were saved from the optimization procedure.

At this stage the solution has the values of the rosters for every depot defined. The idea then is, starting from those values, detect the depots that are less efficient from the rostering perspective. How to measure this efficiency is quite simple. The formula to define the quantity of rosters a depot has is:

$$Q\_rd_i = \left\lceil \sum_{j=1}^{S_i} \frac{rd_{ij}}{long\_week} \right\rceil$$

where:

- S_i : Quantity of shifts of the depot i.
- rd_{ij} : Quantity of resting days associated to the shift j of the depot i.
- $long\_week$: Long of the week of work.

If the linear version is considered:

$$Q\_rd_i = \sum_{j=1}^{S_i} \frac{rd_{ij}}{long\_week}$$

The rest of a integer division made of these two values give us the quantity of extra days this depot has that produce one extra driver. For example, if:

$$Q\_rd_i = \sum_{j=1}^{S_i} \frac{rd_{ij}}{long\_week} = \frac{13}{6} = 2 + \frac{1}{6}$$

1 would be the rest of the integer division, and just because that extra quantity of resting days in that depot it is needed another driver. If those extra days, where founded, could be eliminated the system would need some

fewer drivers. It is easier to eliminate an extra driver where the extra days are not many (1 or 2), but the procedure that search for an alternative solution test any possible reduction considering all those depots where the quantity of rosters was not defined exactly (by an integer number).

The produce search for alternative solutions using as additional columns those that were used to define the best solution founded so far (the set used in the linear relaxation, reduced according to the reduced cost). It was considered to consider more columns, but the problem is that during the linear optimization procedure, there is no a particular clue about what columns of the whole set created may be useful, and save them all could be too memory-demanding.

There is an alternative solution when this situation is present as it will be shown (much easier to construct), but not always may be hard to implement on reality because may produce a situation less fair for some drivers that would have a longer week of work.

The instances where tested with this post-optimization procedure active and non active, because it is time-consuming and as it will be seen, there are only a few cases were it gives improved solutions, which with respect to the global solution (the total quantity of drivers) are not much important (just a small percentage of improvement.

6.7 Post-optimization: Searching fairness in rosters

As it was said before this model, because of its special structure, permits the optimization of the quantity of drivers without construct the final rosters immediately. The task of constructing the final rosters is part of a post-optimization procedure that may be aimed to different purposes. If the rules indicated previously are respected, the quantity of rosters and drivers do not change no matter what is done with the final set of shifts.

In this case a tabu search procedure is applied to construct the final set of rosters. The optimization procedure works for every depot independently because every subset of rosters (defining as a subset those that belong to a particular depot) is isolated of the rest.

The objective function is defined as the sum of the absolute value of

the 'distance' of every shift (its average monthly working hours, mt_{ij}) to a *target* value of monthly working hours (target_hours), which may depend on a legal limit or a company's policy.

$$Min \sum_{i=1}^{R_j} \mid target\_hours - mt_{ij} \mid \tag{6.2}$$

where R_j represents the set of shifts assigned to depot j.

In this case the target is defined depot by depot, as the average working contain of that depot. Knowing how many working minutes the depot has and how many rosters the depot has it is possible to calculate this average, and then, the procedure tries to construct all of the rosters as close as possible to this average, in order to, at least, having a fair distribution of work contain within every depot.

For example, having the following roster:

Shift	Daily Working Minutes	Resting Days
X	400	2
Y	410	1
Z	425	3

Its average monthly working hours are given by equation 5.5:

$$monthly\_hours = \frac{30 \times (400 + 410 + 425) \times 6}{(3+1) \times 6 \times 60} = 9262,5/60 = 154,375$$

The distance of this roster to the target comes given by:

$$distance\_target = target\_hours - 154,375 \tag{6.3}$$

and it will be positive or negative if it is under or above the target, but for the objective function the absolute value is considered, because if not the total value would not vary at all.

Other elements of the tabu search procedure need to be introduced before further information is given:

- Tabu List: Shifts inserted in a roster cannot be moved to another one for M iterations ($M = 3$).
- Neighborhood: All the exchanges that can be done among couples of rosters according to the quantity of resting days and possible idle

resting days without assign.

- Aspiration Criteria: If the movement reduce the value of the objective function.

An example will be given to show how a neighborhood is constructed. Having the following two rosters:

Shift	Daily Working Minutes	Resting Days
X	470	2
Y	450	1
Z	425	2

Shift	Daily Working Minutes	Resting Days
A	350	2
B	365	1
C	395	3

Some possible movements are:

- When both rosters has a shift with the same quantity of resting days (for example exchanges Y-B and X-A).
- When a roster has 'free days' to receive a certain shift (for example, insert shift B into Roster 1)
- When adding resting days from one roster we equal resting days of a shift in the other one (for example, C-(XY) and C-(YZ))

All of these movements has a value, because they change the average monthly working time each one has. First it is necessary to calculate the average monthly working times for the original rosters:

$$Monthly\_hours\_Roster\_1 = \frac{30 \times (470 + 450 + 425) \times 6}{(3+1) \times 6 \times 60} = 168,13$$

$$Monthly\_hours\_Roster\_2 = \frac{30 \times (350 + 365 + 395) \times 6}{(3+1) \times 6 \times 60} = 138,75$$

If, for example, the movement Y-B is done, the new value for every roster will be:

$$Monthly\_hours\_Roster\_1 = \frac{30 \times (470 + \mathbf{365} + 425) \times 6}{(3+1) \times 6 \times 60} = 157,5$$

$$Monthly\_hours\_Roster\_2 = \frac{30 \times (350 + \mathbf{450} + 395) \times 6}{(3+1) \times 6 \times 60} = 149,38$$

Then the movement has a value defined by the reduction of the total sum of distances:

$$movement\_value = \mid target\_hours - 157,5 \mid + \mid target\_hours - 149,38 \mid$$
$$- \mid target\_hours - 168,13 \mid - \mid target\_hours - 138,75 \mid = -21,26$$

It has been mention that two-days shifts needs a special treatment during this optimization, because to fit them it is necessary to create a 'twin' roster: A roster that has the same quantity of resting days. For example, considering the two-days shift of table 6.5.

Table 6.5: Example: A two-days shift

Item	Type	Beg_time	End_time	Length
1	S	08:00	10:35	155
2	R	10:35	11:37	62
3	W	11:37	12:30	53
4	S	12:30	15:50	200
5	R	15:50	07:20	Daily Rest
6	S	07:20	09:35	135
7	R	09:35	10:30	55
8	W	10:30	12:00	150
9	S	12:00	15:15	195

The first four rows represent the first day, and from the sixth row the second day is indicated. In the tables 6.6 and 6.7 this shift will be shift S, making the difference of the days using $S1$ for the first day and $S2$ for the second day.

Table 6.6: Example: Twin-Roster number 1

Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Shift S1	A	B	A	B	A	B	C	D	C	D	C	D	E	F	E	F	E	F	G	H	G	H	G	H
Shift Y	E	G	G	G	G	G	G	A	A	A	A	A	A	C	C	C	C	C	C	E	E	E	E	E
Shift Z	C	C	C	E	E	E	E	E	E	G	G	G	G	G	G	A	A	A	A	A	A	C	C	C
Rest	G	E	E	C	C	C	A	G	G	E	E	E	C	A	A	G	G	G	E	C	C	A	A	A

Table 6.7: Example: Twin-Roster number 2

Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Shift S2	H	A	B	A	B	A	B	C	D	C	D	C	D	E	F	E	F	E	F	G	H	G	H	G
Shift P	F	F	H	H	H	H	H	H	B	B	B	B	B	B	D	D	D	D	D	D	F	F	F	F
Shift R	D	D	D	D	F	F	F	F	F	F	H	H	H	H	H	H	B	B	B	B	B	B	D	D
Rest	B	H	F	F	D	D	D	B	H	H	F	F	F	D	B	B	H	H	H	F	D	D	B	B

Four more shifts are needed to construct these twin rosters, Y, Z, P and R. To construct this structure shifts Y and P needs to have the same quantity of weekend days, and also shifts Z and R. In this case, both days of the shift S has the same quantity of weekend days, but it is also possible to construct these twin rosters even if the two-days shift gives different quantity of weekends days for every day. In that case there will be only an 'unfair' situation with one driver that will take a shorter break than his partner during that week (the one that was performing the complementary day of the two-days shift). In any case, the total quantity of resting days during the whole roster will be always the same for all of the drivers.

Another possible solution, when it is not possible to construct a twin roster, is to give some extra weekend days to some shifts of the depot (it may be a shift of the two-days shift or a different shift) in order to fit them all in twin rosters. This would be specially possible when the quantity of roster of the depot has not been defined as an exact multiple of the week of work. If not, it would be necessary another extra driver in that depot to make the rotation possible.

This control also has to be done in the optimization procedure to avoid the selection of a two-days shift in a depot where it is not possible to construct a twin-roster.

In this post-optimization stage there is another tool to regulate the monthly hours to, but it is much less effective than the movements shown. Because a set covering problem was used to solve the relaxation, and as consequence there is a certain degree of overcovering (segments where two drivers are assigned), this stage also has to decide which driver will perform the job and which one no. It is like a fine-tunning stage which effects are usually quite small because it is about just a few hours at most and not hundred as the total work contain of a shift has. Another possible solution (and maybe more fair) is to allow to each driver to perform those segments the 50% (3 days each one during the week of work where the overcovering is present) of the times and in that case it is not necesary to make an extra optimization to distribute those hours because it would have a null effect.

During the construction procedure, the forward rotation is not really considered, because it does no have an effect over the rosters created. Once the rosters are created on every one of them the shifts have to be sorted as the forward rotation indicates.

Within this module could be possible to include more rules about how shifts are associated, for example, to represent personal preferences. The problem in that case would be that may be not possible to satisfy all those preferences, and then there would be two possibilities. The first one, to satisfy as many as possible those requirements, but not all of them, and the second one, to search an alternative solution that could permit satisfy them. In the literature is the first option the one considered when preferences are part of the problem. This is also valid if more rules to create rosters are included, trying to obtain some good characteristics that are not fundamental, but desirable.

Chapter 7

Computational Results

This chapter shows the computational results obtained. The first section shows some information about the instances considered. The second section shows the basic results (processing times and quantity of drivers obtained). The third section is dedicated to show some detailed results about the regularity and quality obtained which were one of the main objectives of this work.

7.1 Instances

Ten instances were considered, aiming to have different geographic distribution of the cities and different density. Most of the approaches shown in the literature are developed to respond to a particular network and then the performance of the methods are shown only over that area. These are good answers to respond to a particular problem, but the real virtue of a procedure only can be known if it is tested in a wider set of situations.

For example, this study was born from the interest in the Chilean long-distance bus system where cities are usually far away one from each other. Compared with the Italian case, where it is usual to find cities within 30-60 minutes of travel, the Chilean case presents traveling times of several hours between couples of 'near' cities and almost all the cities are in a north-south line, with scarce movements est-west.

Figure 7.1 shows the geography of Chile. The smallest instance tested is contained between the cities of Santiago and Concepcion (little more than 500 Km.) while the biggest one (related to Chile) is contained between Arica

and Puerto Montt (little more than 3000 Km.).

Others countries considered where Italy, Ecuador and Switzerland, plus some mixed instances.

The *distance* considered was in fact the *time* that takes to arrive from one station to another one. Time was considered instead of real distance because it is the relevant measure to create work schedules.

The instances are not real schedules of any company, but they are real-based. They were created using real geography as it was said and defining the frequency of the services according to the size of the cities crossed and/or using an approximation of the real frequency used when available.

The instances considered were the following:

Table 7.1: Instances - Segments perspective

Name	N_Seg	N_Serv	Short.Dist.	Long.Dist.	Av.Dist.	Std.Dev.
CHL_288	288	136	10	250	133,52	81,13
CHL_518	518	238	10	250	113,70	75,81
ITA_908	908	216	5	65	26,76	14,28
CHL_790	790	306	10	255	123,31	69,29
CHL_1086	1086	382	10	255	127,31	73,27
CHL_1392	1392	508	10	300	141,0	76,03
SWI_928	928	494	20	185	61,78	34,60
ECU_1526	1526	422	15	295	72,33	50,60
MIX_1426	1426	454	5	250	58,34	62,96
MIX_4898	4898	1536	5	295	61,23	50,08

The column *Name* contain the name of the instance. *N_Seg* is the number of segments the instance has and *N_Serv* is the quantity of services the instance has. *Short.Dist.* means shortest distance and indicate the length of the shortest segment (in minutes), meanwhile *Long.Dist.* is the longest segment distance (also measured in minutes). *Std.Dev.* is the standard deviation as measure of variability of the lengths of all segments.

The instances named CHL**** are based on the Chilean geography. More segments they have, bigger the area they cover. ITA_908 is an instance based on the Italian geography, particularly centered on the Emilia-Romagna Region, reaching the neighbors regions as limit. SWI_928 is a instance based on the geography of Switzerland and ECU_1526 is an instance based on Ecuador's geography. Both instances called MIX*** are created mixing some of the other instances. MIX_1426 is a mix of the instances CHL_518 and ITA_908 with a homologation of three cities that are at almost the same travel distance in both countries. MIX_4898 is a mix

Figure 7.1: Chile's Geography

of three instances (CHL_1412, SWI_928 and ECU_1526) plus some new services that cross from one instance to the other. There is no homologation of cities in this case.

