Mathematical modeling and methods for rescheduling trains under disrupted operations

Rodrigo Acuña-Agost

ACADÉMIE D'AIX-MARSEILLE
UNIVERSITÉ D'AVIGNON ET DES PAYS DE VAUCLUSE

THÈSE

présentée à l'Université d'Avignon et des Pays de Vaucluse
pour obtenir le diplôme de DOCTORAT

SPÉCIALITÉ : Informatique

École Doctorale 166 « Information Structures Systèmes »
Laboratoire d'Informatique (EA 931)

Mathematical Modeling and Methods for Rescheduling Trains under Disrupted Operations

par
Rodrigo ACUNA-AGOST

Soutenue publiquement le 15 septembre 2009 devant un jury composé de :

M.	Jacques FERLAND	Professeur, Université de Montreal, Canada	Rapporteur
M.	Paolo TOTH	Professeur, Università di Bologna, Italie	Rapporteur
M.	Gilles DESSAGNE	Direction de l'innovation et de la recherche, SNCF, Paris	Examinateur
M.	Dominique FEILLET	Professeur, Ecole des Mines de Saint-Etienne, Gardanne	Co-Directeur de thèse
M.	Serigne GUEYE	Maître de Conférences, LMAH, Université du Havre	Co-Directeur de thèse
M^{me}.	Lorena PRADENAS	Professeur, Universidad de Concepción, Chili	Examinateur
M.	Philippe MICHELON	Professeur, LIA, Université d'Avignon	Directeur de thèse

---

Laboratoire d'Informatique d'Avignon

Acknowledgments

I acknowledge the financial support received from the European Community (ALFA project II-0457-FA-FCD-FI-FC) and PREDIT (MAGES project).

The first person I want to thank is my Tutor Professor Philippe Michelon for giving me the chance of working with him and providing not only academic and professional support, but also for his invaluable friendship. Thank for the dedicated time and for contributing with your precious and irreplaceable professional experience in the development of this Thesis.

I want to thank Serigne Gueye for accepting me in the MAGES project directed by him. Thanks also for his friendly welcome when I arrived in France. Mariela and I really appreciate his help in those complex moments during our adaptation in this country in special for the help in all administrative procedures that he assisted us without any formal obligation.

I want to thank also Dominique Feillet for his outstanding professionalism and the time dedicated to the success of this work. I was really lucky to have him in the research team. I learned so much from him that can be summed up in two words: dedication and rigorousness.

Philippe, Serigne, Dominique, and I integrated an excellent team. It was a nice group where everyone contributed with his own perspective to the success of all proposed tasks. Thanks you three for supporting me in my research ideas and for giving me the autonomy and assistance needed at each phase of this Thesis.

This Thesis has never existed without the invitation of Lorena Pradenas to participate in the ALFA project. I want to thank her for believing in me and proposing me this great opportunity. I always remember the day when I received her phone call with this proposition while I was driving my car. I am not sure why but I immediately said "yes" maybe because the traffic light was red. Seriously, I always wanted to have a doctor degree and Lorena was one of the precursors of this.

There is also an important acknowledgment to the reviewers. Thanks Jacques Ferland and Paolo Toth for accepting to be the reviewers of my doctoral Thesis. I just have to say "what more can you ask for?", they are two very prestigious professors and many students would be happy (like me) to have them in their thesis jury.

I gratefully acknowledge the SNCF department "Direction de l'innovation et de la

recherche" in special to Gilles Dessagne for supplying relevant information that facilitated the construction of the MIP model and the French network. We had several work meetings in Paris and I really appreciated his disposition and interest in our project. In the same way, I want to thanks my friend Jonhson Ahumada of FEPASA[1] for supplying the information needed to build the Chilean railway network presented in some chapters of this Thesis.

Many thanks go to all the other members of LIA (*Laboratoire d'Informatique d'Avignon*), in particular to the members of the Operations Research team: Oliver, Diego, Sylvain, Claire, Boris, Dominique Quadri, and Liudmila. They all gave me support and encouragement during my stay in Avignon. Thanks too Simone and Jocelyne for the efficient support in all administrative tasks at LIA.

This very last paragraph is dedicated to my family and friends. If you read this, please do not be angry because you are the last since you are actually the first on my mind. Thanks Mariela for all your love and patience. I will always thank you all you have renounced for me and this Thesis. Thanks to the members of my family in Chile for all the love they have given me all these years, in special to my fathers (Virginia and Ernesto) and my brothers (Virginia and Mauricio). I also want to thank my latino friends with whom I shared good times in Europe: Silvia Fernandez, Pedro Gonzalez, Sulan Wong, Julio Rojas, Francisco Escandón, and Carolina Reyes. Thanks for your friendship that made my stay in Avignon more pleasant. Finally, I want to thank my office mates: Tembine, Abdellatif, and Saïd; for the agreeable environment of work and good times we had at LIA.

Thank you all !!!

[1] The largest freight railroad company in Chile

to my wife Mariela ...

Abstract

For operational and unpredictable reasons, many small incidents occur day after day in rail transportation systems. Most of them have a local impact; but, in some cases, minimal disruptions can spread out through the whole network and affect significantly the train schedules. In this Thesis, we present the Railway Rescheduling Problem (RRP) as the problem of finding a new schedule of trains after one or several incidents by minimizing some measure of the effect, *e.g.*, the total delay. This Thesis has been developed in the context of the MAGES project that builds mathematical models and algorithms for optimizing railway operations.

Two complementary formulations are proposed to model this problem: Mixed-Integer Programming (MIP) and Constraint Programming (CP). Because of the impossibility of solving real-world instances by using standard solvers, we propose several solutions methods: right-shift rescheduling; a MIP-based local search method; Statistical Analysis of Propagation of Incidents (SAPI); and a CP-based approach. Some methods are presented in different versions by extending them to iterative approaches.

Among them; SAPI is one of the major contributions of this Thesis. It integrates the concepts of right-shift rescheduling and the MIP-based local search method by fixing integer variables and adding linear inequalities (cuts). SAPI assumes that the effects of disruptions can be propagated to other upcoming events. Nevertheless, this propagation is not uniform to all events and could be forecasted by a statistical analysis. Different versions of the methods are compared in two different networks located in France and Chile. From the results, it is possible to conclude that SAPI finds good solutions faster than the other methods, while a cooperative CP/MIP approach that takes advantage of both formulations seems to be appropriate for large instances.

Because of the difficulty to compare SAPI to other methods presented in the literature due to lack of public benchmarks, we applied it to another problem where public instances are available. Hence, the methodology was adapted and applied to the problem of rescheduling passengers, flights, and aircraft under disrupted operations in the context of the ROADEF challenge 2009. SAPI took the third position on this competition, showing that the method seems to be effective solving such type of problems efficiently.

Résumé (French)

En raison de problèmes opérationnels et d'autres événements inattendus, un grand nombre d'incidents se produisent quotidiennement dans les systèmes de transport ferroviaire. Certains d'entre eux ont un impact local, mais quelques fois, essentiellement dans les réseaux ferroviaires plus saturés, des petits incidents peuvent se propager à travers tout le réseau et perturber de manière significative les horaires des trains. Dans cette thèse doctorale, nous présentons le problème de réordonnancement de plan de circulation ferroviaire en cas d'incident comme la problématique de créer un plan de circulation provisoire de manière à minimiser les effets de la propagation des incidents. Ce travail est issu du projet MAGES (Module d'Aide à la Gestion des Sillons) qui développe des systèmes de régulation pour le trafic ferroviaire.

Nous présentons deux modèles différents qui permettent de trouver des solutions à ce problème : Programmation Linéaire en Nombres Entiers (PLNE) et Programmation Par Contraintes (PPC). Du fait de la nature fortement combinatoire du problème et de la nécessité de répondre rapidement aux incidents, il ne paraît pas raisonnable d'envisager une résolution exacte. Les méthodes correctives proposées consistent donc à explorer un voisinage restreint des solutions : *right-shift rescheduling*; une méthode basée sur des coupes de proximité; une méthode d'analyse statistique de la propagation des incidents (SAPI) et un méthode basée sur la PPC. Additionnellement, certaines de ces méthodes ont été adaptées sous forme d'algorithmes itératifs avec l'objectif d'améliorer progressivement la solution quand le temps d'exécution le permet.

SAPI est une des principales contributions de cette thèse. SAPI intègre les concepts de *right-shift rescheduling* avec les coupes de proximité. Du fait de la taille des réseaux en jeu et du nombre de circulations, les phénomènes complexes de propagation d'un incident font qu'il est très difficile de connaitre de manière précise les événements qui seront affectés. Toutefois, il est tout de même envisageable d'évaluer la probabilité qu'un événement soit affecté. Pour calculer cette probabilité, un modèle de régression logistique est utilisé avec des variables explicatives dérivées du réseau et des circulations. Diverses variantes de ces méthodes sont évaluées et comparées en utilisant deux réseaux ferroviaires localisés en France et au Chili. À partir des résultats obtenus, il est possible de conclure que SAPI est meilleure que les autres méthodes en terme de vitesse de convergence vers l'optimum pour les instances de petite taille et moyenne alors qu'une méthode coopérative PNLE/PPC est capable de trouver des solutions pour les instances de plus grande taille.

La difficulté de comparer SAPI avec d'autres méthodes présentées dans la littérature nous a encouragés à appliquer la méthode à un autre problème. Ainsi, cette méthodologie a été également adaptée au problème de réordonnancement de passagers, vols et appareils (avions) en cas de perturbations, problème originalement proposé dans le contexte du Challenge ROADEF 2009. Les résultats montrent que SAPI est efficace pour résoudre ce problème avec des solutions au-dessus de la moyenne des équipes finalistes en obtenant la troisième place du challenge.

Resumen (Spanish)

Debido a problemas operacionales y otros eventos inesperados, numerosos incidentes ocurren a diario en los sistemas ferroviarios. Muchos de ellos tienen un impacto local, sin embargo, sobre todo en redes de alta densidad, pequeñas perturbaciones pueden extenderse por toda la red y afectar significativamente los horarios de varios trenes. En esta tesis doctoral, se trata el problema de replanificación de trenes (RRP: *Railway Rescheduling Problem*) como el proceso de construir un nuevo plan horario reaccionando a incidentes. La construcción de este nuevo plan debe realizarse minimizando el impacto de las perturbaciones, por ejemplo, minimizando el atraso total. Este problema ha sido definido en el contexto del proyecto MAGES que desarrolla modelos matemáticos y algoritmos para operaciones ferroviarias optimizadas.

Para modelar este problema, dos formulaciones complementarias son propuestas: un modelo de programación matemática en números enteros (MIP: *Mixed-Integer Programming*) y un modelo de programación por restricciones (CP: *Constraint Programming*). Debido a que es imposible solucionar instancias reales de este problema usando software generales de optimización, se han propuesto diferentes métodos de resolución: *right-shift rescheduling*; un método de búsqueda local basado en el modelo MIP, análisis estadístico de propagación de incidentes (SAPI: *Statistical Analysis of Propagation of Incidents*) y un algoritmo basado en el modelo CP. Adicionalmente, algunos de estos procedimientos fueron adaptados como algoritmos iterativos con el objetivo de mejorar progresivamente la solución.