Considering services, the characteristics of the instances are the following:

Table 7.2: Instances - Services perspective

Name	N_Serv	Short.Dist.	Long.Dist.	Av.Dist.	Std.Dev.
CHL_288	136	180	465	282,76	102,51
CHL_518	238	90	520	253,86	126,64
ITA_908	216	85	155	111,47	24,02
CHL_790	306	90	910	324,72	211,88
CHL_1086	382	90	910	367,70	225,38
CHL_1392	508	90	1795	391,09	352,56
SWI_928	494	50	250	116,05	71,82
ECU_1526	422	40	645	261,56	167,56
MIX_1426	454	85	520	184,88	116,64
MIX_4898	1536	40	940	197,58	119,23

Columns' names have the same meaning as before, but calculated over the services as a single unit.

Times were set to fit on 5-minutes periods mainly because on real schedules, even if they consider events on every possible minute, to work with time windows so tight as those that could come if detailed times are used could produce problems, for example, when spells with changes are used. Another reason is because usually in long-distances services a distortion of a few minutes is something quite common, and to make this adjustment also could make less probable some disruptions. There is a third reason, and it is that to work with 5-minutes periods reduce the size of the problem significantly because it makes the computational representation easier but good-enough to make the results valid.

7.2 General Results

Results about quantity of drivers obtained and processing times are presented in the following table (7.3). Times are presented in seconds. The implementation was done in C++ and experimentation was done on a processor Intel Core Duo 2 GHZ with 2 GB of RAM. Linear solutions and integer solutions were obtained using CPLEX 10. In any case, the use of

memory of all instances was almost negligible. The post-optimization procedure to search alternative solutions was not active for these results.

Table 7.3: General Results - Best Solutions Founded

Instance	Tot.Time	Ini.Sol.Time	Opt.Time	Q_Sched	Q_rost	Q_total
CHL_288	60	18	39	98	25	123
CHL_518	197	19	175	150	39	189
ITA_908	163	17	146	75	19	94
CHL_790	271	8	254	287	77	364
CHL_1086	237	25	195	390	107	497
CHL_1392	518	32	468	551	151	702
SWI_928	655	23	630	178	46	224
ECU_1526	964	34	929	334	100	434
MIX_1426	770	27	737	229	58	287
MIX_4898	1930	114	1799	891	240	1131

Tot.time represents the total time the procedure took. *Opt.time* is the optimization time, that in other words excludes the pre-processing time (construction of the initials spells, shifts and arrays used) but includes the time of the post-optimization steps (search for alternative solutions and creation and optimization of the rosters). *Bas.sol.time* means basic solution time and it excludes the post-optimization steps, representing only the time of the optimization cycle. It was said before that this model allows to know exactly how many drivers are needed to cover all the working time and how many extra drivers are needed to rotate them. The first quantity is shown in the column *Q_sched* and the second one is shown in the column *Q_rost*. The column *Q_total* is the sum of both quantities of drivers and then the total quantity of drivers of the system.

From these results it is possible to see that time and size has not a direct relationship, but only respect some magnitude order. It seems to indicate that the geography is a much more relevant factor to define the processing time, because only when the change in the size is radical, it influences the processing time. For instances more or less of the same size it cannot be said that the processing time is better when the instances are smaller.

The idea of this approach was to develop a method as general as possible, but due the nature of the instances the same parameters were not possible. For example, for the Chilean instances the density of the connections in some areas is lower than in those areas more populated, and then, it is necessary to allow more wasted time to make connections. The problem of this is to

allow less efficient shifts which means worse quality, so it has to be avoided as long as possible. It remains as future work a version when parameters could be defined from a analysis of the instance (for example, density of possible connections and average length of the segments, even area by area).

It is been said before that the procedure was designed to obtain as many spells with no changes in the service performed in order to make a more robust solution. The following table (7.4) shows information about how many spells were finally used according to the quantity of changes.

Table 7.4: Spells Details

Instance	Shifts	Spells	0	1	2	% 0-change
CHL_288	98	196	189	3	4	96.4
CHL_518	150	320	309	3	8	96.6
ITA_908	75	230	215	8	7	93.5
CHL_790	287	596	580	3	13	97.3
CHL_1086	390	792	761	8	23	96.1
CHL_1392	551	1074	1045	6	23	97.3
SWI_928	178	438	369	25	44	84.2
ECU_1526	334	754	727	7	20	96.4
MIX_1426	229	576	566	4	6	98.3
MIX_4898	891	2181	2179	1	1	99.9

The column *Shifts* indicates the quantity of shifts the solution has and *Spells* indicate the total quantity of spells used by those shifts. The columns *0*, *1*, and *2* indicates the quantity of spells present in the solution with that quantity of changes. *% 0-change* indicates the percentage that spells with 0-change represent of the total quantity of spells used.

From these results it is possible to see that most of the solution was constructed using spells with no change, just as it was intended to be. When changes are used usually the procedure prefers shifts with every spell having changes, even when shifts that have spells with and without changes are also produced and available for optimization. This is obvious in some sense, because with more changes probably is easier to find new shifts that have a better reduced cost to improve the solution. Almost all the shifts constructed are done with just two spells and there are cases when just one was used. This is possible because that single spell has at least one change inside it which make the driver return to the depot immediately without having a resting period due driving outside his depot, and that single spell is also his shift for a day.

It was said before that the monthly quantity of hours was not limited in

order to analyze how much efficiency could this approach achieve. Appendix A shows the detailed results about this for all of the instances. Table 7.5 shows some statistics about it, indicating the quantity of rosters ($Q\_rosters$), the average monthly hours of the system ($Av.Monthly$), the average monthly hours of the 20 % with less working contain on the system ($Av.Min$), the average monthly hours of the 20 % with more working contain on the system ($Av.Max.$), and the standard deviation of the system ($Std.Dev.$):

Table 7.5: Rosters Result: General Information

Instance	Q_rosters	Av.Monthly	Av.Min	Av.Max	Std.Dev.
CHL_288	25	159,5	153,8	165,4	4,1
CHL_518	39	156,4	128,8	174,9	18,2
ITA_908	19	144,1	100,6	168,2	27,1
CHL_790	77	146,3	105,8	166,9	23,2
CHL_1086	107	151,7	112,9	179,1	24,3
CHL_1392	151	150,4	117,1	177,6	21,9
SWI_928	46	139,2	106,8	157,3	22,1
ECU_1526	100	137,7	99,4	163,9	24,6
MIX_1426	58	152,8	125,1	174,1	17,5
MIX_4898	240	140,8	101,6	166,9	24,3

It can be seen that the geography seems to influence in this results. All Chilean instances have more or less the same behavior, while diverse result are founded in the other instances. MIX_1426, an instance made from a Chilean and the Italian instance has results precisely somewhere in the middle of the original.

Traditional approaches usually work with the linear relaxation till no improvement can be founded and then a rostering solution is obtained. More advanced methods, like the one proposed by Caprara et al. [3], control the possible presence of a better crew rostering solution (that is the fundamental idea when integration is considered). As it was mention, this procedure share the same idea with that approach, with the difference that the rostering solution does not need to be constructed and the detection of a better general solution is not only a possibility , but a certainty because it can be calculated. The main point here is to keep through the iterations the best solution found, because there are not guaranties that the best linear solution could give directly also the best integer solution. Because of this, almost never the best solution is founded in the last iteration but before. The following table (7.6) shows that information.

Table 7.6: General Performance: Iterations

Instance	total_it	it_best	total_time	best_time	lin_best	int_best	%_over
CHL_288	10	3	38	15	726	726	0.00
CHL_518	8	3	172	67	1115	1115	0.00
ITA_908	10	4	145	82	554	555	0,18
CHL_790	15	3	251	61	2131	2137	0,28
CHL_1086	13	6	192	121	2952	2954	0,07
CHL_1392	24	12	836	429	4171	4171	0.00
SWI_928	16	10	495	333	1347,5	1358	0,78
ECU_1526	21	16	893	686	2514,5	2516	0,06
MIX_1426	12	4	712	310	1703	1706	0,18
MIX_4898	21	4	1746	326	6554,25	6572	0,27

The quantity of iterations done is indicated in the column *total_it* and the column *it_best* indicates the iteration in which the best integer solution was founded (real solution for the system, because the best integer solution of the integer model is not always the real best one). *total_time* is the total time the iterations after the first one took to finish and *best_time* is the time needed to find the best solution once the initial solution is obtained. *lin_best* and *int_best* are referred to the objective functions values. The first one is the final value of the linear relaxation and the second is the value of the integer relaxation when the best solution was founded. %_*over* indicates how far from the final linear value is the value of the integer relaxation of the best solution.

This can also been seen graphically. In most of the cases just a few iterations are needed to improve the value before falling into a slow improvement. The next example comes from the instance CHL_1086, where the best solution is founded in the 6th iteration, but the procedure tries to improve the linear relaxation (and it does it) for 7 more iterations without improve the best solution.

This last result (how early the best solution is usually founded) suggest that maybe more effort should be focus on the search for alternative solutions instead of optimize the linear value. This idea was suspected because this problem has by nature many alternative solutions and the non-linearity of the roster construction also imply that the best solution is just near the best linear value, and then, more than define a best solution, the linear relaxation establish a reference to search for the best solution for the system. As consequence, to optimize the linear function is not the final answer but a

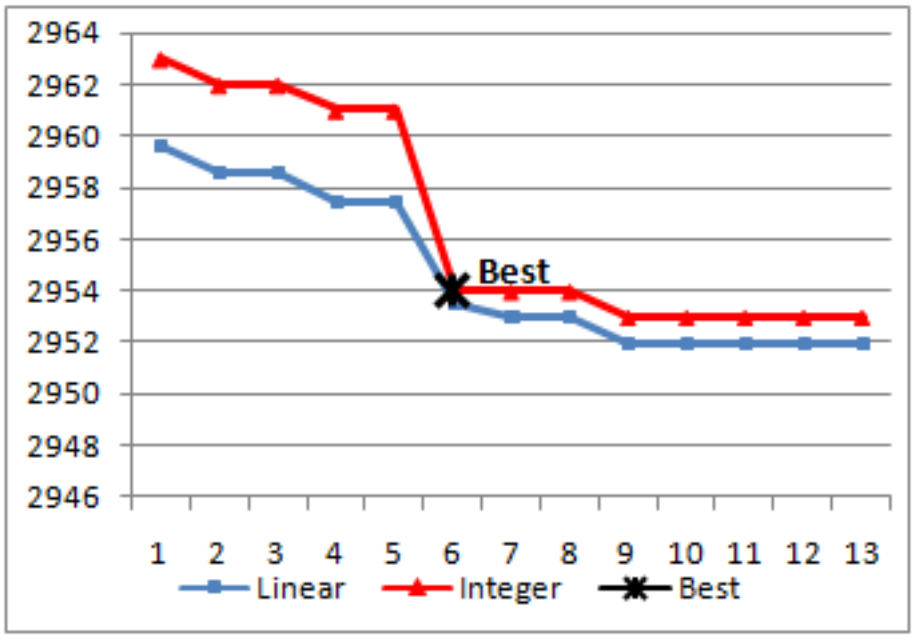

Figure 7.2: Instance CHL_1086 : Linear and Integer Evolution Trough Iterations

starting point from where a better result should be searched, which finally implies (to be really effective) to design a completely different optimization approach.

Traditional approaches usually try to control overcovering (the assignation of a segment to more than one driver, which imply a driver as passenger) because it is reasonable to expect that a lower percent could lead to a better solution. In this case, because the model allow to know the real value of the solution considering both problems, overcovering is not an issue, because it will be the right amount to reach the lowest value of the total drivers. The next table (7.7) shows the resulting percent for the best solutions founded.

It also shows the quantity of two-days shifts used.

It is particularly interesting the case of the instance CHL_1392, because it covers the north of Chile, where cities are very sparse and then the quantity of final two-days is much higher. SWI_928 and ECU_1526 have also high values. Both cases may be explain in part because of regional features. Those countries also have cities in some degree isolated with areas where the quantity of cities are more concentrated, and it is in those isolated places where more two-days shifts are needed.

Table 7.7: Overcovering and Two-days Shifts of the Solutions

Instance	% Over Seg	% Over Work	Q_2_days	% 2 Days
CHL_288	3,125	1,65	0	0,00
CHL_518	4,05	1,37	0	0,00
ITA_908	10,9	10,3	0	0,00
CHL_790	8,1	5,76	4	1,39
CHL_1086	8,56	6,43	6	1,54
CHL_1392	6,25	4,79	44	7,97
SWI_928	7,22	7,24	18	9,52
ECU_1526	10,41	7,79	28	8,38
MIX_1426	7,71	5,20	0	0
MIX_4898	7,23	5,87	30	3,37

Now results when the post-optimization stage of searching for an alternative solution with better general result is considered are shown.

Table 7.8: Improvement Due Post-Optimization Stage

Instance	Q_normal	Q_Improved	%	Time Normal	Time Improved	%
CHL_288	123	–	–	60	–	–
CHL_518	189	–	–	197	–	–
ITA_908	94	–	–	163	–	–
CHL_790	364	363	-0,27	271	465	71,59
CHL_1086	497	–	–	237	–	–
CHL_1392	702	701	-0,14	518	524	1,16
SWI_928	236	–	–	655	–	–
ECU_1526	434	–	–	964	–	–
MIX_1426	287	286	-0,35	770	1212	57,40
MIX_4898	1131	1127	-0,35	1930	3625	87,82

From these results it is possible to see that the effect is quite modest. This can be easily understood specially when the problem has a bigger size, because 'at most' the possible reductions could be equal to one driver for each depot that do not have a quantity of weekends days to assign that is an exact multiple of the week of work. To reduce more than one driver for depot it would be needed a more aggressive search, but intuition says that would be almost impossible to find a solution where more than 6 weekends days (6 because it is the long of the week of work) could be spread or eliminated augmenting the working contain of other shifts or thanks to the substitution of some shifts as this post-optimization procedure does. Everything indicates that a more constructive procedure once a good reference is defined that comes from the linear relaxation of the model could be more appropriated, instead of a linear optimization as it is usually done.

7.3 Regularity and Quality Issues

One of the two main objectives of this work was to obtain regularity in the working and resting hours. Some examples of the quality of the shifts and rosters obtained will be analyzed in this section. Further information is shown in annexes as it will be indicated.

First examples of the shifts constructed are given.

The following figure shows a shift that belongs to the final solution of the instance CHL_288 with no changes on its spells:

Table 7.9: Example: Shift from CHL_288

Item	Type	Seg_Number	Serv_Number	Beg_time	End_time	Length
1	S	1	1	03:30	06:20	170
2	S	2	1	06:20	06:30	10
3	S	3	1	06:30	06:45	15
4	R	–	–	06:45	07:53	68
5	R	–	–	07:53	07:57	4
6	R	–	–	07:57	08:03	6
7	W	–	–	08:03	09:30	87
8	S	78	27	09:30	10:00	30
9	S	79	27	10:00	10:20	20
10	S	80	27	10:20	13:05	165
11	R	–	–	13:05	13:17	12
12	R	–	–	13:17	13:25	8
13	R	–	–	13:25	14:31	66

Item is the number of the event within the shift. *Type* indicates if the row is referred to a shift (S), a legal resting time (R) or a wasted time (W). *Seg_Number* and *Serv_Number* indicates the number of the segment and the service, *Beg_time* and *End_time* are the beginning and ending time of the item and *Length* indicates the quantity of minutes the event last.

This shift was particularly selected because is a worst-case example. It shows the effect of considering too 'soft' rules and how they affect the quality obtained. For example, this shift is not particularly performed during the night or during the morning, but just in the middle. The rules about weekend days gives three days as weekend, but it starts in a quite unusual hour if compared with traditional jobs. A more strict control could just consider this shift as illegal (not feasible) but that would probably give a worse solution. There is also a important quantity of wasted time that connect both spells (87 minutes), which can also be controlled and limited even further, but producing certainly a solution with more drivers as it will be shown. It is important to notice that the structure of the model already gives regularity to the solution (shifts are repeated during one week and at

least, the working hours are kept for a while), but it is important to know how much it is possible to expect if more is asked to the solution in terms of quality.

The problem of the wasted time may be less important than the first one, because its implications are less severe for drivers. The model in any case optimize the quantity of drivers, and because the length of the shift is controlled the worse thing that can happen is to have drivers that had to wait long minutes for a new service to take. Here, again, the possibility of searching for alternative solutions is important, because, if a solution with shorter shifts has the same quantity of drivers, it should be preferred. In that case, to construct shifts using a shortest path approach could help, but certainly it should be controlled to avoid quality problems, such as atomized shifts.

The problem of having shifts which starting and ending times are not 'natural' as working limits is more serious, because as it was explained before (in section 4.3) the limits of the working hours influence many different aspects of the quality of life. One possible solution for this problem can be solved if some reference blocks are considered. For example, if these four blocks are considered:

Time Block	Begin Time	End Time	Length (Hours)
Early	05:00	17:00	12
Morning	07:00	19:00	12
Afternoon	12:00	00:00	12
Night	20:00	09:00	13

What can be done is to consider as feasible only those shifts that are inside these blocks. This strict rule can be also changed to allow a certain quantity of minutes outside these limits. Let's define a parameter called min_out_tb to represent the maximum quantity of minutes outside the best time block possible (the one that has the most of the shift inside). The following table (7.10) shows the results when this parameter is set to three different values: 0, 60, and 120 minutes. A comparison with the free version (the best solution presented before when no limits are considered) is done too. Parameters are kept as they were used to obtain the best solution.

Q indicates the total quantity of drivers of the solution, and % indicate the % of variation with respect to the best solution (value under the column $Free$). As it can be seen, there is a significative difference when such limits are considered, which in some sense may indicate that most of the reduction

Table 7.10: Shifts Limited by Time Blocks - Quantity of Drivers

Instance	Free	0		60		120	
	Q	Q	%	Q	%	Q	%
CHL_288	123	179	31,3	152	19,1	141	12,8
CHL_518	189	281	32,7	235	19,6	210	10,0
ITA_908	94	126	25,4	114	17,5	103	8,7
CHL_790	364	471	22,7	415	12,3	389	6,4
CHL_1086	497	645	22,9	576	13,7	534	6,9
CHL_1392	702	936	25,0	815	13,9	748	6,1
SWI_928	224	289	22,5	250	10,4	239	6,3
ECU_1526	434	568	30,9	510	14,9	469	7,4
MIX_1426	287	415	30,8	349	17,8	309	7,1
MIX_4898	1131	1501	24,7	1301	13,1	1201	5,8

of the quantity of drivers is done using shifts of 'lower' quality (certainly the possibility of obtaining shifts within nice working hours but very fragmented is a alternative).

To obtain the results presented on table 7.3 a limit of 14 hours was considered as the maximum length for a shift. If shorter lengths are establish as maximum, the results are in some degree worst, as it can be seen in the table 7.11.

Table 7.11: Effect of Considering Shorter Shifts

Instance	14-Hours	13-Hours	%	12-Hours	%
CHL_288	123	130	5,69	139	13,01
CHL_518	189	198	4,76	212	12,17
ITA_908	94	110	17,02	120	27,66
CHL_790	364	385	5,77	421	15,66
CHL_1086	497	530	6,64	585	17,71
CHL_1392	702	747	6,41	848	20,80
SWI_928	236	255	8,051	273	15,68
ECU_1526	434	475	9,45	506	16,59
MIX_1426	287	308	7,32	333	16,03
MIX_4898	1131	1217	7,60	1293	14,32

Results are obvious, because shorter the shifts more drivers are needed, but these results seems less severe than those that come, for example, from the limitation of the working hours within time blocks. In any case is another useful tool possible to use to regulate quality.

It was said before that a possibility to achieve a little improvement could come from an adjustment on some depots where the quantity of rosters has not been defined by an exact multiple of the week of work. Two examples will be shown.

The first one comes from the instance ITA_908 where there is one depot with two rosters in this situation:

Table 7.12: Example 1: Original Rosters Depot 16, Instance ITA_908

Roster	Week	Beg Time	End Time	Work Time	Total Time	Weekend	Monthly
Depot 16							
1	1	19:00	08:30	330	810 (13.5)	2	
	2	07:00	17:20	410	620 (10.3)	1	
	3	08:00	20:50	410	770 (12.8)	1	143
2	1	12:00	00:50	460	770 (12.8)	1	
	2	07:00	19:30	410	750 (12.5)	1	
	3	14:00	22:50	360	530 (8.8)	1	153

These two rosters could be converted in just one, but with a week of work having 7 days (because it is the sum of the weekend days of the shifts), as it is indicated in the following tables (7.13 and C.1). Table C.1 on appendix C shows the roster obtained, $(6 + 1) \times 7 = 49$ days long.

Table 7.13: Example 1: Fused Rosters Depot 16, Instance ITA_908

Roster	Week	Beg Time	End Time	Work Time	Total Time	Weekend	Monthly
Depot 16							
1	1	19:00	08:30	330	810 (13.5)	2	
	2	07:00	17:20	410	620 (10.3)	1	
	3	08:00	20:50	410	770 (12.8)	1	
	4	12:00	00:50	460	770 (12.8)	1	
	5	07:00	19:30	410	750 (12.5)	1	
	6	14:00	22:50	360	530 (8.8)	1	170

This first example is about how shifts can be arranged when there are not many rosters in a depot and the efficiency problem is not much. Example 2 shows what can be done when there are several rosters in one depot. This permits to spread many days among many rosters, using sometimes those 'empty' places that were not used when rosters were constructed.

On the solution of the instance CHL_518, on depot 1, the solution has the following rosters:

It is roster 8 the one interesting on this situation, because it is formed by four shifts with just one weekend day. Considering the rest of the rosters, there is just one roster (number 7) that has room for insert a day without change the long of the week of work. The other three days have to be assigned to rosters that already have six weekend days associated, and then it is necessary to use a longer week for those shifts.

The resulting roster will be as indicated in table 7.15, where roster the

Table 7.14: Example 2 - Rosters on depot 1 (Instance CHL_518)

Roster	Week	Beg Time	End Time	Work Time	Total Time	Weekend	Monthly
Depot 1							
1	1	17:35	05:00	395	685 (11.4)	1	
	2	06:10	14:25	400	495 (8.3)	2	
	3	20:50	06:30	405	580 (9.7)	2	
	4	09:45	20:20	390	635 (10.6)	1	159
2	1	06:00	18:15	470	735 (12.3)	2	
	2	18:35	06:15	385	700 (11.7)	2	
	3	13:00	00:00	370	660 (11.0)	1	
	4	10:15	18:00	370	465 (7.8)	1	159
3	1	19:15	06:00	375	645 (10.8)	2	
	2	13:35	01:10	455	695 (11.6)	1	
	3	18:00	05:20	390	680 (11.3)	2	
	4	10:45	19:00	375	495 (8.3)	1	159
4	1	07:25	15:30	375	485 (8.1)	1	
	2	05:00	17:15	455	735 (12.3)	3	
	3	09:00	17:00	375	480 (8.0)	1	
	4	09:30	18:30	385	540 (9.0)	1	159
5	1	09:30	20:30	385	660 (11.0)	1	
	2	09:15	17:30	375	495 (8.3)	1	
	3	08:30	16:00	370	450 (7.5)	1	
	4	08:05	16:30	380	505 (8.4)	1	
	5	18:15	06:00	400	705 (11.8)	2	159
6	1	07:00	18:35	440	695 (11.6)	1	
	2	08:45	20:00	375	675 (11.3)	1	
	3	07:45	15:15	355	450 (7.5)	1	
	4	09:05	17:00	365	475 (7.9)	1	
	5	19:00	05:45	395	645 (10.8)	2	160
7	1	06:40	14:45	395	485 (8.1)	2	
	2	08:30	16:45	415	495 (8.3)	1	
	3	08:05	16:00	380	475 (7.9)	1	
	4	13:00	23:35	395	635 (10.6)	1	158
8	1	17:45	06:15	410	750 (12.5)	1	
	2	09:05	18:00	380	535 (8.9)	1	
	3	08:15	16:25	400	490 (8.2)	1	
	4	11:30	22:30	375	660 (11.0)	1	156
9	1	06:40	15:40	390	540 (9.0)	2	
	2	11:30	20:00	370	510 (8.5)	1	
	3	18:30	06:00	395	690 (11.5)	2	
	4	12:35	01:10	440	755 (12.6)	1	159

original roster 8 is not present anymore, and rosters 1, 2, 7, and 9 of the original rosters have been modified.

These two examples it can be seen how the quantity of average monthly hours is increased when a longer week is considered, but it is not because the week is longer, but because of the particular combinations of shifts reached. In any case, it gives another tool to regulate the fairness among rosters and drivers when this kind of arrangement has to be done. Moreover, as far as the author knows, there is no reference in the literature about the effect the long of the week has on the results for scheduling or rostering. From the model it is possible to see that longer the week, fewer drivers could be expected.

On these rosters it can be seen that the rotation indicated is not the forward rotation. That is because this is the pure output of the solution procedure and this does not affect the possibility of making a forward rotation, because it is just a matter of sorting the shifts in a certain way. It would not change the result.

Table 7.15: Example 2 - Rosters on depot 1 after modification (Instance CHL_518)

Roster	Week	Beg Time	End Time	Work Time	Total Time	Weekend	Monthly
Depot 1							
1	1	17:35	05:00	395	685 (11.4)	1	
	2	06:10	14:25	400	495 (8.3)	2	
	3	20:50	06:30	405	580 (9.7)	2	
	4	09:45	20:20	390	635 (10.6)	1	
*	5	09:05	18:00	380	535 (8.9)	1	164
2	1	06:00	18:15	470	735 (12.3)	2	
	2	18:35	06:15	385	700 (11.7)	2	
	3	13:00	00:00	370	660 (11.0)	1	
	4	10:15	18:00	370	465 (7.8)	1	
*	5	17:45	06:15	410	750 (12.5)	1	167
3	1	19:15	06:00	375	645 (10.8)	2	
	2	13:35	01:10	455	695 (11.6)	1	
	3	18:00	05:20	390	680 (11.3)	2	
	4	10:45	19:00	375	495 (8.3)	1	159
4	1	07:25	15:30	375	485 (8.1)	1	
	2	05:00	17:15	455	735 (12.3)	3	
	3	09:00	17:00	375	480 (8.0)	1	
	4	09:30	18:30	385	540 (9.0)	1	159
5	1	09:30	20:30	385	660 (11.0)	1	
	2	09:15	17:30	375	495 (8.3)	1	
	3	08:30	16:00	370	450 (7.5)	1	
	4	08:05	16:30	380	505 (8.4)	1	
	5	18:15	06:00	400	705 (11.8)	2	159
6	1	07:00	18:35	440	695 (11.6)	1	
	2	08:45	20:00	375	675 (11.3)	1	
	3	07:45	15:15	355	450 (7.5)	1	
	4	09:05	17:00	365	475 (7.9)	1	
	5	19:00	05:45	395	645 (10.8)	2	160
7	1	06:40	14:45	395	485 (8.1)	2	
	2	08:30	16:45	415	495 (8.3)	1	
	3	08:05	16:00	380	475 (7.9)	1	
	4	13:00	23:35	395	635 (10.6)	1	
*	5	11:30	22:30	375	660 (11.0)	1	191
8	1	06:40	15:40	390	540 (9.0)	2	
	2	11:30	20:00	370	510 (8.5)	1	
	3	18:30	06:00	395	690 (11.5)	2	
	4	12:35	01:10	440	755 (12.6)	1	
*	5	08:15	16:25	400	490 (8.2)	1	166

From these results, and some other features not showed here, it is possible to see that, even if the structure of the model limits how rosters have to constructed, it allows some flexibility on the final assignation of the shifts. This characteristic has not only an influence on the mathematical result, but also gives some tools to the decision makers about the management of the system, which usually is not considered when crew scheduling and rostering problems are optimized in traditional approaches.