SAPI es una de las principales contribuciones de esta tesis. El integra los conceptos de *right-shift rescheduling* y el método de búsqueda local basado en el modelo MIP. El se fundamenta en el supuesto que los efectos de las perturbaciones son propagados de una manera no uniforme. De esta manera, con la ayuda de un análisis estadístico, SAPI estudia la probabilidad de que un evento particular sea afectado por las perturbaciones. Esta información es luego utilizanda para fijar variables de decisión enteras y agregar inecuaciones válidas (cortes). Diferentes variaciones de estos métodos son probados y comparados usando dos diferentes redes ferroviarias las cuales están localizadas en Francia y Chile. A partir de los resultados obtenidos, es posible concluir que SAPI es el procedimiento que entrega soluciones de buena calidad con la mayor velocidad de convergencia al óptimo. A su vez, un algoritmo cooperativo CP/MIP que considera las ventajas de ambos modelos, muestra ser el más apropiado para instancias de gran tamaño.

Debido a la dificultad de comparar SAPI con otros métodos presentes en la literatura, se ha aplicado a otro problema del cual existen instancias públicas y resultados de otros autores. Así, esta metodología fue empleada para la replanificación de pasajeros, vuelos y aviones bajo operaciones perturbadas; problema propuesto en el contexto del desafío ROADEF 2009. SAPI obtuvo el tercer lugar de la competencia, mostrando ser eficiente para resolver este tipo de problemas.

Contents

Part I

Introduction and General Concepts

Chapter 1

Introduction

The Operations Research (OR) community has been working for many decades in computational solutions to many combinatorial problems in railways. In spite of the improvements in solution methods and hardware, it is a fact that the complexity of these problems has significantly increased in the last years and many problems are still open due to several reasons such as new regulation rules and an increasing demand of railway services.

First, in Europe and particularly in France, new regulation rules specifies that infrastructure administration and operations of trains have to be performed by separate entities. Thereby, infrastructure management should be guaranteed by governments while train operations (cargo and passengers) should be carried out by independent companies based on a commercial basis.

In France, the infrastructure network is now managed by the Railway Network of France (RFF)[1]. On this network operates the (historical) French National Railway Company (SNCF)[2] for both passenger and freight transportation; however since the liberalization of freight traffic in 2006, the network is open to other private companies. In the same way, railway passenger traffic will be also open to other companies in January 2010. This new situation is particularly complicated for traffic control. For example in France, this task is still assured by SNCF and RFF must pay for this service while SNCF pay to RFF for using the infrastructure[3]. This situation may change in the future and more than 14000 employees charged of these tasks (traffic control) should pass from SNCF to a new filial of RFF (Honore, 2008).

In the second place, geographic mobility has increased in the last years. This fact is particularly true in Europe where real and psychological frontiers are disappearing progressively. This is one of the main reasons why the demand of passengers has increased in both national and international trips. It is also expected that the irruption of new competitors in 2010 helps to decrease the price of tickets and so the demand

[1] RFF: Réseau Ferré de France

[2] SNCF: Société Nationale des Chemins de fer Français

[3] RFF payed 2700 millon of euros to SNCF in 2007 (Bauer, 2008) (RFF, 2009)

of train tickets would augment because some price-sensitive passengers will pass from other means of transportation. An instructive example is the intention of Air France to operate their own trains for routes where they have closed their flights definitely, *e.g.*, Paris - Brussels (Lomazzi, 2008)[4].

As a consequence, European railway networks are every day more and more saturated and there is no evidence that this trend will change in the next few years. It is then a necessity to manage the capacity of the network adequately in order to maintain a good service for passengers and rentable operations for all implied companies.

A railway schedule includes arrival/departure times of a set of trains at each station of a railway subnetwork and should also include an assignment of railway resources, *e.g.*, tracks and platforms. During the planning phase, managers try to minimize operational costs by meeting the requirements of customers and respecting operational, marketing, political, and other kinds of constraints.

However, planning is only half part of the whole process. No matter how good the original schedule is, in the execution phase, unexpected events (such as bad weather conditions, equipment breakdowns, and accidents) disturb the system making the plan suboptimal or infeasible. In that case, conflicts in the use of tracks and platforms may cause further propagation of disturbances. At this point, a reschedule procedure is needed to suggest new arrival and departure times and a new assignment of resources in a best possible way. As a consequence, traffic dispatchers have to take several decisions including changes in the order of trains, performing unplanned stops, and reassigning tracks and platforms. In parallel, they need to assure that the new assignment is as similar as possible to the original (optimized) schedule with the aim of guaranteeing the stability of the system. In the temporal dimension, this is equivalent to minimize the total delay of trains. The main problem treated in this Thesis is to construct a new provisional (temporary) train schedule after disruptions by respecting all these constraints and minimizing the impact of the incidents. We refer to this problem as the **Railway Rescheduling Problem** (RRP).

This Thesis is developed in the context of the MAGES project that had the goal to develop mathematical models and algorithms for optimizing railway operations[5]. The project was supervised by the PREDIT with a financial support of ADEME and one important partner organization: SNCF [6] [7]. In particular, along this document we try to answer one of the main questions proposed in the project[8]:

How could a train schedule be repaired in real-time after an incident?

This question is vital for SNCF and RFF. Note that SNCF is still assuming all traffic

[4]Air France: French airline and one of the world's largest airlines: http://www.airfrance.com

[5]MAGES: Modules d'Aide à la GEstion des Sillons (Assistance Modules for Managing Tracks) http://awal.univ-lehavre.fr/ lmah/mages/

[6]PREDIT: Programme de Recherche et D'Innovation dans les Transport terrestres (Innovation and Recherche Program in Ground Transportation) http://www.predit.prd.fr

[7]ADEME: Agence de l'Environnement et de la Maîtrise de l'Energie (French Environment and Energy Management Agency) http://www.ademe.fr/

[8]For more information about the objectives of the MAGES project see (Gueye, 2008)

control tasks; and RFF is the legal responsible but will be the exclusive executor of these tasks in the future.

Therefore, our work is about the development of new models and solution methods based on OR techniques for the RRP in the context of the MAGES project. For the purpose of answering the question proposed in the MAGES project, the objectives of this Thesis are to:

- study the railway rescheduling problem (RRP).
- develop complementary mathematical models (formulations) for the RRP.
- find, describe, implement and test new solution methods to solve the RRP.
- develop a new general method that may apply to any rescheduling problem in the transportation industry.

These objectives are answered though different chapters of this Thesis. The document is thereby organized in four parts: I) Introduction and General Concepts, II) Modeling and Solving the Railway Rescheduling Problem, III) Generalization of SAPI, and IV) Conclusions and Appendices.

The first part presents general concepts needed to understand the RRP. Chapter 2 introduces the reader to the main optimization problems in railways in order to grasp the context and the relationship of RRP with other important problems in railways. Chapter 3 presents general concepts of *Disruption Management*, and describes deeper the RRP with an analysis of the literature on this problem.

The second part of the Thesis, II) Modeling and Solving the Railway Rescheduling Problem, explains the proposed approaches for modeling and solving the problem. In Chapter 4, we present a Mixed Integer Programming (MIP) formulation and some solution methods based on this model. Chapter 5 presents a new solution approach called *Statistical Analysis of Propagation of Incidents* (SAPI). This method is one of the most important contributions of this Thesis because of its originality, the quality of the results and its applicability to other rescheduling problems. A Constraint Programming (CP) approach is then presented in Chapter 6. This method is a response to an inconvenient of the MIP model: large instances imply a huge number of binary variables. The CP formulation requires less number of variables and constraints and a cooperative approach mixing MIP and CP is developed to take into account the advantages of both paradigms. This cooperative method shows to be appropriate for solving large instances.

The third part of this Thesis, III) Generalization of SAPI, presents a generalization of SAPI and an additional application. Chapter 7 gives an explanation of a general methodology for disruption management. This approach is then completed by an additional application of SAPI for rescheduling flights, aircraft, and passengers under disrupted operations applied in the context of ROADEF challenge 2009 (Chapter 8). The results show that SAPI is also efficient for this additional application and seems to be appropriate to any kind of rescheduling problem. Finally, the conclusions, future directions of research, and the appendices are summed up in the last part of the Thesis.

Chapter 2

Operations Research in Railways

Contents

Abstract of the Chapter

This chapter introduces the main decision problems in rail transportation that have been studied by the Operations Research (OR) community. The objective is to understand the context of the railway rescheduling problem and show the relationship with other problems. This analysis is based on a hierarchical two-dimensional (time/space) classification.

2.1 Introduction

Rail transport consists in moving goods or passengers using railroads or railways. A railroad is composed of two parallel rails (steel or iron), attached perpendicularly to beams [1] (wood, steel or concrete) to keep a constant distance apart.

[1] called "sleepers" in U.K. and Australia and "crossties" or "ties" in U.S. and Canada.

The vehicles moving over the rails are arranged in a train: a set of vehicles coupled together. These vehicles could be classified in powered and unpowered. Powered vehicles are referred to as locomotive while unpowered vehicles are referred to as cars, carriages, wagons or coaches (for passengers).

When cars are moving over railways, they make much less friction than do vehicles with tires over paved roads. Actually, in a steel-on-steel rail system the coefficient of adhesion is eight times less than in road traffic (Pachl, 2004). As a result a train requires less energy to transport a given tonnage of freight, or a given number of passengers, than does road transport. Therefore, rail transport is the most energy-efficient land transportation system. Nevertheless, it is also a capital-intensive mean of transport, because of the investments needed to create, maintain and operate the whole railway network.

As other means of transport, rail transport is a combination of many components that give several optimization problems studied in operations research. This chapter focuses on the classification and a brief description of the main problems studied in the literature.

A two-dimensional classification of rail transportation problems and their relationship is then presented. This classification consists of two basic aspects of decision: time (when) and space (where). A two-dimensional approach was also presented in (Huisman et al., 2005), but is extended in this chapter for studying the relationship between different decision problems. This interaction of problems is interesting to evaluate the possibility of future integrated procedures by unifying horizontal and/or vertical adjacent decision problems and to evaluate eventual repercussions to other problems.

This chapter is organized as follows. Section 2.2 gives the classification and relationship of the problems in railway transportation. This section has been divided in several subsections explaining individually the most relevant decision problems. The last section draws concluding remarks for the chapter.

2.2 Classification Hierarchy

Figure 2.1 presents a two-dimensional point of view of planning problems in rail transportation: time and space. A relationship of these decisions (arrows connecting problems) is also presented with the purpose for describing how a decision impacts to others and to study the possibility of integration of problems. Depending on the robustness of the current solutions, changes in upper levels could affect lower levels. It is also important to remark that this chapter is mainly based on passenger trains and some exclusive cargo problems such as car blocking, train routing, and empty car distribution are not covered.