Chapter 8

Conclusions

Both main objectives of this research were achieved: To obtain a new model that integrates crew scheduling and rostering problems and to obtain results with regular rest patterns and regular working hours.

The single model presented here to solve two problems that were seen as separated so far permits to consider them as one now within the structure offered. This model not only is a tool to solve the problem, but also it offers concepts about how to do the shifts management based on quality. Parameters offer different ways to control quality which could be fundamental to balance economic aspects related to the size of the workforce and the quality of life that can be offered to them.

This model was done is such a way that it is clear the effect decisions have in the final result. To have drivers directly represented permits to have a clear vision about how scheduling and rostering are related, which so far was not known with this clarity, and it allows to control the search for a better solution in a focused manner. The intrinsic regularity of the model, helped by the solution procedure in order to obtain robustness and a better quality of life for drivers has no precedents.

The model by construction proposes a philosophy of management, which considers several aspects that are important for planners in transportation companies, and which are usually hidden or not considered in traditional approaches. Minimization of the drivers is an important issue, but there are many more factors that have to be included in the analysis, which for a manager have to be balanced. A mathematical model is more or less useful if it allows to represent the reality and the requirements of the users in a

good enough manner. This model includes and makes explicit some of those aspects.

The advantage of using a set covering problem model is also evident, because a good percentage of the research on these topics has been done with this model, and to adapt existing approaches to this proposal should not be hard to do. This idea and the existing methods could give a rich source of research, plus the adaptation to other specific rules.

Nevertheless, according to the results showed and given the special non-linear relationship between scheduling and rostering the best future approach could be more oriented to the exploration of the alternative solutions than the optimization of a linear value as how it has been done during almost the whole history of this problem. That is certainly the base, but there is much more from there on. The optimization of the linear relaxation of the problem gives a good start for the search of the best solution, but it does not say when that search is really finished. A two-stages approach, where the first stage gives the best result of the linear relaxation to use as reference (to evaluate new shifts) in a second optimization stage, probably more heuristically oriented, could search for the real best solution.

It has been said in literature that integration of the crew scheduling and rostering problems was a key element to improve the general solution, but these results shows that is not only integration the only element that achieve this goal, but a exploration of the surroundings of the linear solution obtained (and not even the best integer solution obtained), which requires a complete new approach of optimization where a more traditional method as the one used here or as the ones presented in the literature are only a support to the exploration of the possible solutions. This model permits to control the optimization as no other, but it needs more development in order to achieve a better solution procedure that can no only obtain the best possible solution, but also may give alternatives solutions for the decision maker.

The integration achieved here is the first step for having a procedure that integrates also the previous stages of the general transportation problem. But it is not only about integration in order to obtain a better result in terms of quantity of drivers. The quality obtained could be also be improved if the solution methods for both previous problems (timetable and rolling stock) are designed in order to obtain solutions that make easier the construction of

shifts and rosters of good quality. The idea of how regularity and quality is obtained in this approach is clear, and then to traduce those characteristics into conditions for the timetable problem and the rolling stock problem should not be hard to do and give good results.

The inclusion of personal preferences could lead to a creation of the rosters day by day, which would be another kind of post-optimization stage.

Bibliography

[1] A. Caprara, M. Fischetti, P. Toth, D. Vigo, and P.L. Guida. Algorithms for railway crew management. *Mathematical Programming*, (79):125–141, 1997.

[2] A. Caprara, F. Focacci, E. Lamma, P. Mello, M. Milano, P. Toth, , and D. Vigo. Integrating constraint logic programming and operations research techniques for the crew rostering problem. *Software Practice and Experience*, 28(1):49–76, 1998.

[3] A. Caprara, M. Monaci, and P. Toth. *Computer-Aided Transit Scheduling, Lecture Notes in Economics and Mathematical Systems 505*, chapter A Global Method for Crew Planning in Railway Applications, pages 17–36. Springer-Verlag, Berlin, 2001.

[4] L. Cavique, C. Rego, and I. Themido. Subgraph ejection chains and tabu search for the crew scheduling problem. *Journal of the Operational Research Society*, 50:608–616, 1999.

[5] J.F. Cordeau, F. Soumis, and J. Desrosiers. A benders decomposition approach for the locomotive and car assignment problem. *Transportation Science*, 34(2):133–149, May 2000.

[6] J. Daduna and S. Voss, editors. *Computer-Aided Scheduling of Public Transport, Lecture Notes in Economics and Mathematical Systems*, volume 505. Springer Publishers, 2001.

[7] J. Daduna and A. Wren, editors. *Computer-Aided Transit Scheduling, Lecture Notes in Economics and Mathematical Systems*, volume 308. Springer-Verlag, Berlin, 1988.

[8] A. de Silva. Combining constraint programming and linear programming on an example of bus driver scheduling. *Annals of Operations Research*, 108:277–291, 2001.

[9] J.Pinho de Sousa, T Galvo Dias, and J. Falco e Cunha. Interactive multi-objetive genetic algorithms for the bus driver scheduling problem. *Proceedings of the 10th EWGT Meeting and 16th Mini-EURO Conference*, pages 698–705, 2005.

[10] M. Desrochers and J. Rousseau, editors. *Computer-Aided Transit Scheduling, Lecture Notes in Economics and Mathematical Systems*, volume 386. Springer-Verlag, Berlin, 1992.

[11] S. Elias. The use of digital computers in the economic scheduling for both man and machine in public transportation. *Kansas State University Bulletin*, 1964.

[12] A.T. Ernst, H. Jiang, M. Krishnamoorthy, H. Nott, and D. Sier. An integrated optimization model for train crew management. *Annals of Operations Research*, 108:211–224, 2001.

[13] A.T. Ernst, H. Jiang, M. Krishnamoorthy, B. Owens, and D. Sier. An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, 127:21–144, 2004.

[14] A.T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models, journal=European Journal of Operational Research, year=2004, volume=153, pages=3-27.

[15] R. Freling, R. Lentink, and A. Wagelmans. A decision support system for crew planning in passenger transportation using a flexible branch-and-price algorithm. *Annals of Operations Research: Staff Scheduling and Rostering: Theory and Applications, Part I*, 127(1–4):203–222, March 2004.

[16] D. Klabjan, E. L. Johnson, G. L. Nemhauser, E. Gelman, and S. Ramaswamy. Airline crew scheduling with regularity. *Transportation Science*, 35(4):359–374, November 2001.

[17] N. Kohl and S.E. Karisch. Airline crew rostering: Problem types, modeling, and optimization. *Annals of Operations Research*, 127:223–257, 2004.

[18] N. Lingaya, J.F. Cordeau, G. Desaulniers, J. Desrosiers, and F. Soumis. Simultaneous locomotive and car assignment at via rail canada. *Transportation Research, PART B*, 36:755–778, 2002.

[19] H. Loureno, J. Paixo, and R. Portugal. Multiobjective metaheuristics for the bus-driver scheduling problem. *Transportation Science*, 35(3):331–341, 2001.

[20] B. Nicoletti. Automatic crew rostering. *Transportation Science*, 9(1):33–42, 1975.

[21] M.A. Odijk, H.E. Romeijn, and H. van Maaren. Generation of classes of robust periodic railway timetables. *Computers & Operations Research*, 33:2283–2299, 2006.

[22] Samuel J. Raff. Routing and scheduling of vehicles and crews : The state of the art. *Computers & Operation Research*, 10(2):63–67, 1983.

[23] J. Rubin. A technique for the solution of massive set covering problem, with application to airline crew scheduling. *Transportation Science*, 7(1):34–48, 1973.

[24] P.A. Smith, A. Smith L. Di Milia, W. Gee, and R. Mackay. Managing fatigue through roster design. *Proceedings of The Queensland Mining Industry Health and Safety Conference: A New Era in Mine Health and Safety Management*, 2000.

[25] J. Weir. *A Three Phase Approach to Solving the Crew Bidline Problem with Rules for Mitigating Crew Fatigue*. PhD thesis, Georgia Institute of Technology, 2002.

[26] N. Wilson, editor. *Lecture Notes in Economics and Mathematical Systems: Computer-Aided Transit Scheduling*, volume 471. Springer-Verlag, Berlin, 1999.

[27] A. Wren. A review of computer scheduling of buses and crews. *Proceedings of Public Transport Analysis Seminar*, pages 42–47, 1968.

[28] A. Wren, S. Fores, A. Kwan, R. Kwan, M. Parker, and L. Proll. A flexible system for scheduling drivers. *Journal of Scheduling*, 6:437–455, 2003.

Appendix A

Working Loads

Columns Names

- Depot: Number of the depot
- Q_shifts: Quantity of shifts of the roster.
- Q_drivers: Quantity of drivers of the roster.
- Roster_long: Long of the roster.
- Min: Average monthly minutes of the roster.
- Hours: Average monthly hours of the roster.

Instance CHL_288

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
0	4	5	30	9990	166
	4	5	30	10050	167
	3	4	24	10125	168
1	4	5	30	9330	155
	4	5	30	9270	154
	5	6	36	9700	161
	5	6	36	9600	160
	5	6	36	9575	159
	4	5	30	9270	154
2	3	4	24	9375	156
	3	4	24	9412	156
	3	4	24	9187	153
	4	5	30	9600	160
	3	4	24	9075	151
	4	5	30	9600	160
	4	5	30	9630	160
	4	5	30	9570	159
	4	5	30	9540	159
	4	5	30	9660	161
	4	5	30	9630	160
	4	5	30	9570	159
	4	5	30	9600	160
	4	5	30	9540	159
	4	5	30	9540	159
	4	5	30	9750	162

Instance CHL_518

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
0	4	5	30	10500	175
	4	5	30	10530	175
	5	6	36	10475	174
	5	6	36	10500	175
	5	6	36	10525	175
	4	5	30	10560	176
1	4	5	30	9540	159
	4	5	30	9570	159
	4	5	30	9570	159
	4	5	30	9540	159
	5	6	36	9550	159
	5	6	36	9650	160
	4	5	30	9510	158
	4	5	30	9390	156
	4	5	30	9570	159
2	3	4	24	9487	158
	3	4	24	9525	158
	3	4	24	9375	156
	3	4	24	9300	155
	3	4	24	9225	153
	3	4	24	9075	151
	4	5	30	9570	159
	4	5	30	9570	159
	4	5	30	9750	162
	4	5	30	9630	160
	4	5	30	9600	160
	4	5	30	9810	163
	4	5	30	9720	162
	4	5	30	9570	159
	4	5	30	9570	159
	4	5	30	9600	160
	4	5	30	9570	159
3	3	4	24	8137	135
	3	4	24	6600	110
	3	4	24	6600	110
	2	3	18	6300	105
4	4	5	30	6600	110
5	4	5	30	10440	174
	4	5	30	10440	174

Instance ITA_908

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
0	5	6	36	6750	112
	5	6	36	6600	110
4	3	4	24	10050	167
	4	5	30	10080	168
	4	5	30	10080	168
	5	6	36	10150	169
	4	5	30	10020	167
	4	5	30	9960	166
7	5	6	36	8900	148
	4	5	30	9450	157
16	3	4	24	8625	143
	3	4	24	9225	153
22	4	5	30	4800	80
23	4	5	30	9360	156
25	4	5	30	8160	136
	3	4	24	6975	116
30	4	5	30	6000	100
33	4	5	30	9600	160
	3	4	24	9450	157

Instance CHL_790

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
0	3	4	24	9562	159
	4	5	30	10170	169
	4	5	30	10200	170
	4	5	30	10290	171
	4	5	30	10290	171
	4	5	30	10350	172
	4	5	30	10200	170
	4	5	30	10140	169
	4	5	30	10230	170
1	4	5	30	9450	157
	4	5	30	9420	157
	4	5	30	9420	157
	4	5	30	9390	156
	5	6	36	9525	158
	5	6	36	9600	160
	4	5	30	9390	156
	4	5	30	9420	157
	5	6	36	9475	157
	5	6	36	9425	157
	5	6	36	9450	157
	4	5	30	9210	153
2	3	4	24	9300	155
	3	4	24	9525	158
	3	4	24	9412	156
	3	4	24	9300	155
	3	4	24	9225	153
	3	4	24	9450	157
	4	5	30	9510	158
	4	5	30	9360	156
	4	5	30	9540	159
	4	5	30	9390	156
	4	5	30	9630	160
	4	5	30	9600	160
	4	5	30	9510	158
	4	5	30	9510	158
	4	5	30	9450	157
	4	5	30	9660	161
	4	5	30	9630	160
	5	6	36	9350	155
	5	6	36	9325	155
	4	5	30	9480	158
	4	5	30	9420	157
3	3	4	24	6600	110
	4	5	30	6600	110
	3	4	24	6600	110
	3	4	24	6487	108
4	1	2	12	2700	45
5	4	5	30	8820	147
	4	5	30	9000	150
	5	6	36	8625	143
	4	5	30	9000	150
	4	5	30	8940	149
	4	5	30	9540	159
6	2	3	18	8100	135
	3	4	24	9675	161
	3	4	24	9150	152
	3	4	24	9900	165
	3	4	24	9900	165
	3	4	24	9900	165
	3	4	24	8625	143
	3	4	24	9300	155
	4	5	30	8520	142
	4	5	30	8460	141
	4	5	30	9180	153
	3	4	24	8100	135
7	4	5	30	7080	118
	3	4	24	6750	112
	3	4	24	6300	105
8	3	4	24	6750	112
	4	5	30	6630	110
	4	5	30	7710	128
	4	5	30	6600	110
	3	4	24	7200	120
9	4	5	30	6450	107
	4	5	30	6450	107
	3	4	24	6562	109
	3	4	24	6637	110

Instance CHL_1086

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
0	4	5	30	10080	168
	4	5	30	10260	171
	4	5	30	10110	168
	5	6	36	10125	168
	5	6	36	10250	170
	5	6	36	10225	170
	5	6	36	10225	170
	5	6	36	10225	170
1	3	4	24	9262	154
	3	4	24	9112	151
	3	4	24	9112	151
	3	4	24	8850	147
	4	5	30	9660	161
	4	5	30	9780	163
	5	6	36	9275	154
	5	6	36	9975	166
	5	6	36	9775	162
	5	6	36	9725	162
	5	6	36	9650	160
	5	6	36	9550	159
	5	6	36	9450	157
	5	6	36	9100	151
	5	6	36	9000	150
2	3	4	24	9262	154
	3	4	24	9600	160
	3	4	24	9487	158
	3	4	24	9337	155
	3	4	24	9300	155
	3	4	24	9187	153
	3	4	24	9150	152
	3	4	24	9150	152
	3	4	24	9450	157
	4	5	30	9360	156
	4	5	30	9570	159
	4	5	30	9540	159
	4	5	30	9600	160
	4	5	30	9840	164
	4	5	30	9660	161
	4	5	30	10020	167
	4	5	30	9900	165
	4	5	30	9690	161
	5	6	36	9750	162
	5	6	36	9750	162
	5	6	36	9375	156
	5	6	36	9300	155
3	3	4	24	8437	140
	3	4	24	9750	162
	4	5	30	9000	150
	4	5	30	9450	157
4	1	2	12	2700	45
5	4	5	30	9660	161
	4	5	30	9540	159
	5	6	36	9150	152
	4	5	30	9360	156
	4	5	30	8940	149
	4	5	30	8820	147
6	3	4	24	9900	165
	3	4	24	9075	151
	3	4	24	9225	153
	3	4	24	8400	140
	3	4	24	8625	143
	3	4	24	8325	138
	3	4	24	8325	138
	3	4	24	8550	142
	3	4	24	8475	141
	3	4	24	8175	136
	3	4	24	8400	140
	3	4	24	8175	136
	3	4	24	9600	160
	2	3	18	10200	170
7	4	5	30	7080	118
	4	5	30	7080	118
8	3	4	24	6637	110
	4	5	30	6540	109
	3	4	24	5887	98
	3	4	24	6637	110
	3	4	24	6637	110

Instance CHL_1086 (Cont.)

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
9	4	5	30	6960	116
	4	5	30	7080	118
	4	5	30	7080	118
	4	5	30	6540	109
10	3	4	24	10425	173
	3	4	24	11475	191
	3	4	24	11025	183
	3	4	24	10875	181
	3	4	24	10875	181
	4	5	30	10410	173
	3	4	24	10087	168
	3	4	24	10650	177
11	3	4	24	11100	185
	3	4	24	11175	186
	3	4	24	11100	185
	3	4	24	11100	185
	3	4	24	11250	187
	4	5	30	11220	187
	4	5	30	10860	181
	4	5	30	10800	180
	4	5	30	10740	179
12	3	4	24	7650	127
	3	4	24	7200	120
	3	4	24	7425	123
	3	4	24	6525	108
	3	4	24	6975	116
	3	4	24	6525	108
	3	4	24	8325	138
	2	3	18	7500	125
13	5	6	36	6000	100

Instance CHL_1392

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
0	4	5	30	10200	170
	4	5	30	10170	169
	4	5	30	9960	166
	5	6	36	10200	170
	5	6	36	10150	169
	5	6	36	10250	170
	5	6	36	10150	169
	5	6	36	10250	170
1	3	4	24	9262	154
	4	5	30	9480	158
	4	5	30	9450	157
	5	6	36	9700	161
	4	5	30	9600	160
	5	6	36	9700	161
	5	6	36	9675	161
	4	5	30	9510	158
	5	6	36	9600	160
	5	6	36	9600	160
	5	6	36	9650	160
	5	6	36	9600	160
	5	6	36	9675	161
	5	6	36	9650	160
	5	6	36	9625	160
2	3	4	24	9600	160
	3	4	24	9412	156
	3	4	24	9262	154
	3	4	24	9225	153
	3	4	24	9112	151
	3	4	24	9075	151
	4	5	30	9480	158
	4	5	30	9630	160
	4	5	30	9720	162
	4	5	30	9600	160
	4	5	30	10020	167
	4	5	30	9900	165
	4	5	30	9960	166
	4	5	30	9810	163
	5	6	36	9675	161
	5	6	36	9350	155
	4	5	30	9570	159
	4	5	30	9540	159
	4	5	30	9510	158
	4	5	30	9480	158
	4	5	30	9480	158

Instance CHL_1392 (Cont.)

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
3	3	4	24	8062	134
	3	4	24	7050	117
	4	5	30	7140	119
	4	5	30	7080	118
	3	4	24	8325	138
4	2	3	18	8100	135
5	3	4	24	8775	146
	4	5	30	9180	153
	4	5	30	9360	156
	4	5	30	9180	153
	4	5	30	9000	150
	4	5	30	9360	156
	4	5	30	9300	155
6	3	4	24	9450	157
	3	4	24	9150	152
	3	4	24	8850	147
	3	4	24	8325	138
	3	4	24	8175	136
	4	5	30	8460	141
	4	5	30	8340	139
	4	5	30	9240	154
	3	4	24	8475	141
	3	4	24	9750	162
	3	4	24	9750	162
	3	4	24	9975	166
7	4	5	30	7140	119
	4	5	30	7140	119
8	3	4	24	6900	115
	4	5	30	6600	110
	4	5	30	6810	113
	4	5	30	6450	107
	3	4	24	6787	113
9	4	5	30	7080	118
	4	5	30	6960	116
	3	4	24	7275	121
	3	4	24	6637	110
10	3	4	24	10087	168
	3	4	24	10875	181
	3	4	24	10650	177
	3	4	24	10425	173
	3	4	24	10425	173
	5	6	36	10175	169
	4	5	30	10860	181
	4	5	30	10860	181
	4	5	30	10680	178
11	2	3	18	10600	176
	3	4	24	10725	178
	3	4	24	10275	171
	3	4	24	10350	172
	3	4	24	10875	181
	3	4	24	10875	181
	4	5	30	10800	180
	4	5	30	10800	180
	4	5	30	11040	184
	4	5	30	10680	178
	3	4	24	10875	181
	3	4	24	10875	181
	3	4	24	10875	181
	3	4	24	10875	181
	3	4	24	10875	181
12	3	4	24	7425	123
	3	4	24	7200	120
	3	4	24	8100	135
	3	4	24	7200	120
	4	5	30	6660	111
	4	5	30	8100	135
	4	5	30	7560	126
	4	5	30	7380	123
	3	4	24	7425	123
13	3	4	24	6075	101
	3	4	24	6075	101
	2	3	18	7200	120
14	3	4	24	8100	135
	3	4	24	6750	112
	3	4	24	6300	105
	3	4	24	7200	120
	2	3	18	7800	130

Instance CHL_1392 (Cont.)

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
15	3	4	24	7875	131
	3	4	24	8100	135
	3	4	24	7425	123
	3	4	24	8100	135
	3	4	24	8100	135
	4	5	30	8460	141
	4	5	30	8280	138
	4	5	30	8280	138
	4	5	30	8280	138
	4	5	30	8010	133
	4	5	30	7650	127
	4	5	30	8460	141
16	3	4	24	7875	131
	3	4	24	7875	131
	3	4	24	7875	131
	3	4	24	7650	127
	3	4	24	7650	127
17	3	4	24	10125	168
	2	3	18	10800	180
	4	5	30	11340	189
	3	4	24	10800	180
	2	3	18	9000	150
18	3	4	24	10125	168
	4	5	30	9720	162
	4	5	30	9720	162
	4	5	30	8100	135
19	5	6	36	9750	162
	5	6	36	9750	162
	5	6	36	9750	162
	5	6	36	9750	162

Instance SWI_928

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
0	4	5	30	9510	158
	4	5	30	9420	157
	4	5	30	9480	158
	4	5	30	9360	156
	3	4	24	10012	166
1	5	6	36	9375	156
	5	6	36	9350	155
	4	5	30	9480	158
2	4	5	30	9120	152
	5	6	36	9150	152
	4	5	30	9090	151
3	4	5	30	8520	142
	4	5	30	8340	139
	4	5	30	9180	153
4	1	2	12	4800	80
5	3	4	24	8700	145
	3	4	24	8362	139
	4	5	30	8340	139
	4	5	30	8370	139
	4	5	30	9000	150
	4	5	30	8310	138
	3	4	24	8925	148
6	3	4	24	6225	103
	3	4	24	4950	82
7	2	3	18	8400	140
8	4	5	30	9240	154
	4	5	30	9060	151
10	4	5	30	8670	144
	3	4	24	8775	146
11	4	5	30	4740	79
	4	5	30	4560	76
12	5	6	36	9050	150
	4	5	30	7860	131
13	4	5	30	8760	146
15	4	5	30	6000	100

Instance SWI_928 (Cont.)

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
17	4	5	30	8460	141
	4	5	30	8700	145
	4	5	30	8700	145
	4	5	30	8340	139
	4	5	30	8400	140
	4	5	30	8760	146
	4	5	30	8700	145
	5	6	36	8300	138
	5	6	36	8450	140
	4	5	30	8430	140
	4	5	30	8460	141

Instance ECU_1526

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
0	3	4	24	8025	133
	3	4	24	8625	143
	3	4	24	7725	128
	3	4	24	8250	137
1	2	3	18	9600	160
	3	4	24	9900	165
	3	4	24	10200	170
	3	4	24	10125	168
	3	4	24	10200	170
	2	3	18	9900	165
	2	3	18	9800	163
3	3	4	24	5775	96
	3	4	24	6637	110
	3	4	24	6637	110
	3	4	24	5137	85
	4	5	30	5730	95
5	3	4	24	9375	156
	3	4	24	9375	156
	3	4	24	9975	166
	3	4	24	9375	156
	2	3	18	8600	143
6	5	6	36	2800	46
7	4	5	30	8190	136
	4	5	30	8190	136
	4	5	30	8880	148
8	3	4	24	8962	149
	3	4	24	8400	140
	4	5	30	8250	137
	5	6	36	8500	141
	4	5	30	8880	148
	4	5	30	8490	141
	4	5	30	8100	135
	4	5	30	8280	138
9	3	4	24	6150	102
	3	4	24	6075	101
	3	4	24	6300	105
10	3	4	24	9450	157
	3	4	24	9450	157
11	3	4	24	7762	129
	4	5	30	7680	128
	4	5	30	8040	134
12	3	4	24	9000	150
	2	3	18	9700	161
	3	4	24	9450	157
13	3	4	24	7125	118
	3	4	24	7275	121
	3	4	24	7650	127
	3	4	24	6825	113
	4	5	30	7440	124
14	3	4	24	5512	91
15	3	4	24	9300	155
	4	5	30	9120	152
	3	4	24	9000	150
18	3	4	24	7725	128
	4	5	30	6720	112
	4	5	30	6210	103
	5	6	36	6075	101
20	4	5	30	6750	112
	4	5	30	6420	107
	4	5	30	7500	125
21	4	5	30	9420	157
	4	5	30	8880	148

Instance MIX_1526 (Cont.)

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
22	3	4	24	8850	147
	3	4	24	7500	125
	2	3	18	8000	133
23	3	4	24	7950	132
	3	4	24	8400	140
	3	4	24	7575	126
	3	4	24	8550	142
	3	4	24	8325	138
24	1	2	12	2250	37
25	4	5	30	8160	136
	3	4	24	9675	161
	3	4	24	9600	160
	3	4	24	9450	157
	3	4	24	9300	155
	3	4	24	7875	131
	4	5	30	9000	150
	4	5	30	8700	145
	3	4	24	9600	160
	5	6	36	9650	160
	5	6	36	9500	158
	4	5	30	9360	156
	4	5	30	9300	155
	4	5	30	9240	154
26	4	5	30	9780	163
	4	5	30	10320	172
	5	6	36	9750	162
	4	5	30	9600	160
27	4	5	30	6900	115
	3	4	24	8062	134
	3	4	24	8100	135
28	3	4	24	8850	147
	4	5	30	7740	129
29	3	4	24	7200	120
30	3	4	24	8775	146
	3	4	24	8325	138
	3	4	24	8325	138
31	3	4	24	10350	172
	2	3	18	9400	156

Instance MIX_1426

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
0	5	6	36	8325	138
	5	6	36	8350	139
4	4	5	30	10530	175
	4	5	30	10320	172
	4	5	30	10380	173
	4	5	30	10500	175
	4	5	30	10560	176
	5	6	36	10475	174
	4	5	30	10470	174
	4	5	30	10380	173
	5	6	36	10275	171
	5	6	36	10575	176
	4	5	30	10110	168
	4	5	30	10020	167
7	4	5	30	8850	147
	5	6	36	8800	146
16	4	5	30	7980	133
	4	5	30	7680	128
22	6	7	42	9771	162
	6	7	42	9664	161
23	5	6	36	9100	151
25	4	5	30	7440	124
	4	5	30	7440	124
30	4	5	30	6000	100
33	4	5	30	9300	155
	3	4	24	9150	152
34	3	4	24	9075	151
	5	6	36	9275	154
	5	6	36	9050	150
	4	5	30	9210	153
	4	5	30	9240	154
	4	5	30	9120	152
	4	5	30	9120	152

Instance MIX_1426 (Cont.)