Note that Figure 2.1 only includes relations (arrows) between adjacent problems. For example, some of the inputs of "Line Planning" are the output of "Demand Analysis", nevertheless there is no arrow connecting them because there is an indirect rela-

tion, passing through "Network Planning", that connects these problems. This was a convention taken in order to keep this figure as simple as possible.

The first dimension is **time**. Railways problems have been classified according to the planning horizon in previous surveys as (Bussieck et al., 1997; Cordeau et al., 1998; Huisman et al., 2005; Caprara et al., 2007). The long-term planning horizon is also called strategic level and is focused on durable acquisition/planning of resources. Tactical planning concerns mid-term decisions while operational problems deal with the construction of detailed plans. The last horizon is the very short-term and corresponds to the control of railway operations and the response to disruptions.

The second dimension is **space**. Let consider the whole rail network as a graph: a node represents a station or a bifurcation gate, and an arc represents the railways connecting two nodes. While central decisions involve solving problems related to the graph (entire network), local decisions concern the problems related only to one arc or one node of this graph representation. Note that a node can be composed of several internal tracks (or platforms).

The possibility of **integration** is an important result of this classification. The progress in OR methods can be employed to solve simultaneously more than one individual problems in order to find a more "global" solution taking into account more precisely the objectives of the organization. This integration can be considered for related problems in both dimensions: horizontal and vertical.

2.2.1 Strategic Decisions

A strategy is a long-term coherent plan made to achieve a certain objective. In contrast to operational and tactical levels, the strategic level always has a global vision focused on a long-term success. Thus, in the context of railway operations, strategic decisions are concerned with the development of the network and long-term acquisition of resources, *e.g.*, *Network Planning*, *Rolling Stock Acquisition*, *Crew Planning*, and *Line Planning*. However, models for long-term decisions also have to include traffic demand data. For this reason, *Demand Analysis* is also considered at this level.

Demand Analysis

Demand Analysis involves to determine appropriately the future demand for rail transportation considering both: passengers and freight. As can be seen in Figure 2.1, demand analysis is the base of planning problems and gives one of the most important inputs for constructing the network and determining the magnitude of operations.

The estimation of demand for traveling is based on "traffic counts" or passenger surveys on some existing or future sections of the network. This problem consists in constructing the *origin-destination matrix* (OD matrix). This matrix contains information about the traffic between different zones of the network. However, because of some practical issues, *e.g.*, the samples are not taken simultaneously, data from "traffic

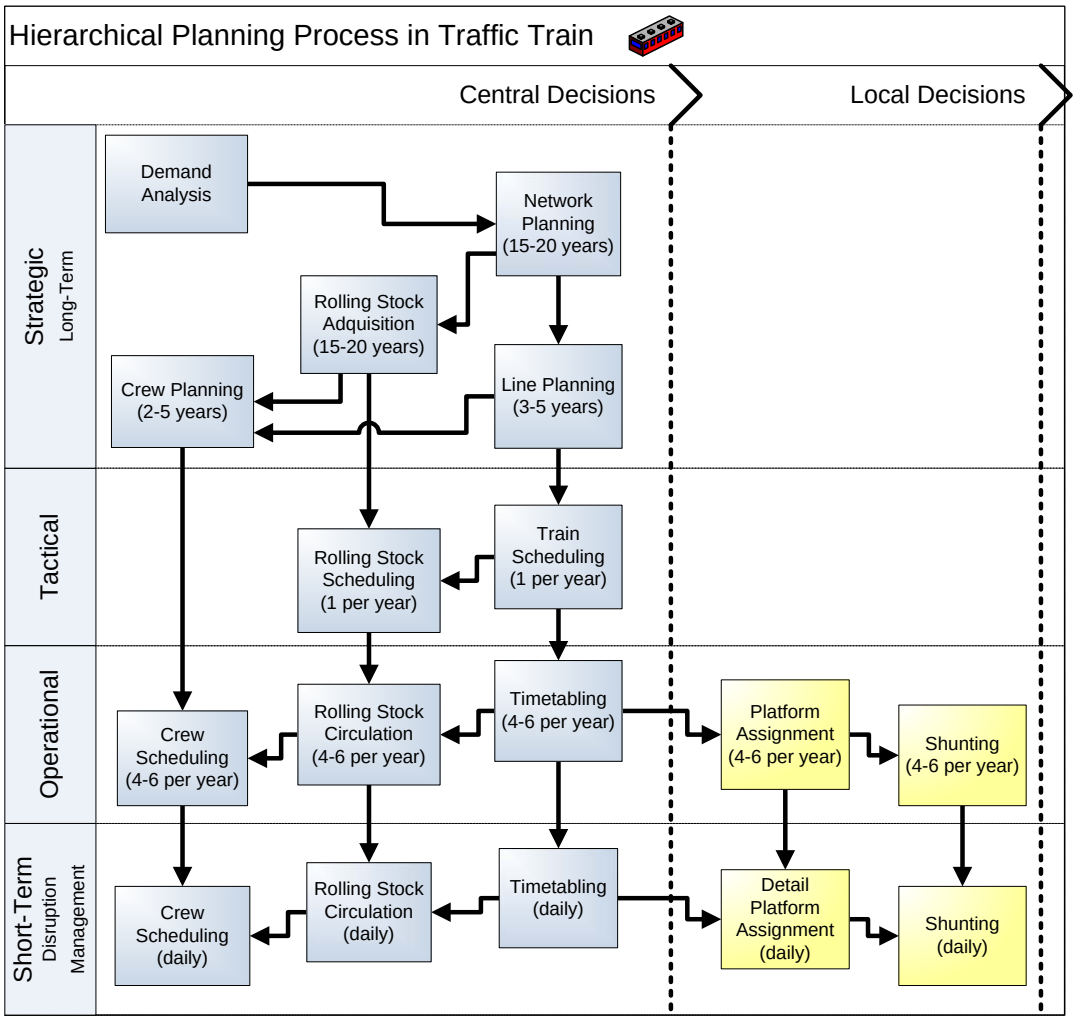


Figure 2.1: *Two-dimensional classification of rail transportation problems*

counts" present inconsistencies. Thus several approaches have been developed to obtain OD matrices using traffic counts. A good survey of these approaches can be found in (Abrahamsson, 1998). In this analysis, the network is divided in zones, where every zone has a representative point called centroid. It is assumed that all trips are between centroids, but in practice, centroids could constitute a subset of the nodes. The problem is that there are many OD matrices reproducing the observed traffic counts. The goal is to find the "best" OD matrix causing the observed traffic counts. After this, it is necessary to assign an OD matrix to a transportation network, *i.e.*, to allocate the demand between every pair of zones to available routes in the network.

Network Planning

The Network Planning Problem deals with the design of railroad networks. The complexity of this problem has increased in the last years, and making good decisions implies to consider new kinds of trains, new stations, new routes, changes in demand, and also to take into account the development of other means of transport. The complexity of the whole network requires managing several railroad subsystems separately. Furthermore, the infrastructure is shared by different types of services: long distance, medium/short distances, and freight trains.

Due to the fact that investment decisions require important amount of money and are taken for many years, this problem is classified as a strategic problem. It is important to remark that this problematic is more difficult to handle by optimization approaches than tactical and operational issues due to the extremely long planning horizons that are involved. This is the main reason why network planning decisions are taken using economic analysis considering several aspects: financial, political, environmental, and technological.

Like other capital budgeting projects, railway network planning requires to use classical economic evaluation methods as *net present value* analysis (NPV), *profitability index*, *internal rate of return* (IRR), and equivalent annuity. These methods use the incremental cash flows from the investment as new acquisition/constructions and modification/maintenances of the infrastructure.

Nevertheless, a pure economic analysis is not usually used to this kind of investment, because of the social nature of the railway context and political factors. Consequently, other tools like *Social Return on Investment* (SROI) have to be considered (REDF, 2001). SROI is an attempt to measure the social (and also financial) value created by a project in a long-term horizon.

Even though these tools are related to finances, operations research techniques could be used to help taking technical decisions as the location of stations and the capacity of the infrastructure. A good review for facility location in networks can be find in (Mesa and Boffey, 1996).

Rolling Stock Acquisition

Rolling Stock Acquisition corresponds to the decision of buying, selling, leasing, hiring, or wasting locomotives and cars. As shown in Figure 2.1, this decision depends on other important decisions: network planning and indirectly on demand analysis. With these decisions, it is possible to estimate the magnitude of the needed vehicles, but it is still necessarily to decide the quantity, the type, and the location of each one. This problem is considered as a strategic planning process because of the cost and the expected lifetime of the units, *e.g.*, the expected lifetime of rolling stock in France varies between 10 and 30 years (SNCF-Group, 2005).

In the scientific world, long-term rolling stock management has not received as

much attention as the operational and tactical versions of rolling stock problems (allocation and circulation of rolling stock). Thus, quantitative methods on scenario analysis have been used rather than operational research models even though rolling stock acquisition has direct implications in the quality of the service and the total cost. An instructive example can be seen in the French railways where the SNCF manages more than 7000 locomotives, 400 high speed trains (TGV) [2], and more than 47000 cars; with an investment over 1000 millions of EUROS per year (SNCF-Group, 2005).

Operators have to decide: selection of the type of rolling stock, acquisition of new rolling stock, hiring or leasing of rolling stock, lifetime extension of existing rolling stock, selling redundant rolling stock and destruction of obsolete rolling stock. For acquisition, it is necessary to determine not only the number of units, but also the capacity of every subcategory, *i.e.*, first and second classes for passenger units.

Additionally, some tradeoffs can be tackled by OR techniques. For example big capacity units are less flexible with regard to operations, but the smaller ones are relatively more expensive. It is also important to consider the differences between peak demand and off-peak demand. If the difference is high, it might be more advisable to acquire small units, using a big number of them in peak periods. Another important factor is the distance of the trajectories. It is reasonable to think that long distance services are better with large units. There is no doubt that all these aspects could be modeled by OR techniques.

In conclusion, the problem is to determine the number of units (using an appropriate aggregation level of details) respecting a given service level while minimizing the expected life cycle costs of the rolling stock.

Crew Planning (Long-Term)

Crew Planning is a very well known problem historically associated with airlines and mass transit companies. Long-term crew planning is related to the problem of deciding the labor levels for long periods of time. The decision is the quantity of contracted persons for every zone (or station). This decision depends on the estimation of the trips, labor laws, professional offer and other related constraints. The objective is to minimize the total cost of the long-term crew plan, considering not only the number of persons, but also the place where they are hired because of the difference of salaries and eventually hotels and travel allowances when the person has to stay in a different city.

Line Planning

A line is a route in a railway network that connects two terminal stations. In addition, a line frequency corresponds to the number of trains that use a line in a fixed interval, *e.g.*, one hour.

[2] TGV: Train à Grande Vitesse

The *Line Planning Problem* corresponds to the problem of selecting the set of lines and their frequencies in order to satisfy the demand while optimizing a fitness measure (see Figure 2.2). There are two main conflicting objectives: minimizing the operational costs of the railway system, and maximizing the number of travelers with direct connections.

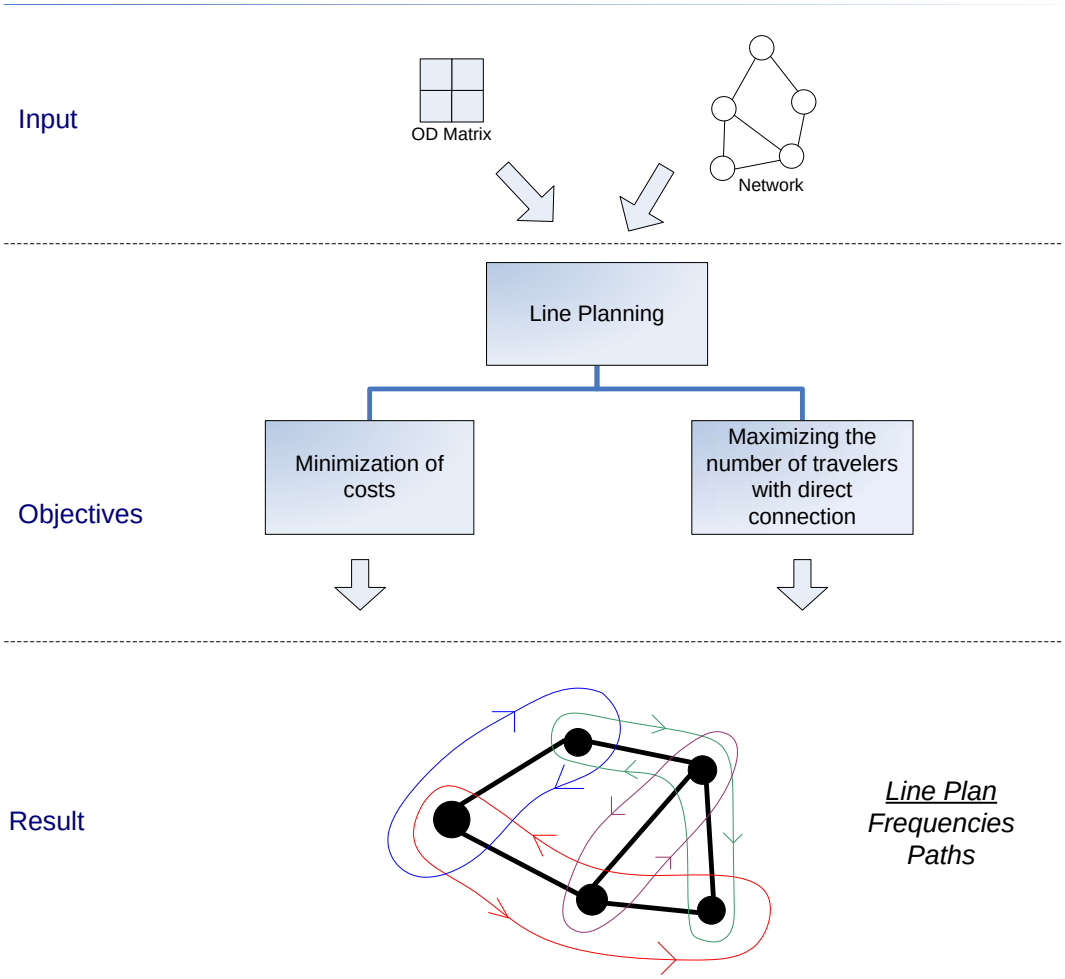


Figure 2.2: *The Line Planning Problem.*

Maximizing the number of travelers with direct connections results in long lines, *i.e.*, the longer a line, the more passengers traveling with direct trips. Nevertheless, long lines propagate delays more easily and deny an efficient allocation of rolling stock.

On the other hand, minimizing operational costs of the railway system implies shorter lines and passengers may perform changes in trains (connections). It is observed in previous surveys that both objectives have been studied, but the combination of the two in one model has not been described yet (Caprara et al., 2007).

As can be seen in Figures 2.1 and 2.2, the main input for the line planning problem are the railway infrastructure and the expected demand for railway transportation represented by an origin/destination matrix. In spite of the fact that the demand is not

symmetric because of the peak hours, line systems are usually symmetric, *e.g.*, for each pair of stations (Station A and Station B), the number of direct trains from Station A to Station B is approximately the same as the number of direct trains from Station B to Station A.

2.2.2 Tactical Decisions

Tactical planning concerns mid-term decisions, *e.g.*, once per year. In this section we present two different problems: *rolling stock scheduling* and *train scheduling*. They both involve centralized decision. In the literature there is not a clear consensus about the difference between tactical and operational decision in railways. That is, *rolling stock scheduling* and *rolling stock circulation* are in general the same problem. Some authors uses more clear names in order to be more precise in the definition of the problem, *e.g.*, (Fioole et al., 2006) deals with the *weekly* rolling stock planning problem. In our analysis, we separate tactical and operational problems with the aim of differentiating aggregated plans to detailed schedules.

Rolling Stock Scheduling (Tactical)

Rolling stock is a collection of available equipments like locomotives and railroad cars. Because of the limited stock of railway units, the problem of assigning them does not have a trivial solution. Moreover, some aspects have to be considered as maintenance of units and location of warehouses. Thus, the problem corresponds to assign the available stock of locomotives and cars while minimizing the total cost and satisfying all the constraints.

The objectives of the *Rolling Stock Scheduling* problem are: service quality, efficiency and robustness. The objective of service quality could be the minimization of seat shortage, efficiency the minimization of carriage kilometers and finally robustness the minimization of shunting. In principle, this tactical version of rolling stock scheduling should address to aggregated plan of all the vehicles that move on a railway and not necessary a detailed plan. This aggregated information is employed for planning mid-term operations and defining the budget of operations. Later, this aggregated plan must be converted to a detailed one.

Train Scheduling (Tactical)

Train scheduling corresponds to the problem of constructing the timetable of trains based on the line plan (see Section 2.2.1). In other words, a train schedule defines the arrival and departing time of the trains in a regular period. In the process of construction, several operational and marketing constraints have to be considered. Maximal speed of trains, minimal headway between consecutive trains, and maximal admitted longitude of the trains in stations are examples of operational constraints. On the other

hand, some examples of marketing constraints are the minimal and maximal stopping time in stations and minimal and maximal time for connections of trains. Usually, the objective is to minimize the total transit of passengers, *e.g.*, to minimize the total time used by the passengers in the system (including the travel and waiting time).

In real systems, an optimized procedure could give a good base solution that experienced human planners must adapt taking account of several local constraints that are difficult to include in mathematical models.

2.2.3 Operational Decisions

Operational problems deal with the construction of detailed plans. In this category we include the problems solved for horizons shorter than one year excluding very-short term problem that are treated in Section 2.2.4. Thus, this category includes: *detailed timetabling*, *rolling stock circulation*, *crew scheduling*, *platform assignment*, and *shunting*.

Detailed Timetabling (Operational)

The objective of this problem is to construct a detailed timetable for a set of trains on a certain part of the railway network valid during the planning horizon. Two kinds of timetables are identified in the literature: cyclical and noncyclical timetables.

On the one hand, cyclical timetables define a frequency of trains that are usually easy to remember for passengers. Nevertheless, they are relatively more expensive than noncyclical timetables and do not adapt to the changes in demand optimally (peak hours versus off-peak hours). In a rigid cyclical timetable, the only possible solution for these variations is to change the length of trains, adapting the capacity of the system. Some interesting works in this kind of timetables are (Caprara et al., 2001, 2002). They build a periodic timetable in a unidirectional track by representing the problem using a directed graph. The graph is then transformed to an integer programming model that is solved by a lagrangian relaxation procedure.

On the other hand, noncyclical timetables are more relevant when it is not possible (or too expensive) to follow a rigid timetable. Good examples are heavy-traffic and long-distance networks. The headway of train operators and the infrastructure manager will imply that each operator will submit requests for trips (including ideal arrival/departure times) and the infrastructure manager after processing this information will run the timetable procedure (optimization tool) to assign all requests while maximizing the total profit and respecting several operational constraints.

This is one of the most referred railways problems in the literature. Many works are based on the *Periodic Event Scheduling Problem* (PESP) introduced by (Serafini and Ukovich, 1989). The PESP considers the problem of scheduling a set of periodically recurring events under time-windows constraints. An alternative approach is to consider this problem as a special case of the job-shop scheduling problem. In that case, jobs are

trains and resources are tracks (Silva de Oliveira, 2001). This approach is solved by a hybrid algorithm based on Constraint Programming.

Most papers found in literature are actually based on mixed integer programming (MIP) formulations. Continuous variables are added for modeling time (arrival and departure) while integer (binary) variables are included for modeling the order of trains at each component of the network (stations). Some of the works based on this kind of models are (Szpigel, 1973), (Jovanovic and Harker, 1991), (Cai and Goh, 1994), (Carey and Lockwood, 1995), (Higgings et al., 1997).

Rolling Stock Circulation (Operational)

The *Rolling Stock Circulation* problem is an important problem for railways operators all around the world. Acquisition decisions of rolling materials are taken for a long period of time and so, the efficiency of assigning the current vehicles are crucial.

The problem is defined as follows. Given a planned timetable and the expected number of passengers (demand), the model determines an allocation of rolling stock to the services. The model can consider the possibility to add or remove cars to trains in some stations in order to accomplish the demand. Moreover, it is viable to consider splitting or combining two or more trains. Simultaneously, several constraints have to be satisfied such as the order of the cars within the trains and the different natures of the train units. Some of the objectives are the minimization of the expected seat shortages, the maximization of a measure of robustness, or the minimization of the total cost of the rolling stock circulation.

Various versions of this problem have been identified in the literature depending on the nature of the network and the equipment (Caprara et al., 2007). Concerning the nature of the network, two different cases are possible: sparse networks (long distances, long travel times and low frequencies of trains); and dense networks (high frequency of trains and relatively short distances). For sparse networks, the assignment of units is usually determined in detail, *i.e.*, it includes the identification code of each unit. On the contrary, in dense networks the problem is usually anonymous (and so aggregated) because the short distances permit to exchange units easily in order to assure the maintenance events.

Authors that have studied this problem recognize the importance of the work of (Schrijver, 1993) as one of the first papers dealing with this problem. The article considers the problem of minimizing the number of train units of different subtypes for an hourly line while satisfying a given seat demand of passengers. More recently, (M. Peeters, 2008) gives a more complete version of the problem that takes into account the changes in the composition of trains with the aim of having an indication of the robustness of the solution.

For dense networks, (Fioole et al., 2006) is one of the last articles and also gives more recent references that can be consulted to study this problem deeply.