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
35	4	5	30	9630	160
	3	4	24	9262	154
	3	4	24	9375	156
	3	4	24	9375	156
	3	4	24	9300	155
	3	4	24	9262	154
	3	4	24	9187	153
	3	4	24	9337	155
	3	4	24	8925	148
	3	4	24	8925	148
	3	4	24	9375	156
	4	5	30	9570	159
	4	5	30	9510	158
	4	5	30	9510	158
	4	5	30	9510	158
	4	5	30	9450	157
	4	5	30	9450	157
	4	5	30	9450	157
	4	5	30	9450	157
	4	5	30	9660	161
	4	5	30	9720	162
37	3	4	24	8250	137
	3	4	24	6600	110
	3	4	24	6600	110
	3	4	24	6600	110

Instance MIX_4898

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
0	3	4	24	8625	143
	3	4	24	8850	147
	4	5	30	8520	142
	3	4	24	8625	143
	4	5	30	8610	143
	3	4	24	8775	146
4	4	5	30	10140	169
	4	5	30	10170	169
	4	5	30	10110	168
	4	5	30	10200	170
	4	5	30	10140	169
	4	5	30	10200	170
	5	6	36	10250	170
	4	5	30	10200	170
	4	5	30	10440	174
	5	6	36	10450	174
	5	6	36	10025	167
	5	6	36	9875	164
	5	6	36	9825	163
	5	6	36	9900	165
	5	6	36	10125	168
	4	5	30	9840	164
5	2	3	18	4200	70
7	5	6	36	8950	149
	5	6	36	8950	149
10	5	6	36	6250	104
	4	5	30	6630	110
	5	6	36	6725	112
	4	5	30	6960	116
16	4	5	30	9000	150
	5	6	36	8800	146
	5	6	36	8700	145
	5	6	36	8000	133
	4	5	30	8220	137
20	2	3	18	5500	91
21	2	3	18	10000	166
	2	3	18	8800	146
22	5	6	36	10975	182
	5	6	36	10500	175
23	3	4	24	9900	165
	3	4	24	10650	177
25	3	4	24	7800	130
	4	5	30	6720	112
27	1	2	12	3000	50
30	4	5	30	6000	100
33	3	4	24	10050	167
	3	4	24	8175	136

Instance MIX_4898 (Cont.)

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
34	4	5	30	9450	157
	4	5	30	9420	157
	4	5	30	9420	157
	5	6	36	9550	159
	4	5	30	9570	159
	5	6	36	9300	155
	4	5	30	9330	155
	4	5	30	9330	155
	4	5	30	9390	156
35	3	4	24	9375	156
	3	4	24	9225	153
	3	4	24	9000	150
	3	4	24	9000	150
	3	4	24	8812	146
	4	5	30	9690	161
	4	5	30	9780	163
	4	5	30	9930	165
	4	5	30	9780	163
	4	5	30	9570	159
	4	5	30	9480	158
	4	5	30	9600	160
	4	5	30	9600	160
	4	5	30	9480	158
	4	5	30	9510	158
	4	5	30	9480	158
	3	4	24	8925	148
37	3	4	24	6600	110
	4	5	30	7920	132
	3	4	24	6600	110
	3	4	24	6600	110
38	3	4	24	7875	131
	3	4	24	7575	126
	3	4	24	6600	110
	3	4	24	8175	136
39	2	3	18	9100	151
	2	3	18	8700	145
	3	4	24	9825	163
	3	4	24	9525	158
	3	4	24	9525	158
	4	5	30	9960	166
	3	4	24	9000	150
41	4	5	30	4680	78
	3	4	24	5850	97
	3	4	24	6637	110
	3	4	24	5137	85
	4	5	30	4680	78
	3	4	24	5775	96
42	2	3	18	5800	96
43	3	4	24	9375	156
	3	4	24	9375	156
	3	4	24	9375	156
	3	4	24	9750	162
44	4	5	30	3840	64
45	3	4	24	10350	172
	4	5	30	10020	167
	4	5	30	10050	167
	4	5	30	9780	163
	4	5	30	10560	176
	4	5	30	9810	163
46	3	4	24	8925	148
	4	5	30	8640	144
	4	5	30	8640	144
	4	5	30	8820	147
	5	6	36	9050	150
	5	6	36	9000	150
	5	6	36	8950	149
47	3	4	24	7050	117
	4	5	30	7860	131
	4	5	30	7440	124
48	3	4	24	9450	157
	3	4	24	9000	150
49	4	5	30	7680	128
	3	4	24	7200	120
50	3	4	24	8100	135
	3	4	24	8175	136
	2	3	18	9500	158
51	3	4	24	6825	113
	3	4	24	6675	111
	3	4	24	7125	118
	3	4	24	7125	118
	2	3	18	6600	110
52	2	3	18	5800	96

Instance MIX_4898 (Cont.)

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
53	3	4	24	6900	115
	3	4	24	9000	150
	3	4	24	7800	130
54	3	4	24	8850	147
56	4	5	30	6720	112
	4	5	30	6720	112
	4	5	30	6720	112
	3	4	24	7050	117
58	4	5	30	8520	142
	4	5	30	7920	132
59	4	5	30	9120	152
	4	5	30	8880	148
	3	4	24	8400	140
60	3	4	24	7425	123
	2	3	18	5900	98
61	4	5	30	8580	143
	3	4	24	9075	151
	3	4	24	9600	160
	4	5	30	8700	145
62	1	2	12	4425	73
63	3	4	24	7800	130
	4	5	30	7920	132
	4	5	30	7920	132
	3	4	24	8325	138
	4	5	30	8220	137
	4	5	30	8280	138
	4	5	30	8220	137
	4	5	30	8130	135
	4	5	30	8160	136
	4	5	30	8220	137
	4	5	30	8160	136
	4	5	30	8040	134
	5	6	36	7950	132
	5	6	36	7950	132
	4	5	30	8070	134
	4	5	30	8010	133
64	4	5	30	9600	160
	4	5	30	9420	157
	4	5	30	9840	164
	5	6	36	9000	150
	4	5	30	9480	158
65	4	5	30	7080	118
	4	5	30	6900	115
	3	4	24	7875	131
66	4	5	30	6840	114
	4	5	30	6960	116
67	3	4	24	7200	120
	4	5	30	7200	120
68	3	4	24	9225	153
	3	4	24	8325	138
	3	4	24	8325	138
69	3	4	24	10800	180
	2	3	18	9400	156
70	3	4	24	8625	143
	4	5	30	8760	146
	4	5	30	8700	145
	4	5	30	9240	154
	5	6	36	8250	137
	5	6	36	8450	140
	4	5	30	8280	138
71	3	4	24	7050	117
	4	5	30	6960	116
	4	5	30	6960	116
	4	5	30	7440	124
	3	4	24	6750	112
72	5	6	36	8700	145
	4	5	30	8100	135
73	4	5	30	9480	158
	4	5	30	9540	159
	4	5	30	8520	142

Instance MIX_4898 (Cont.)

Depot	Q_shifts	Q_drivers	Roster_long	Min	Hours
75	3	4	24	9600	160
	4	5	30	9600	160
	4	5	30	9120	152
	4	5	30	9120	152
	4	5	30	9120	152
	5	6	36	9000	150
76	4	5	30	6300	105
	3	4	24	6225	103
77	4	5	30	9300	155
78	4	5	30	9240	154
	5	6	36	9400	156
80	4	5	30	7980	133
	4	5	30	8160	136
81	3	4	24	6375	106
	3	4	24	6000	100
	3	4	24	6187	103
	4	5	30	4920	82
82	5	6	36	9200	153
	4	5	30	10680	178
83	4	5	30	8010	133
	4	5	30	7980	133
	3	4	24	8062	134
85	3	4	24	5550	92
	3	4	24	4500	75
87	4	5	30	9600	160
	4	5	30	9600	160
	4	5	30	9360	156
	5	6	36	9550	159
	4	5	30	9360	156
	4	5	30	9480	158
	5	6	36	9550	159
	4	5	30	9780	163
	5	6	36	9575	159
	4	5	30	9540	159
	5	6	36	9450	157
	5	6	36	9500	158
	4	5	30	9540	159
	4	5	30	9540	159
	4	5	30	9600	160
	4	5	30	9540	159
	4	5	30	9540	159

Appendix B

Working with time Blocks

This Appendix shows the complete solutions for the instance CHL_288 when time blocks are considered.

CHL_288 - No Time Block Considered

Roster	Week	Beg Time	End Time	Work Time	Total Time	Weekend	Monthly
Depot 0							
1	1	22:20	06:40	405	500 (8.3)	3	
	2	12:15	23:45	490	690 (11.5)	1	
	3	15:00	23:45	395	525 (8.8)	1	
	4	15:00	23:00	375	480 (8.0)	1	166
2	1	21:45	09:45	480	720 (12.0)	2	
	2	08:00	16:15	405	495 (8.3)	1	
	3	18:20	06:00	375	700 (11.7)	2	
	4	13:00	22:30	415	570 (9.5)	1	167
3	1	12:15	21:50	470	575 (9.6)	1	
	2	13:30	00:00	400	630 (10.5)	1	
	3	07:30	18:45	480	675 (11.3)	1	168
Depot 1							
1	1	06:10	15:40	400	570 (9.5)	2	
	2	19:15	06:00	375	645 (10.8)	2	
	3	07:55	16:30	405	515 (8.6)	1	
	4	10:00	22:15	375	735 (12.3)	1	155
2	1	18:30	06:00	395	690 (11.5)	2	
	2	18:00	06:15	375	735 (12.3)	2	
	3	08:15	16:25	400	490 (8.2)	1	
	4	07:25	15:30	375	485 (8.1)	1	154
3	1	19:00	06:00	390	660 (11.0)	2	
	2	17:45	06:00	415	735 (12.3)	1	
	3	10:45	19:30	390	525 (8.8)	1	
	4	09:15	17:00	375	465 (7.8)	1	
	5	12:30	00:00	370	690 (11.5)	1	161
4	1	06:40	16:00	380	560 (9.3)	2	
	2	13:00	21:45	415	525 (8.8)	1	
	3	17:15	05:00	390	705 (11.8)	1	
	4	12:45	20:30	375	465 (7.8)	1	
	5	09:00	17:00	360	480 (8.0)	1	160
5	1	06:40	14:30	380	470 (7.8)	2	
	2	08:30	16:45	415	495 (8.3)	1	
	3	11:30	20:20	390	530 (8.8)	1	
	4	09:45	17:30	375	465 (7.8)	1	
	5	07:45	15:15	355	450 (7.5)	1	159
6	1	20:50	06:15	375	565 (9.4)	2	
	2	16:45	05:20	415	755 (12.6)	1	
	3	09:30	21:00	385	690 (11.5)	1	
	4	08:30	16:00	370	450 (7.5)	1	154