On the other hand, (Cordeau et al., 2001) presents a model for sparse networks. The article considers a time-space network representing all possible consecutive train sequences that available units can make. The LP relaxation of this model is solved by column generation and integer solutions are then obtained heuristically. In this kind of networks, the rolling stock schedule must be adjusted because of the maintenance. (Maróti and Kroon, 2007) presents an integer programming model for the problem of routing train units in order to reach regular preventive maintenance. Finally, for more information, (Maróti, 2006) is a complete work (Ph.D. Thesis) entirely dedicated to this problem.

Crew Scheduling (Operational)

Every train has to be operated by a crew including the machinist and his companions. Train crew scheduling corresponds to the development of a timetable for each member of the crew, to cover a given train timetable. Technical, educational, and legal constraints must be taken into account.

A good definition can be found in (Caprara et al., 1999). The authors define this problem as follows. A planned timetable for the *train services* is given that has to be performed every day for a certain time period. Also, every service is divided in several *trips* that is the minimal route served by the same crew. Additionally, every trip starts at a defined time in a depart station and ends in the arrival station. A *roster* is defined as a cyclical sequence of trips performed by each crew. It is necessary that each trip is performed by one crew. Therefore, the crew scheduling problem consist in finding a set of rosters covering every trip once, satisfying all the operational constraints with the minimum cost, more precisely minimizing the number of crew members needed to perform all the trips for a given planning horizon.

According to (Caprara et al., 2007), crew planning is normally approached in two phases: crew scheduling and crew rostering. The first one consists in selecting the duties to cover all the trips by respecting several kinds of constraints. Complementary, crew rostering obtains the final rosters by sequencing these duties.

The first works on this topic found in the literature come from applications in airlines. In railways, (Caprara et al., 1997) gives a survey for this problem. It should be noted that most of the methods are focused on the first phase (crew scheduling) solving a set covering problem, some of them with additional constraints. For the second phase, only the Italian case is refereed in the literature in railways (Caprara et al., 2007). This work corresponds to (Caprara et al., 1998) that proposes a model using a lagrangian lower bound based on the solution of an assignment problem.

Platform Assignment (Operational)

Previously, we have presented some centralized problems such as rolling stock scheduling, crew scheduling and train scheduling. All these decisions give feasible solutions

valid for the whole network (or an important part). At this point, it is still necessary to transform them to a feasible detailed plan for real operations at each station. The question to answer is how to route the trains through the infrastructure in a specific station limits. This problem is also found in the literature as "train platforming".

Because of the demand, railway operators have to construct train schedules nearing full capacity without considering all the elements of the network (like junctions). Hence, given a timetable (arrival and departure times), the local station limits infrastructure, and the safety system; the routing problem at this level aims at routing all trains optimizing a fitness function, *e.g.*, minimizing the sum of delays.

The input of this problem includes the directions of the trains, their original (ideal) scheduled arrival/departure time, and complete information about the topology of platforms. It should be considered not only trains that pass the station, but also those trains or locomotives coming (going) from (to) the shunting area.

Unlike other decision problems described in this chapter, this problem have not received too much attention by the OR community. (Billionnet, 2003) presents one of the first advances on this topic after the original work of (DeLuca-Cardillo and Mione, 1998). They both formulated the problem as a graph-coloring problem. Another point of view is considered in (Zwaneveld et al., 1996) that models the problem as a node packing problem solved by a procedure based on branch-and-cut.

Shunting (Operational)

Shunting[3] is the process of sorting and parking units of rolling stock that are not necessary in the current timetable. Unused units have to be parked at *shunt yards* otherwise they could interfere with the normal operation of planned trains. This is a local decision because only one station is considered. The objective is to choose the configuration and location of units at the rail yards in such way that it will be necessary to perform the minimal movements possible to accomplish the future movements of the timetable. While the process is called shunting, the corresponding planning problem is often called the *Train Unit Shunting Problem* (TUSP).

There are some aspects to be considered in the model. The first one is that there are different kinds of units according to the use and the length of them. The type and subtype of the unit could limit the possible shunt tracks available, for example electric units can be only parked in tracks with catenaries. Also, some tracks can be accessed by only one side (LIFO: *last in last out*- tracks) and other from both sides (*free* tracks). Additionally, train units of the same subtype can be used interchangeably while other units could be fixed for a particular circulation.

The TUSP is defined as follows: given a railway station, a shunt yard, and a timetable; the Train Unit Shunting Problem consists in matching the arriving and departing units, and parking these shunt units on the shunt tracks, such that the total shunting costs are

[3]Shunting is a term used in Great Britain, the United States equivalent is *switching*

minimal and no crossing occur (Huisman et al., 2005; Freling et al., 2002; Kroon et al., 2007). A crossing is when a train blocks another one during its departure or arrival. Therefore, a feasible solution to the TUSP assigns arriving shunt units to departing ones and to a shunt track. Moreover, if such a track can be approached from both sides, the solution also describes the arrival and departure sides for each train unit parked at the track.

2.2.4 Very Short-Term

The very short-term corresponds to the control of railway operations. We consider problems related to recovering railways operations after disruptions also referred as *Disruption Management in Railway Transportation*. Disruption management in railways is mainly related to timetabling, the rolling stock circulation and the crew scheduling. A very good introduction to this topic can be found in (Jespersen-Groth et al., 2007).

It is possible to observe that all planning problems presented earlier (from strategic to operational decision) try to find optimal or near optimal operation guides, *i.e.*, the best allocation of resources (crew, rolling stock and tracks) that maximizes the efficiency of the operations. As a consequence, any kind of disruption moves away the operation from the optimal. The goal of these problems presented in this section is to react to disturbances in order to return as soon as possible to normal operations. Five different decision problems are identified: *crew scheduling*, *rolling stock circulation*, *timetabling*, *platform assignment*, and *shunting*.

Timetabling (Very Short-Term)

Operational train scheduling (timetabling) gives the itineraries of the trains. That is the corresponding arrival and departure times of every train, for every element of the network. This planning gives a "theoretical scheduling plan". In daily operation, incidents often take place which would usually affect the plan. In some cases, mainly in dense networks, minimal disturbances can imply a large impact in the whole network. The objective in the Timetabling Daily Decision Problem is to minimize the impact of incidents (disruptions).

The first element in this process it is to define a measure of the difference between the theoretical plan and the new provisional one. A very common approach is to consider the total delay: the sum of all differences in the arrival times of the trains in the stations. For example, (Semet and Schoenauer, 2006) stated that in the French railroad company (SNCF) each minute of delay costs 1000 EUROS. Other names for the same problem are railway traffic rescheduling, train dispatching, and train scheduling under disturbances.

This is the main problem treated in this Thesis and will be described deeply in Chapter 3.

Rolling Stock Circulation (Very Short-Term)

Disruption management in rolling stock circulation has to deal with the problem of finding a new assignment of train units after disturbances. As other reactive problems, the main objective is to go back as soon as possible to normal operations. Going back to planned rolling stock assignment guaranties optimal operations.

As the original rolling stock circulation problem, there are commercial and technical constraints to be considered. Firstly, the rolling stock has to assure the operation of the timetable, i.e., all trips must be served by rolling stock units. Additionally, the rolling stock type has to be compatible with the assigned service and trajectory. For example, only passenger cars can be assigned to passenger services, only diesel locomotives can be used in non electrified lines, and trains should not be longer than the shortest platform on its trajectory.

Normally, this problem appears when a new provisional timetable has been constructed after a disrupted situation. Then, the goal is to construct a new rolling stock schedule considering the new situation with a minimum cost and passenger inconveniences. Nevertheless, there are also other situations when it is necessary to reconstruct the plan. For example when rolling stock units fail or when they need urgent or unplanned maintenance services.

A good reference to this problem is given in (Maróti, 2006).

Crew Scheduling (Very Short-Term)

Because of some disruptions, the current list of crew may be affected. The disturbances could be from different sources, as changes in circulations and health problem of the persons. Another important input of this problem is the solution of the previous disruption management problems (timetabling and rolling stock circulation). In fact, crew schedules need to be updated to be sure that all services on the modified timetable will have drivers and will respect the minimal crew requirements.

The objective of this problem is a combination of feasibility, minimization of operational costs, and maximization of stability (Jespersen-Groth et al., 2007). The feasibility aspect is not evident for this problem and so, decisions as cancelation of trains have to be also considered because sometimes there are not enough available crew members to cover all services. Cancelations can be modeled as penalties on the objective function. This last aspect shows that this problem is strongly related to timetabling and rolling stock circulation because cancelations will affect them. Concerning operational cost, the cost of repositioning and transportation for crews should be also considered. Finally, stability is an important objective of any disruption management problem, in this particular case a solution is more stable if the number of modified duties is smaller.

Only few papers deal with this problem. (Jespersen-Groth et al., 2007) show this problem as part of the whole process of disruption management at DSB S-tog and NS (Denmark and Netherlands), while the paper of (Walker et al., 2005) deals with this

problem directly. This paper presents a model divided in two parts: a timetable adjustment and a set partitioning model for the crew schedules.

Platform assignment (Very Short-Term)

This problem deals with the construction of train routes in order to avoid conflicts and delays with the maximum level of detail possible. Normally, this problem appears when a new provisional timetable has been constructed after a disrupted situation and, subsequently, it is necessary to re-route the trains at the local level.

There are only few papers dealing with this problem and one example is given by (Rodriguez, 2007) using constraint programming for the routing and train scheduling at junctions in real-time.

Note that the solution of this problem can be integrated with the solution of timetabling. A very good example is the problem solved in this Thesis, where the construction of a new provisional schedule is accompanied with a new plan of tracks and platforms. Nevertheless, our application only includes the trains going (from/to) or passing a station, but it does not include all shunting movements that can use the tracks and platform considered in the problem.

Shunting (Very Short-Term)

Shunting operations (see Section 2.2.3) are also affected by disturbances and some train units could not finish the day at the location where they were planned (see Section 2.2.4). As a result, the number of units per type at the end of the day could be different than the real one. Therefore, in the next morning, additional trips are needed to go back to the planned situation. The shunting problem in the very short-time deals with disruptions during normal operations. Then, it is not only the process of sorting and parking units of rolling stock that are not necessary in the current timetable, but also utilizing parked units demanded by the new conditions, *i.e.*, after disturbances.

We include this problem in this classification because it seems that OR techniques can improve the performance of the system in case of disruptions as it has been already proved in other problems like rolling stock circulation, timetabling and crew scheduling. Another prospect utilization of this problem is to consider it within a more general integrated recovering system. For example it appears to be appropriate to include shunting aspects in disruption management for rolling stock circulation.

Nevertheless, to the best of our knowledge, there are no papers dealing with this problem. An explication of this fact is given by (Jespersen-Groth et al., 2007). They state that shunting decisions are taken after all other decisions were taken and in case of a disruption; and, in practice, shunting operations are reduced as much as possible. In our opinion, rather than reducing these operations, it would be interesting to adapt them in order to achieve the objective of returning as fast as possible to normal operations.