Roster	Week	Beg Time	End Time	Work Time	Total Time	Weekend	Monthly
Depot 2							
1	1	06:30	15:00	395	510 (8.5)	2	
	2	15:15	02:45	410	690 (11.5)	1	
	3	03:00	12:00	445	540 (9.0)	3	156
2	1	19:05	03:25	390	500 (8.3)	2	
	2	09:00	18:05	410	545 (9.1)	1	
	3	02:55	13:15	455	620 (10.3)	3	156
3	1	19:00	03:20	385	500 (8.3)	2	
	2	10:50	20:30	405	580 (9.7)	1	
	3	03:20	12:35	435	555 (9.3)	3	153
4	1	18:30	03:00	380	510 (8.5)	2	
	2	12:15	20:30	405	495 (8.3)	1	
	3	16:15	02:25	400	610 (10.2)	1	
	4	18:30	03:30	415	540 (9.0)	2	160
5	1	02:45	15:05	425	740 (12.3)	3	
	2	19:30	03:30	380	480 (8.0)	2	
	3	16:30	03:45	405	675 (11.3)	1	151
6	1	20:30	03:55	360	445 (7.4)	2	
	2	15:05	02:30	355	685 (11.4)	1	
	3	14:45	02:55	490	730 (12.2)	1	
	4	18:10	02:45	395	515 (8.6)	2	160
7	1	07:00	18:30	500	690 (11.5)	1	
	2	03:45	12:00	390	495 (8.3)	3	
	3	15:00	03:05	365	725 (12.1)	1	
	4	16:30	03:20	350	650 (10.8)	1	160
8	1	03:30	13:05	410	575 (9.6)	3	
	2	12:00	19:55	400	475 (7.9)	1	
	3	07:20	15:15	380	475 (7.9)	1	
	4	15:00	02:45	405	705 (11.8)	1	159
9	1	02:55	12:55	400	600 (10.0)	3	
	2	17:05	02:45	400	580 (9.7)	1	
	3	17:00	03:35	375	635 (10.6)	1	
	4	15:45	03:15	415	690 (11.5)	1	159
10	1	03:30	16:15	400	765 (12.8)	3	
	2	10:00	21:05	435	665 (11.1)	1	
	3	10:00	18:10	400	490 (8.2)	1	
	4	08:00	16:15	375	495 (8.3)	1	161
11	1	02:25	14:45	395	740 (12.3)	3	
	2	16:30	02:40	435	610 (10.2)	1	
	3	09:55	18:10	400	495 (8.3)	1	
	4	09:00	17:15	375	495 (8.3)	1	160
12	1	03:00	12:15	390	555 (9.3)	3	
	2	14:45	03:00	435	735 (12.3)	1	
	3	09:00	17:05	395	485 (8.1)	1	
	4	12:00	20:15	375	495 (8.3)	1	159
13	1	03:15	13:30	390	615 (10.3)	3	
	2	11:00	19:05	395	485 (8.1)	1	
	3	16:00	03:15	375	675 (11.3)	1	
	4	11:45	21:05	440	560 (9.3)	1	160
14	1	11:30	20:15	390	525 (8.8)	1	
	2	10:30	18:10	370	460 (7.7)	1	
	3	03:20	14:40	420	680 (11.3)	3	
	4	17:15	03:30	410	615 (10.3)	1	159
15	1	11:15	19:15	385	480 (8.0)	1	
	2	15:30	02:55	370	685 (11.4)	1	
	3	03:15	14:15	420	660 (11.0)	3	
	4	15:00	03:10	415	730 (12.2)	1	159
16	1	10:50	19:00	385	490 (8.2)	1	
	2	10:40	19:00	410	500 (8.3)	1	
	3	11:00	18:40	385	460 (7.7)	1	
	4	03:05	13:05	445	600 (10.0)	3	162

CHL_288 - Time Blocks Considered ($min\_out\_tb = 0$)

Roster	Week	Beg Time	End Time	Work Time	Total Time	Weekend	Monthly
Depot 0							
1	1	22:15	07:10	435	535 (8.9)	3	
	2	02:30	06:00	210	210 (3.5)	3	107
2	1	22:20	06:45	415	505 (8.4)	3	
	2	02:45	06:15	210	210 (3.5)	3	104
3	1	23:00	08:45	395	585 (9.8)	3	
	2	23:00	06:25	370	445 (7.4)	3	127
4	1	23:20	07:00	385	460 (7.7)	3	
	2	23:00	06:15	365	435 (7.3)	3	125
5	1	23:30	15:45	175	975 (16.3)	3	
	2	09:00	18:30	465	570 (9.5)	1	
	3	11:00	14:30	210	210 (3.5)	1	
	4	13:10	22:30	455	560 (9.3)	1	130
6	1	23:20	07:10	380	470 (7.8)	3	
	2	07:30	18:00	405	630 (10.5)	1	
	3	15:45	19:45	240	240 (4.0)	1	
	4	10:40	18:00	290	440 (7.3)	1	131
7	1	16:30	20:20	230	230 (3.8)	1	
	2	23:45	16:30	205	1005 (16.8)	3	
	3	08:00	18:00	390	600 (10.0)	1	
	4	08:00	18:45	480	645 (10.8)	1	130
8	1	15:00	23:45	395	525 (8.8)	1	
	2	11:00	02:45	220	945 (15.8)	1	
	3	22:15	10:40	250	745 (12.4)	3	
	4	13:25	23:00	415	575 (9.6)	1	128
9	1	23:00	06:40	365	460 (7.7)	3	
	2	23:00	07:00	385	480 (8.0)	3	125
10	1	23:00	06:40	350	460 (7.7)	3	
	2	16:30	20:00	210	210 (3.5)	1	
	3	15:00	23:45	395	525 (8.8)	1	
	4	16:00	23:45	340	465 (7.8)	1	129
11	1	14:30	23:45	370	555 (9.3)	1	
	2	09:30	18:20	195	530 (8.8)	1	
	3	23:45	17:15	195	1050 (17.5)	3	
	4	12:15	21:50	470	575 (9.6)	1	123
12	1	22:45	11:00	250	735 (12.3)	3	
	2	09:00	18:30	410	570 (9.5)	1	
	3	17:15	20:30	195	195 (3.3)	1	
	4	10:20	18:45	405	505 (8.4)	1	126
13	1	23:10	16:30	215	1040 (17.3)	3	
	2	12:15	21:00	410	525 (8.8)	1	
	3	11:00	02:30	195	930 (15.5)	1	
	4	12:20	23:50	485	690 (11.5)	1	130

Roster	Week	Beg Time	End Time	Work Time	Total Time	Weekend	Monthly
Depot 1							
1	1	20:50	05:00	405	490 (8.2)	2	
	2	09:30	18:30	385	540 (9.0)	1	
	3	00:00	17:30	185	1050 (17.5)	3	121
2	1	23:00	11:00	225	720 (12.0)	3	
	2	20:50	06:00	405	550 (9.2)	2	
	3	10:45	18:30	375	465 (7.8)	1	125
3	1	23:50	17:00	215	1030 (17.2)	3	
	2	20:50	06:00	405	550 (9.2)	2	
	3	07:25	15:30	375	485 (8.1)	1	124
4	1	23:55	18:05	205	1090 (18.2)	3	
	2	20:50	06:00	405	550 (9.2)	2	
	3	12:45	22:00	375	555 (9.3)	1	123
5	1	00:10	16:00	200	950 (15.8)	3	
	2	06:10	14:25	400	495 (8.3)	2	
	3	08:30	16:30	375	480 (8.0)	1	121
6	1	23:40	19:00	200	1160 (19.3)	3	
	2	20:50	06:15	395	565 (9.4)	2	
	3	08:15	16:30	375	495 (8.3)	1	121
7	1	00:20	17:05	195	1005 (16.8)	3	
	2	06:40	14:45	395	485 (8.1)	2	
	3	08:45	17:00	375	495 (8.3)	1	120
8	1	00:00	18:30	195	1110 (18.5)	3	
	2	06:40	14:30	380	470 (7.8)	2	
	3	09:00	18:00	375	540 (9.0)	1	118
9	1	23:45	17:05	190	1040 (17.3)	3	
	2	20:50	06:15	375	565 (9.4)	2	
	3	09:45	18:30	375	525 (8.8)	1	117
10	1	18:05	21:45	220	220 (3.7)	2	
	2	09:15	18:30	375	555 (9.3)	1	
	3	23:30	07:00	375	450 (7.5)	3	121
11	1	20:50	05:20	410	510 (8.5)	2	
	2	18:30	21:45	195	195 (3.3)	2	
	3	12:00	21:00	375	540 (9.0)	1	
	4	10:00	18:00	360	480 (8.0)	1	134
12	1	20:50	05:45	410	535 (8.9)	2	
	2	19:00	22:15	195	195 (3.3)	2	
	3	17:05	20:20	195	195 (3.3)	1	
	4	07:55	18:30	405	635 (10.6)	1	120
13	1	20:50	06:00	410	550 (9.2)	2	
	2	08:30	16:45	415	495 (8.3)	1	
	3	14:00	22:30	375	510 (8.5)	1	
	4	16:00	19:00	180	180 (3.0)	1	
	5	17:05	20:00	175	175 (2.9)	1	129
14	1	20:50	06:10	405	560 (9.3)	2	
	2	13:00	22:00	370	540 (9.0)	1	
	3	11:00	14:00	180	180 (3.0)	1	
	4	12:30	21:30	375	540 (9.0)	1	133
15	1	20:50	06:15	405	565 (9.4)	2	
	2	07:45	15:40	385	475 (7.9)	1	
	3	09:00	17:00	360	480 (8.0)	1	
	4	17:00	20:00	180	180 (3.0)	1	133
16	1	20:50	06:30	405	580 (9.7)	2	
	2	09:30	17:30	385	480 (8.0)	1	
	3	15:45	23:15	360	450 (7.5)	1	
	4	17:30	20:30	180	180 (3.0)	1	133

Roster	Week	Beg Time	End Time	Work Time	Total Time	Weekend	Monthly
Depot 2							
1	1	18:10	09:00	270	890 (14.8)	2	
	2	13:05	21:20	390	495 (8.3)	1	
	3	06:30	14:45	395	495 (8.3)	2	
	4	16:15	16:00	195	1425 (23.8)	1	125
2	1	09:55	18:40	385	525 (8.8)	1	
	2	06:30	14:45	395	495 (8.3)	2	
	3	16:00	19:15	195	195 (3.3)	1	
	4	20:15	03:30	365	435 (7.3)	2	134
3	1	00:30	03:45	195	195 (3.3)	3	
	2	07:00	16:25	415	565 (9.4)	1	
	3	12:00	21:50	380	590 (9.8)	1	
	4	10:30	18:10	370	460 (7.7)	1	136
4	1	12:15	20:30	405	495 (8.3)	1	
	2	23:00	02:45	225	225 (3.8)	3	
	3	08:00	16:15	375	495 (8.3)	1	
	4	15:05	22:15	355	430 (7.2)	1	136
5	1	12:00	19:55	400	475 (7.9)	1	
	2	12:55	21:05	375	490 (8.2)	1	
	3	12:35	20:15	355	460 (7.7)	1	
	4	23:50	03:30	220	220 (3.7)	3	135
6	1	10:00	18:10	400	490 (8.2)	1	
	2	10:15	13:30	195	195 (3.3)	1	
	3	21:05	10:15	175	790 (13.2)	2	
	4	07:20	15:00	380	460 (7.7)	1	
	5	13:00	20:30	360	450 (7.5)	1	125
7	1	20:30	03:55	360	445 (7.4)	2	
	2	09:00	18:10	400	550 (9.2)	1	
	3	12:00	20:30	375	510 (8.5)	1	
	4	13:40	18:30	290	290 (4.8)	1	
	5	10:50	23:00	190	730 (12.2)	1	134
8	1	20:15	03:15	340	420 (7.0)	2	
	2	09:00	17:05	395	485 (8.1)	1	
	3	12:00	20:15	375	495 (8.3)	1	
	4	09:00	13:05	245	245 (4.1)	1	
	5	16:30	13:40	180	1270 (21.2)	1	127
9	1	20:30	03:20	335	410 (6.8)	2	
	2	12:00	20:20	395	500 (8.3)	1	
	3	13:00	22:15	375	555 (9.3)	1	
	4	10:00	00:30	250	870 (14.5)	1	135
10	1	20:15	02:45	325	390 (6.5)	2	
	2	13:00	21:10	395	490 (8.2)	1	
	3	10:50	18:30	370	460 (7.7)	1	
	4	11:45	23:50	225	725 (12.1)	1	131

CHL_288 - Time Blocks Considered ($min\_out\_tb = 60$)

Roster	Week	Beg Time	End Time	Work Time	Total Time	Weekend	Monthly
Depot 0							
1	1	22:15	08:45	435	630 (10.5)	3	
	2	23:20	07:00	380	460 (7.7)	3	135
2	1	22:20	06:40	405	500 (8.3)	3	
	2	21:45	11:00	295	795 (13.3)	2	
	3	11:00	20:20	450	560 (9.3)	1	143
3	1	23:00	07:10	390	490 (8.2)	3	
	2	07:30	19:30	500	720 (12.0)	1	
	3	10:20	19:45	405	565 (9.4)	1	
	4	11:00	14:30	210	210 (3.5)	1	150
4	1	23:20	07:00	385	460 (7.7)	3	
	2	08:00	16:15	405	495 (8.3)	1	
	3	13:40	02:30	290	770 (12.8)	1	
	4	11:00	20:00	405	540 (9.0)	1	148
5	1	23:00	07:10	385	490 (8.2)	3	
	2	08:00	18:45	480	645 (10.8)	1	
	3	12:15	15:45	145	210 (3.5)	1	
	4	09:00	18:45	485	585 (9.8)	1	149
6	1	09:00	18:30	465	570 (9.5)	1	
	2	15:00	23:45	395	525 (8.8)	1	
	3	02:30	06:00	210	210 (3.5)	3	133
7	1	23:00	06:25	370	445 (7.4)	3	
	2	09:30	18:00	395	510 (8.5)	1	
	3	09:00	18:30	410	570 (9.5)	1	146
8	1	23:00	06:15	365	435 (7.3)	3	
	2	13:00	21:50	415	530 (8.8)	1	
	3	12:20	20:00	390	460 (7.7)	1	146
9	1	23:00	06:40	365	460 (7.7)	3	
	2	12:15	23:00	415	645 (10.8)	1	
	3	13:25	00:00	390	635 (10.6)	1	146
10	1	23:20	06:25	360	425 (7.1)	3	
	2	11:30	20:30	410	540 (9.0)	1	
	3	13:00	21:00	375	480 (8.0)	1	143
Depot 1							
1	1	23:00	09:50	425	650 (10.8)	3	
	2	19:15	06:00	375	645 (10.8)	2	
	3	13:00	21:45	390	525 (8.8)	1	148
2	1	23:30	07:00	375	450 (7.5)	3	
	2	19:15	06:00	375	645 (10.8)	2	
	3	07:55	17:00	390	545 (9.1)	1	142
3	1	06:10	14:45	405	515 (8.6)	2	
	2	19:15	06:00	375	645 (10.8)	2	
	3	11:45	20:20	390	515 (8.6)	1	
	4	09:15	17:30	375	495 (8.3)	1	154
4	1	19:00	05:00	390	600 (10.0)	2	
	2	19:15	06:15	375	660 (11.0)	2	
	3	09:30	18:00	385	510 (8.5)	1	
	4	10:15	19:30	375	555 (9.3)	1	152
5	1	19:00	06:30	390	690 (11.5)	2	
	2	20:50	06:15	375	565 (9.4)	2	
	3	09:30	18:30	385	540 (9.0)	1	
	4	11:30	20:00	375	510 (8.5)	1	152
6	1	19:15	05:20	380	605 (10.1)	2	
	2	19:15	06:15	365	660 (11.0)	2	
	3	13:00	21:30	375	510 (8.5)	1	
	4	12:45	20:30	375	465 (7.8)	1	149
7	1	19:15	06:00	380	645 (10.8)	2	
	2	11:30	21:45	415	615 (10.2)	1	
	3	07:45	15:30	375	465 (7.8)	1	
	4	12:00	20:00	370	480 (8.0)	1	154
8	1	06:40	19:00	380	740 (12.3)	2	
	2	08:30	16:45	415	495 (8.3)	1	
	3	08:45	16:30	375	465 (7.8)	1	
	4	16:00	00:00	370	480 (8.0)	1	154
9	1	06:40	14:30	380	470 (7.8)	2	
	2	08:15	16:25	400	490 (8.2)	1	
	3	09:45	19:30	375	585 (9.8)	1	
	4	09:00	17:00	360	480 (8.0)	1	151
10	1	19:15	05:45	380	630 (10.5)	2	
	2	10:45	19:30	390	525 (8.8)	1	
	3	12:30	21:00	375	510 (8.5)	1	
	4	07:25	15:15	355	470 (7.8)	1	150
11	1	19:15	06:10	375	655 (10.9)	2	
	2	08:30	19:30	390	660 (11.0)	1	
	3	09:00	17:00	375	480 (8.0)	1	
	4	10:00	18:00	355	480 (8.0)	1	149

Roster	Week	Beg Time	End Time	Work Time	Total Time	Weekend	Monthly
Depot 2							
1	1	02:45	17:00	255	855 (14.3)	3	
	2	19:30	03:15	375	465 (7.8)	2	
	3	07:00	17:20	440	620 (10.3)	1	133
2	1	19:15	03:30	410	495 (8.3)	2	
	2	09:00	18:05	410	545 (9.1)	1	
	3	17:00	20:30	210	210 (3.5)	1	
	4	19:00	03:25	410	505 (8.4)	2	144
3	1	19:15	03:15	370	480 (8.0)	2	
	2	10:40	19:00	410	500 (8.3)	1	
	3	11:00	21:50	380	650 (10.8)	1	
	4	19:00	03:55	375	535 (8.9)	2	153
4	1	19:30	02:55	370	445 (7.4)	2	
	2	10:50	19:55	410	545 (9.1)	1	
	3	12:00	20:15	375	495 (8.3)	1	
	4	19:05	03:30	375	505 (8.4)	2	153
5	1	19:15	02:45	365	450 (7.5)	2	
	2	12:15	20:30	405	495 (8.3)	1	
	3	08:00	17:15	375	555 (9.3)	1	
	4	19:00	03:20	385	500 (8.3)	2	153
6	1	19:00	03:00	395	480 (8)	2	
	2	19:30	03:05	365	455 (7.6)	2	
	3	07:20	18:10	405	650 (10.8)	1	
	4	09:55	19:00	375	545 (9.1)	1	154
7	1	19:00	03:35	390	515 (8.6)	2	
	2	19:15	02:40	355	445 (7.4)	2	
	3	09:00	18:10	400	550 (9.2)	1	
	4	13:00	21:10	370	490 (8.2)	1	151
8	1	19:15	03:10	385	475 (7.9)	2	
	2	19:15	03:00	345	465 (7.8)	2	
	3	12:00	20:20	395	500 (8.3)	1	
	4	10:00	18:10	370	490 (8.2)	1	149
9	1	20:30	03:20	335	410 (6.8)	2	
	2	09:00	17:05	395	485 (8.1)	1	
	3	10:50	19:05	365	495 (8.3)	1	
	4	19:00	03:30	400	510 (8.5)	2	149
10	1	19:15	02:25	325	430 (7.2)	2	
	2	13:05	21:20	390	495 (8.3)	1	
	3	12:30	20:30	385	480 (8.0)	1	
	4	19:00	03:45	405	525 (8.8)	2	150
11	1	19:15	02:45	375	450 (7.5)	2	
	2	11:15	20:15	385	540 (9.0)	1	
	3	10:00	18:30	415	510 (8.5)	1	146
12	1	19:15	02:45	375	450 (7.5)	2	
	2	11:45	21:05	440	560 (9.3)	1	
	3	10:30	18:40	385	490 (8.2)	1	150

CHL_288 - Time Blocks Considered ($min\_out\_tb = 120$)

Roster	Week	Beg Time	End Time	Work Time	Total Time	Weekend	Monthly
Depot 0							
1	1	22:20	06:40	405	500 (8.3)	3	
	2	18:00	06:15	385	735 (12.3)	2	
	3	11:15	20:00	405	525 (8.8)	1	149
2	1	03:20	06:40	200	200 (3.3)	3	
	2	08:00	20:20	470	740 (12.3)	1	
	3	08:00	16:15	405	495 (8.3)	1	
	4	12:15	23:00	415	645 (10.8)	1	149
3	1	18:00	06:15	395	735 (12.3)	2	
	2	14:30	22:45	365	495 (8.3)	1	
	3	18:00	06:25	385	745 (12.4)	2	
	4	15:00	23:45	395	525 (8.8)	1	154
4	1	18:00	06:00	395	720 (12.0)	2	
	2	13:30	00:00	400	630 (10.5)	1	
	3	15:00	23:45	365	525 (8.8)	1	
	4	13:25	22:45	370	560 (9.3)	1	153
5	1	18:00	06:25	390	745 (12.4)	2	
	2	09:00	18:30	410	570 (9.5)	1	
	3	08:20	16:25	400	485 (8.1)	1	
	4	13:40	03:20	290	820 (13.7)	1	149
6	1	07:30	18:00	405	630 (10.5)	1	
	2	12:15	15:45	105	210 (3.5)	1	
	3	09:00	18:30	465	570 (9.5)	1	
	4	11:00	20:00	405	540 (9.0)	1	138
Depot 1							
1	1	12:30	20:20	390	470 (7.8)	1	
	2	00:40	10:25	380	585 (9.8)	3	
	3	19:15	05:45	380	630 (10.5)	2	143
2	1	09:30	17:30	385	480 (8.0)	1	
	2	23:50	09:50	420	600 (10.0)	3	
	3	18:00	05:00	385	660 (11.0)	2	148
3	1	23:30	07:00	375	450 (7.5)	3	
	2	06:40	19:00	380	740 (12.3)	2	
	3	09:00	18:30	375	570 (9.5)	1	141
4	1	20:50	06:00	410	550 (9.2)	2	
	2	12:00	20:00	375	480 (8.0)	1	
	3	10:15	18:00	370	465 (7.8)	1	
	4	19:00	06:00	390	660 (11.0)	2	154
5	1	08:30	16:45	415	495 (8.3)	1	
	2	13:00	00:00	370	660 (11.0)	1	
	3	12:45	20:30	375	465 (7.8)	1	
	4	10:00	18:00	360	480 (8.0)	1	152
6	1	07:55	19:30	405	695 (11.6)	1	
	2	08:30	16:30	375	480 (8.0)	1	
	3	11:45	20:00	370	495 (8.3)	1	
	4	06:40	16:00	375	560 (9.3)	2	152
7	1	06:10	15:00	390	530 (8.8)	2	
	2	08:15	16:25	400	490 (8.2)	1	
	3	09:15	17:00	375	465 (7.8)	1	
	4	09:00	17:00	360	480 (8.0)	1	152
8	1	18:00	06:00	385	720 (12.0)	2	
	2	11:30	19:30	390	480 (8.0)	1	
	3	18:30	06:00	395	690 (11.5)	2	
	4	07:45	16:00	375	495 (8.3)	1	154

Roster	Week	Beg Time	End Time	Work Time	Total Time	Weekend	Monthly
Depot 2							
1	1	19:00	03:30	395	510 (8.5)	2	
	2	11:30	21:20	410	590 (9.8)	1	
	3	03:10	14:45	380	695 (11.6)	3	148
2	1	19:30	03:25	395	475 (7.9)	2	
	2	10:40	19:00	410	500 (8.3)	1	
	3	03:15	14:00	390	645 (10.8)	3	149
3	1	19:00	03:20	385	500 (8.3)	2	
	2	10:00	19:05	410	545 (9.1)	1	
	3	02:00	10:40	395	520 (8.7)	3	148
4	1	03:30	15:05	410	695 (11.6)	3	
	2	18:30	03:05	380	515 (8.6)	2	
	3	11:45	21:05	410	560 (9.3)	1	150
5	1	03:00	12:55	400	595 (9.9)	3	
	2	18:10	02:45	380	515 (8.6)	2	
	3	11:15	20:20	405	545 (9.1)	1	148
6	1	03:30	16:15	400	765 (12.8)	3	
	2	18:10	02:45	380	515 (8.6)	2	
	3	14:45	01:45	405	660 (11.0)	1	148
7	1	19:05	03:00	375	475 (7.9)	2	
	2	12:15	20:30	405	495 (8.3)	1	
	3	03:20	13:05	435	585 (9.8)	3	151
8	1	02:55	13:00	390	605 (10.1)	3	
	2	18:30	03:15	375	525 (8.8)	2	
	3	17:15	01:45	405	510 (8.5)	1	146
9	1	19:15	03:15	370	480 (8.0)	2	
	2	15:45	01:45	405	600 (10.0)	1	
	3	03:25	14:40	445	675 (11.3)	3	152
10	1	18:10	02:45	370	515 (8.6)	2	
	2	14:45	01:45	405	660 (11.0)	1	
	3	03:15	13:05	470	590 (9.8)	3	155
11	1	03:30	12:00	360	510 (8.5)	3	
	2	12:00	19:55	400	475 (7.9)	1	
	3	12:35	20:15	355	460 (7.7)	1	
	4	11:00	20:30	420	570 (9.5)	1	153
12	1	19:15	02:40	355	445 (7.4)	2	
	2	09:55	18:10	400	495 (8.3)	1	
	3	11:00	18:40	385	460 (7.7)	1	
	4	18:10	02:55	405	525 (8.8)	2	154
13	1	18:05	03:35	415	570 (9.5)	2	
	2	19:15	03:10	355	475 (7.9)	2	
	3	10:00	18:10	400	490 (8.2)	1	
	4	09:00	17:15	375	495 (8.3)	1	154
14	1	18:05	02:55	410	530 (8.8)	2	
	2	18:10	02:25	345	495 (8.3)	2	
	3	09:00	17:05	395	485 (8.1)	1	
	4	11:30	19:15	375	465 (7.8)	1	152
15	1	20:30	03:20	335	410 (6.8)	2	
	2	10:50	21:05	395	615 (10.3)	1	
	3	12:30	20:30	375	480 (8.0)	1	
	4	18:10	03:00	415	530 (8.8)	2	152
16	1	18:00	02:45	405	525 (8.8)	2	
	2	10:50	19:00	385	490 (8.2)	1	
	3	19:15	03:30	365	495 (8.3)	2	
	4	08:00	16:15	375	495 (8.3)	1	153
17	1	19:00	03:45	405	525 (8.8)	2	
	2	07:00	16:25	415	565 (9.4)	1	
	3	12:30	20:15	385	465 (7.8)	1	
	4	10:30	18:10	370	460 (7.7)	1	157

Appendix C

Roster with a 7-days week of work

Table C.1: Example 1 - Fused Rosters Depot 16, Instance ITA_908: Roster with a 7-days week of work

Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
Shift 1	A	A	A	A	A	A	A	B	B	B	B	B	B	B	B	C	C	C	C	C	C	C	D	D	D	D	D	D	E	E	E	E	E	E	E	F	F	F	F	G	G	G	G	G	G	G	G
Shift 2	F	F	H	H	H	H	H	H	H	H	A	A	A	A	A	A	B	B	B	B	B	B	B	C	C	C	C	C	C	D	D	D	D	D	D	D	E	E	E	E	E	E	E	F	F	F	F
Shift 3	E	E	E	F	F	F	F	F	F	F	F	H	H	H	H	H	H	A	A	A	A	A	A	A	B	B	B	B	B	B	B	C	C	C	C	C	C	C	D	D	D	D	D	D	D	E	E
Shift 4	D	D	D	D	E	E	E	E	E	E	E	E	F	F	F	F	F	F	H	H	H	H	H	H	H	A	A	A	A	A	A	A	B	B	B	B	B	B	C	C	C	C	C	C	C	D	D
Shift 5	C	C	C	C	C	C	D	D	D	D	D	D	E	E	E	E	E	E	E	F	F	F	F	F	F	F	H	H	H	H	H	H	H	A	A	A	A	A	A	B	B	B	B	B	B	C	C
Shift 6	B	B	B	B	B	B	C	C	C	C	C	C	C	D	D	D	D	D	D	D	E	E	E	E	E	E	E	F	F	F	F	F	F	F	H	H	H	H	H	H	H	A	A	A	A	A	B
Rest	G	G	F	E	D	C	B	A	A	H	F	E	D	C	B	B	A	H	F	E	D	C	C	B	A	H	F	E	D	C	B	B	A	H	F	E	D	C	B	A	H	F	E	D	C	B	A