2.3 Conclusions

In this chapter, we have presented a classification of several planning problems in railway transportation systems. Historically, these problems have been organized according to the time horizon that they consider: strategic, tactical, operational, and very short-term planning. Past surveys have been concentrated from strategic to operational levels where all well-studied classical problems are concentrated, *e.g.*, line planning, train scheduling, crew planning and rolling stock scheduling. In this chapter we include not only these problems, but also very short-term problems that are part of the problems studied in *disruption management in railway transportation*. This is a promising research area as it already is in other fields of Operations Research such as airlines.

This classification has to be understood as a reference to different generic problems in railways and their relationship. As a consequence, it does not represent any particular network/organization/country because of the nature of the infrastructure and the policy of the concerned organizations. Additionally, some recent research papers do not deal with a pure isolated problem defined in a square of Figure 2.1. In fact, real implementations should consider aspects of several individual problems and the repercussions in the rest of the system. This classification helps to identify the immediately affected problems and so a possible integration in both vertical and horizontal dimensions.

Another important conclusion of this chapter concerns the railway rescheduling problem (RRP). The version of the RRP presented in this Thesis is a horizontal integration of two generic decision problems: "time tabling daily" taking some aspects of "detailed platform assignment daily". Using the proposed classification, we also conclude that this problem is strongly related to the disruption management versions of "crew scheduling", "rolling stock", and "shunting". That is, the solution of the RRP has a direct impact on these problems and *vice versa*. Therefore, real implementations of the methods proposed in this Thesis should take into account the repercussions of solutions to the other problems. The next chapter gives a general overview of *Disruption Management* and a deeper description of the RRP.

Chapter 3

Disruption Management and the Railway Rescheduling Problem

Contents

Abstract of the Chapter

This chapter describes the *Railway Rescheduling Problem* (RRP). It includes an introduction to Disruption Management, a complete definition of the RRP detailing the operational aspect to be considered, and a literature review. Finally, the scope of this Thesis is presented using the same aspects as considered in the analysis of the literature.

3.1 Introduction to Disruption Management

For many years Operations Research (OR) methods were used in the planning process obtaining optimal or near-optimal operational plans. Nevertheless, when an operation plan is executed, unexpected problems may occur and the current plan could not remain optimal or even feasible. Figure 3.1 presents different approaches to deal with unexpected events. In this Thesis we are interesting in **Disruption Management** because it is the approach that seems to fit best with the proposed problem in the context of the MAGES project. It is able to cope with all disruptions without knowing them

in advance and a published schedule have to be used as reference in order to return to normal operations as soon as possible.

In this context, unexpected events are often called **disruptions**. Thus, Disruption Management is the process of revising dynamically the original plan and obtaining a new one that reflects the constraints and objectives of the evolved environment while minimizing the negative impact of the disruption. A complete introduction to disruption management can be found in (Clausen et al., 2001; Yu and Qi, 2004).

Approach	**Description**	**Advantages**	**Disadvantages**
Contingency planning (Proactive)	It is completely scenario-based. Thus, for each scenario, it must be identified the options and costs. Finally, documentation for each scenario is constructed and it must be follow in case of disruptions.	It permits to study the consequence of possible future disruptions.	It can only handle a limited set of disruptions. When an unprepared scenario occurs, the performance of the system may worsen.
Stochastic Models (Proactive)	An operational plan is constructed optimal in terms of the average outcome.	Ideally, if all future possibilities are considerate, no real-time re-planning is needed. Is has been approved to be effective in many situations.	The precise probability distributions of possible uncertainties have to be known in advance.
Robust Optimization (Proactive)	It generates a plan that is good for most scenarios and acceptable for the worst.	Probability distributions are not required. It generates a robust plan.	All possible scenarios have to be specified, with their probabilities. The operational plan could be too conservative if the worst-case scenario has a very small probability.
Recoverable Robustness (Proactive/ Reactive)	It integrates recoverability and robustness in a unified framework.	The solution no longer has to be feasible for all possible scenarios, but a recovery phase is allowed to turn an infeasible solution into a feasible one.	Two compatible algorithms need to be developed: a robust algorithm and a recovery algorithm. This approach does not take care of defining details about these algorithms.
Disruption Management (Reactive)	It is a reparation approach that considers both: original objectives and deviation costs, minimizing the negative impact of the disruption.	It is able to cope with all disruptions without knowing them in advance.	It requires a published schedule in advance. Ideally, has to be complemented with a good proactive approach.
Pure Reparation Rescheduling (Reactive)	It is a reconstruction of the plan considering exactly the same objectives and constraint used in the first schedule.	It is a good approach when it is not needed to have a published schedule in advance.	It does not take into account the deviation costs. Thus, it produces suboptimal solutions.

Figure 3.1: *Approaches to uncertainties. The main approach considered in this Thesis is Disruption Management*

Railway Disruption Management is defined as "the joint approach of the involved or-

ganizations to deal with the impact of disruptions in order to ensure the best possible service for the passengers" (Jespersen-Groth et al., 2007). This is done by modifying the current schedules of rolling stock, crew, and trains (timetable); during and after disruptions.

OR methods have proven to be effective for supporting disruption management processes in the airline context. For example Continental Airlines[1] estimates that in 2001 the CrewSolver[2] system helped it save approximately US $40 million for major disruptions (Yu et al., 2003). This is the reason why we believe that OR community can play a more important role to minimize the impact of disruptions and to improve the performance of railway systems.

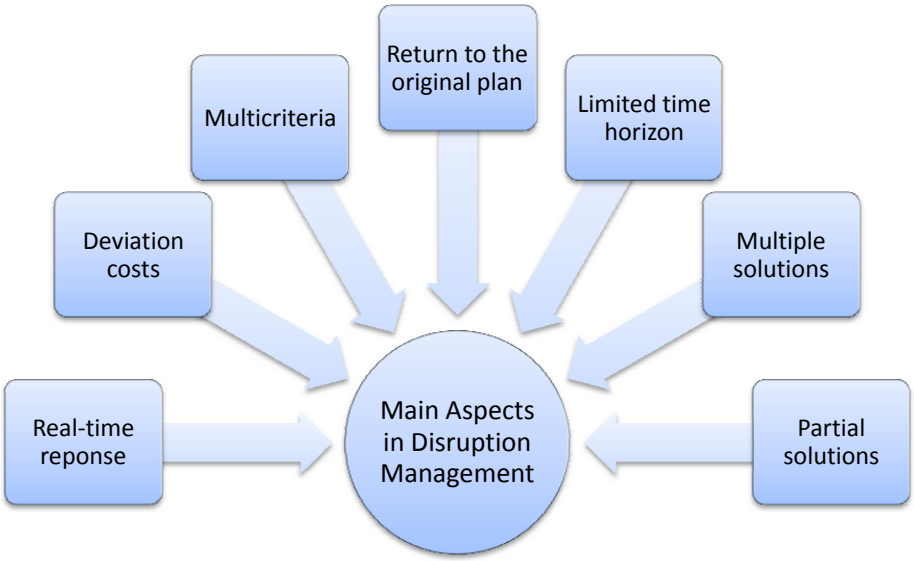


Figure 3.2: *Aspects in Disruption Management*

Figure 3.2 presents the main aspects in disruption management applications:

Real-time: Disruption management is a real-time practice and requires a fast solution when interferences occur. Often, disruptions take place in the execution phase (*i.e.*, only in few particular cases, it is possible to know a disruption in-advance), and a provisional plan has to be provided quickly. Thus, it is necessary to construct efficient algorithms to have good solution in minutes; and in large scale systems, this task could be a very hard job.

Deviation costs: Dealing with disruptions, it is necessary to identify and define a measure of deviation costs. Maybe, it is not easy to assign a real cost to deviation from the original plan, but it has to be taken into consideration in the solution method to assure having a solution as close as possible to the original plan. In real systems, the

[1]Contineal Airlines (http://www.continental.com)

[2]CALEB Technologies developed the CrewSolver decision-support system for Continental Airlines to generate globally optimal, or near optimal, crew-recovery solutions (http://www.calebtech.com/)

plan is connected with actions in different sections of the same organization and with external components that are prepared, even optimized, to work according to the published schedule. As a consequence, a quantification of the difference with the published plan and the provisional one should always be included in the construction of the new provisional plan.

Multicriteria: As a consequence of the deviation costs, a disruption management problem should be modeled as a multicriteria decision making problem. The first criterion to consider is exactly the criterion used to construct the original plan. On the other hand, there is a new criterion: the deviation of the original plan. One of the ways to deal with this is to include an objective function that weights different criteria. The weights should be flexible to study different solutions.

Return to the original plan: In some case, the provisional plan has to converge to the original. This is especially true, when the activities are repeated in the long term as occur in transportation schedules. Then, one of the main goals in disruption management is to return to the original plan as soon as possible.

Disruption management time horizon: After a disruption, the planner has to define an appropriate time point when the plan returns to normal operations. This is the disruption management time horizon. By setting a time windows, the consequences of the disruptions could be limited to a time period. As it is expected, in practice there is a trade-off between the size of the time window and the quality of the solution. That is, a short time window could imply a high recuperation cost.

Multiple solutions: Because of the multiple criteria used and the time limited solution approach, it should be desired to generate multiple solutions. Multiple high-quality solutions with different emphasis (depending on the optimization criteria) or with different constraints are presented to a human (or many) planner.

Partial solutions: As we said before, this is a real time problem and, in order to put into execution immediately, it is possible to ignore some less important constraints or related to future events. Thus, partial solutions are allowed, where a partial solution refers to a solution that does not satisfy all the original constraints. One approach to deal with this is to classify all constraints in two groups: hard and soft constraints. On the one hand, hard constraints must be respected in any feasible solution. In other words, if one of the constraints is violated the solution is automatically considered unfeasible and cannot be implemented. On the other hand, soft constraints could be interpreted as "desirable" requests and they contribute to a penalty if they are violated.

3.2 The Railway Rescheduling Problem (RRP)

Daily trips of trains are usually implemented according to train schedules defined by rail transportation companies at a tactical level. These schedules are valid for long periods of time and are periodically revised (says once a year) to take into account changes in the demand or other factors, *e.g.*, political decisions. Yet, for operational and unpredictable reasons, many small incidents occur day after day. Most of them have a very local impact, but, in some cases, mainly in dense networks, minimal disruptions can spread out through the whole network and affect the complete train schedule significantly. Table 3.1 shows possible incidents and their average duration calculated using historical data of the regional railway network in Asturias, Spain (Adenso-Diaz et al., 1999). These events are difficult to take into consideration when generating the initial schedules. As a consequence, repair procedures are needed to react to incidents and minimize as possible their effects on the performance of the system.

Incident	**Average duration (min)**
Engine breakdown	19.9
Manoeuvering delays in stations	17.5
External causes (weather conditions and others)	13.7
Signal problems	11.8
Catenary problems	10.7
Human Errors in circulation	9.0
Staff shortages	7.0
Incident on the track	5.2
Various	8.9

Table 3.1: *List of possible incidents in rail transportation (Adenso-Diaz et al., 1999).*

The literature on railway systems shows important efforts spent on the development of methods able to generate optimal or near-optimal schedules from scratch. On the contrary, very few papers deal with rescheduling problems. In more general scheduling problems, this is also true. Thus, though the standard three-fields classification scheme proposed by Graham *et al.* (Graham et al., 1979) is commonly used for scheduling problems, no standard classification scheme exists for rescheduling. For this reason, we propose to classify our problem with the framework presented by (Vieira et al., 2003). This framework defines four important dimensions for any kind of rescheduling problem: *rescheduling environment*, *rescheduling strategy*, *rescheduling policy* and *rescheduling method*. Figures 3.3, 3.4 and 3.5 show the complete rescheduling framework.

A rescheduling environment identifies the set of jobs (or trains in our specific case) to be taken into consideration in the schedule (see Figure 3.3). Two different types of environments can be defined: static and dynamic. Static environments consider a finite set of jobs, all known from the beginning. In dynamic environments, one has to deal with an infinite set of jobs, that is, new jobs (unknown in advance) continue arriving in the system after the incident. In this work, we consider a static environment: the set of passengers and stops scheduled in the stations are given and fixed.

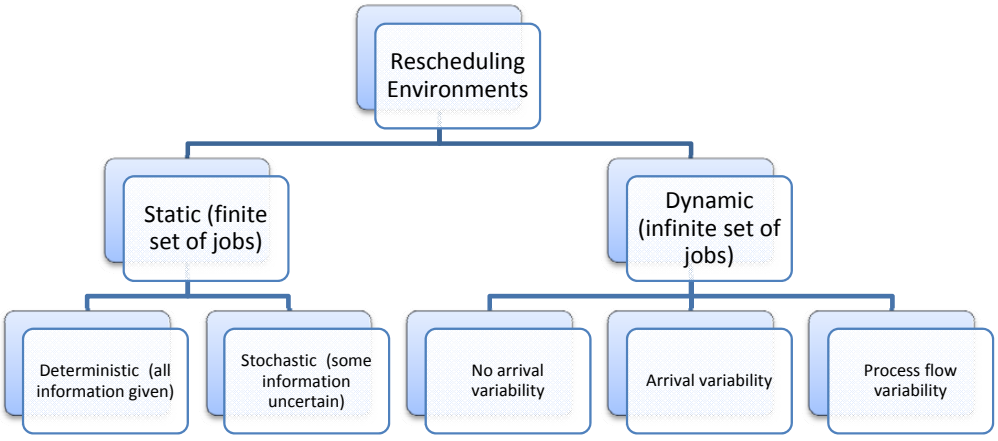


Figure 3.3: *Rescheduling Framework: Environments (Vieira et al., 2003)*

This framework identifies two kinds of static environments: deterministic or stochastic. In deterministic static rescheduling problems, it is assumed that data are not subject to any kind of uncertainty. In contrast, in stochastic environments, some variables are uncertain (e.g., travel times in a railway context). In this Thesis, we consider the environment as deterministic, i.e., it is assumed that all information is known in advance.

The second dimension is the rescheduling strategy (see Figure 3.4). There are two generic rescheduling strategies: dynamic and predictive-reactive. Dynamic strategies have been used for years in rail transportation systems. These strategies do not create a provisional schedule. Instead, trains are dispatched using local information with a decentralized control method (e.g., priority rules). Such control system is usually easy to implement, but provides rarely satisfactory solutions. On the other hand, predictive-reactive strategies update the whole schedule at once. Rescheduling then relies on a global view of the system and is expected to reach better solutions. This is the strategy investigated in this Thesis. It is noticeable that the resulting rescheduling problem is quite different than the problem of generating the initial schedule. Indeed, this latter does not share the same objective, does not have to be computed in real time and often depends on several other factors (technical, economic and even political). Extensive surveys of methods that have been used to generate train schedules can be found in (Assad, 1980; Bussieck et al., 1997; Cordeau et al., 1998; Huisman et al., 2005).

To implement a predictive-reactive rescheduling strategy, a rescheduling policy is needed. In the literature, it is possible to identify two generic types of policies: periodic and event-driven. These policies define how are controlled the plan updates. A periodic policy performs a rescheduling procedure after a predefined time interval. For example, a railway operator could check and reschedule every hour. On the other hand, an event-driven policy reacts after every relevant incident. It is the incident (event) that triggers a change of the plan. A third possibility is to use a mixed policy, called *hybrid*.

The moment in time when the schedule is revised is called the rescheduling point. The rescheduling period is the time between two consecutive rescheduling points. If the

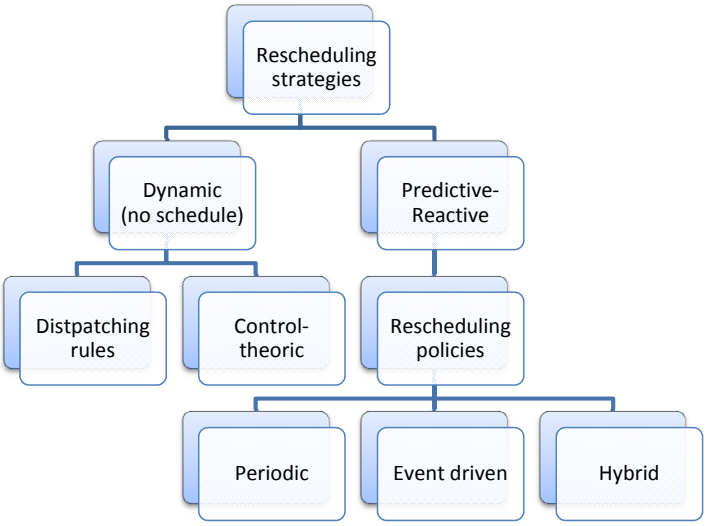


Figure 3.4: *Rescheduling Framework: Strategies (Vieira et al., 2003)*

rescheduling policy is periodic, this value is going to be constant; then, the rescheduling frequency is the inverse of the rescheduling period. The model presented in this Thesis allows including more than one incident and, as a consequence, can be used with the three kinds of policies.

The fourth and last dimension of a rescheduling framework is the solution method (see Figure 3.5). Three generic methods are used to update infeasible schedules: *right-shift rescheduling*, *partial rescheduling* and *complete regeneration*. Right-shift rescheduling is the easiest way to repair the schedule. In a train system, it maintains both the original relative order of trains and the original track assignments. Obviously, the delays caused by the incidents are going to be propagated easily.

Secondly, a partial rescheduling method will consider other possible types of changes (delay some trains to prioritize others, change tracks...), but will be restricted to a subproblem. Some examples of how the subproblem could be constructed are: selecting a subset of trains, selecting just some network components (stations), limiting the time period, or simply combining these possibilities. Actually, the appropriate construction of this subproblem is very difficult by itself. Finally, a complete regeneration of the schedule is also possible. The strong drawback here is the computing time needed to generate the new schedule. The models presented in this Thesis allow using any of these methods.

An important concept related to the solution methodology is the notion of scheduling stability (or, conversely, nervousness). It measures the number of changes implied by the rescheduling. For example, in a "nervous" train system, each rescheduling results in many changes and the system presents little predictability. This can be troublesome for the safety of the system. Successive changes increase risks of human errors, which

might induce grave accidents. This can also be problematic for passengers: last-minute changes of platforms in stations can provoke passengers missing their train. Recapitulating, Table 3.2 summarizes the railway rescheduling problem treated in this Thesis using the framework of (Vieira et al., 2003).

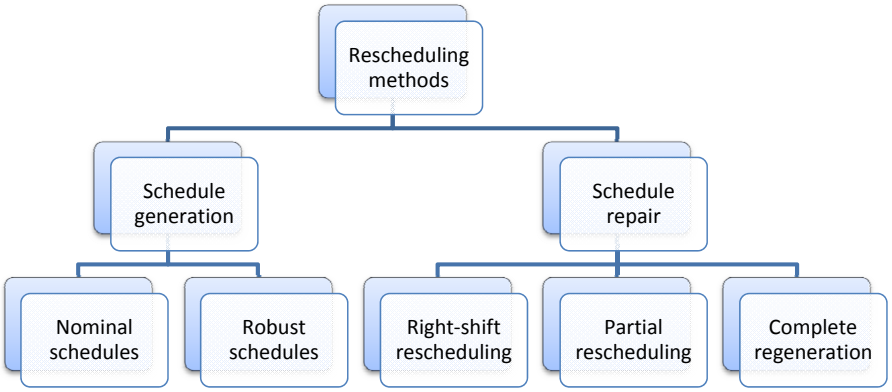


Figure 3.5: *Rescheduling Framework: Methods (Vieira et al., 2003)*

Dimension	Classification
Environment	Static and deterministic
Strategy	Predictive-Reactive
Policy	Periodic, event-driven, hybrid
Method	Right-shift rescheduling, partial rescheduling, complete regeneration

Table 3.2: *Rescheduling framework (Vieira et al., 2003) for the railway rescheduling problem in this Thesis.*

Note that another line of research for limiting impacts of disruptions is the generation of robust schedules. Here, incidents are anticipated during the construction of the schedule. One thus expects to define schedules able to cope with "most" scenarios and also acceptable for the "worst". The problem with robust optimization is the risk to define too conservative schedules and, so, underutilizing the available resources. Several approaches have been developed in the context of manufacturing systems, but very few works exist in rail transportation environments. A recent work can however be found in (Herrmann, 2006). It is also important to remark that the generation of robust schedules is completely compatible with rescheduling, even desirable, and so a complementary line of research.

3.3 Operational Aspects of the RRP

As we stated in the previous chapter, rail transportation is the movement of passengers or freight over railways. There are several distinctive characteristics:

- It is a one-dimension system, *i.e.*, changes of direction are not allowed;

- The adherence between wheels and rails is low; this permits to use a low quantity of energy, either to transport heavy convoys or to move fast passenger trains.

As a consequence of these characteristics, a precise schedule is needed to organize the traffic in the network and several constraints have to be introduced. First, two trains cannot share the same track in opposite directions at the same time. Also, because of the low friction between wheels and railways, trains need a long distance to stop. Therefore, trains using the same track in the same direction have to be sufficiently separated. A solution of these problems is the so called *block system* (see Figure 3.6). At any time, at most one train is allowed per block (for each direction). This principle forms the basis of most railway safety systems. In most systems, blocks are *fixed*, that is, they demarcate a section of track between two fixed points (stations or signals). Usually, however, the length of the blocks is not fixed. Low density lines use blocks that are many kilometers long, while high density lines (as urban trains) can have blocks a few hundred meters long. For high speed lines the rule can be different. For example, a train is only allowed to enter a block if the two next blocks are empty. In order to increase the capacity of the line, some countries prefer using *moving blocks*. An appropriate safe spacing is then constantly maintained between trains. Such system requires to know exactly the position, speed and direction of every train and to calculate in real time the spacing.

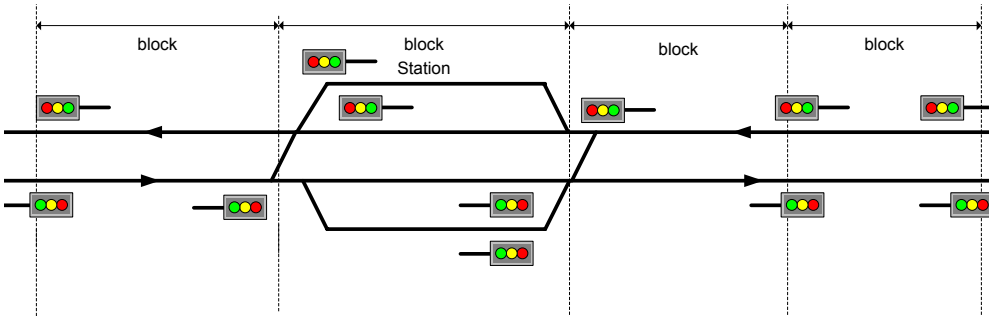


Figure 3.6: *Representation of the block system.*

Another important feature of a railway network is the possibility of driving along the tracks in different directions. In some systems, tracks are constrained to a single direction, while the two opposite directions are authorized in others. Also, networks might combine single and bidirectional lines.

Junctions of lines that permit to perform changes of ways, also represent a difficulty. There are four generic type of junctions: simple branching, triple branching, simple crossing and double crossing junctions (Figure 3.7). For safety reasons, two trains cannot be on the same junction at the same time.

Besides these physical aspects, other technical or commercial constraints have to be considered. The minimum and maximum idle time of a train in a station is imposed by the marketing department. Passengers neither desire short pauses, which might prevent them entering or leaving the train comfortably, nor long pauses that are understood as waste of times. The upper limit can however understandably be adapted in case of connections.

The maximal speed of trains needs also to be considered. It is remarkable that the maximal speed authorized does not necessarily coincide with the maximal speed supported technically by trains. The speed might indeed be limited for many reasons related to maintenance policies, spacing between trains, security, costs, etc. This is for example true in the case of the TGVs (French high speed trains) that could technically run quite faster than they normally do.

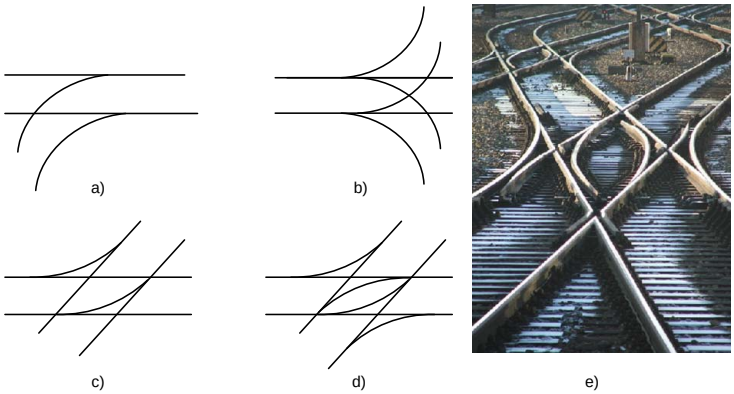


Figure 3.7: *Different type of junctions: a) simple branching, b) triple branching, c) simple crossing junction, d) double crossing junction, e) real double crossing junction (Flickr.com, 2006)*

All the characteristics presented before remain true both for the initial schedule generation and the rescheduling. Some additional features are specific for rescheduling. First, the new schedule has to be reasonably similar to the original plan. Therefore, an appropriate fitness measure is needed to evaluate the difference. A very common measure is to consider the total delay induced by the rescheduling. Nevertheless, other measures are possible like minimizing the changes in the allocation of tracks and platforms. Secondly, the new provisional schedule has to be consistent with the current state of the system. That is, the model has to consider that some trains are already delayed at the moment of generating the new plan.

For solving this problem, there are several possible actions, but only some of them can be completely automatized using computer software. Table 3.3 presents a list of possible actions or decisions that can be considered after disruptions (adapted from (Norio et al., 2005)).

3.4 Analysis of the Literature on the RRP

This section reviews research published in the last years that are related to the problem of minimizing the impact of the deviations of train schedules after disruption. Authors have given different names to this problem depending on several aspect, *e.g.*, the emphasis, the scope, and the origin (North American railways versus European railways). For example in the literature, it is possible to find the *Train Dispatching Problem* (TDP) that deals practically with the same problem as the RRP. The main difference between

Method	Description
Cancelation	To cancel operation of trains
Partial cancelation	To cancel a part of operating area of trains
Extra train	To operate an extra train which is not contained in the original schedule
Extension of train	To extend the operating section of a train
Change of operation schedule	To change the operation schedule of a train-set
Change of track	To change the track of a train in a station or section
Change of departing order	To change the departing order of trains (often, change the station where a rapid train passes a local train)
Change of meeting order	To change the meeting order of trains (either in single track line or at a station where two lines come together)
Change of stop/pass	To make a train stop at a station in which it was originally scheduled to pass
Change of train types	To change the type of a train (to change a rapid train to a local train, etc.)
Change the speed	to change the speed of trains

Table 3.3: *Actions to reschedule trains, adapted from (Norio et al., 2005).*

them is that TDP is focused in solving conflicts to avoid delays and congestions. Thus, one of the main tasks of the TDP is conflict detection and evaluation of possible solutions. On the other hand, the RRP studied in this document allows changing assigned tracks and platform without previous conflicts in order to find a global optimal solution. Another important difference is the horizon, TDP is normally limited to one or two hours in one or few stations while, for the RRP, we are interested in finding good solutions for longer horizons and many stations.

Because of the large number of papers considered in this review, we decide to summarize them with the aim of describing the trends, common aspects, and difference of them rather than studying each of them individually. The papers considered in this analysis are presented in Tables 3.4 and 3.5. The main aspects to classify them are: type of the system, railway topology, and formulation/solution of the problem.

Two **types of system** are identified: centralized and decentralized. Most of the studied papers implement centralized solution methods. Centralized methods obtain better solutions near to the global optimum, while decentralized approaches are faster but clearly suboptimal. Some papers based on decentralized methods are: (Vernazza and Zunino, 1990), (Jia and Zhang, 1994), (Iyer and Ghosh, 1995), (Lamma et al., 1997), and (Missikoff, 1997).

Previous works differ significantly regarding the system **topology**. Some papers are based on railway lines, while other on general networks. A railway line is a set of consecutive stations and segments limited by two extreme stations. On the other hand, a network is a set of railway lines, where one station can be connected directly with

two or more stations. Another important aspect is the type of tracks and platforms: directional/unidirectional and single/parallel tracks. These aspects constitute a very important factor of complexity, as a consequence, in most cases, the representation is simplified. It should be noted that, in general, the most complex models cannot deal efficiently with more than 30 trains. Some papers considering general network topologies are: (Vernazza and Zunino, 1990), (Schaefer and Pferdmenges, 1994), (Iyer and Ghosh, 1995), (Larroche et al., 1996), (Missikoff, 1997), (Lamma et al., 1997), (Vieira et al., 1999), (Ho and Yeung, 2001), (Törnquist and Davidsson, 2002), (Wegele and Schnieder, 2004a,b), (Tornquist, 2006a), (Tornquist and Persson, 2005), (Sahin et al., 2005), (Tornquist, 2006b), (Törnquist, 2007), (Törnquist and Persson, 2007), (D'Ariano et al., 2007), and (D'Ariano et al., 2008).

We also study the **formulation and solution** of the problem. Most of the authors agree in the definition of an objective function: minimization of the delay. Nevertheless, it is possible to observe some variations, for example the introduction of weights to emphasize the delay of some trains or penalizing changes in the resource assignment. Mixed integer linear programming (MIP) formulations and graph theory are the approaches generally used by authors. In general, methods consider continuous variables to represent arrival/departure times of trains, and integer variables are added for modeling the sequence of trains. Within the works based on a MIP, we distinguish (Tornquist, 2006b) and (Törnquist and Persson, 2007) that are the base of the MIP model presented in this Thesis (see Chapter 4).

Regarding the solution methods, there is no dominance of anyone: branch and bound (Wegele and Schnieder, 2004a,b; Tornquist, 2006b; Törnquist and Persson, 2007), metaheuristics such as simulated annealing (Ho and Yeung, 2001; Tornquist and Persson, 2005), tabu search (Ho and Yeung, 2001; Tornquist and Persson, 2005), and genetic algorithms (Ho and Yeung, 2001; Ping et al., 2001; Wegele and Schnieder, 2004a,b) have shown to be effective to solve different versions of this problem. Other families of algorithms are also used such as greedy algorithms (Schaefer and Pferdmenges, 1994; D'Ariano et al., 2007, 2008), simulation heuristics (Sahin et al., 2005), and fuzzy-based algorithms (Vieira et al., 1999). In spite of the fact that constraint programming (CP) has shown to be appropriated for scheduling problems, few papers deal with rescheduling.

Briefly, it is possible to find several applications dealing with scheduling and rescheduling in railways. It is possible to compare the type of systems, the infrastructure and models, but comparing the efficiency of these algorithms is however very difficult, due to the differences in the infrastructure representations and the lack of benchmark instances.

3.5 Scope of the RRP in this Thesis

The scope of the RRP in this Thesis is defined using the aspects discussed in the previous sections: type of the system, topology, formulation & solution, and actions (decisions). Figure 3.8 sums up all these aspects.

First, our system is centralized, that is a new provisional schedule is constructed for several stations and trains by minimizing the global cost. The justification is that decentralized strategies could imply suboptimal, less robust and undependable solutions. Undependable solutions imply that there are several independent solutions, says one per station, that cannot be connected. We take over the difficulties in handling large problems and scalability for centralized system. This is the reason why, this Thesis is mainly focused on setting up efficient solution methods in order to tackle these difficulties.

Concerning the topology of the system, our model works with general networks and railway lines, accepts bidirectional and several parallel tracks in sections and parallel platforms in stations. Some additional real-life characteristics are also considered: variable-speed, unplanned stops, extra time for acceleration & brake, and connection of trains.

Third, regarding the formulation and solution, we have proposed a MIP and a CP models that permit to develop both: exact and heuristic procedures. The models and solution methods are presented in the next part of this Thesis (Chapters 4, 6 and 5).

The last aspect is the set of actions (decisions) taken by the models. These actions are very important to define the decision variables of the models adequately. It is important to comment that the definition of decision variables depends mainly on the nature of the model. Thus, order of trains and assignment of tracks will be carried out by binary integer variables in the MIP model. In contrast, CP permits to consider more natural ways of modeling decision variables. The actions are then assured by the definition of constraints and by the interpretation of the values of decision variables in the solution.

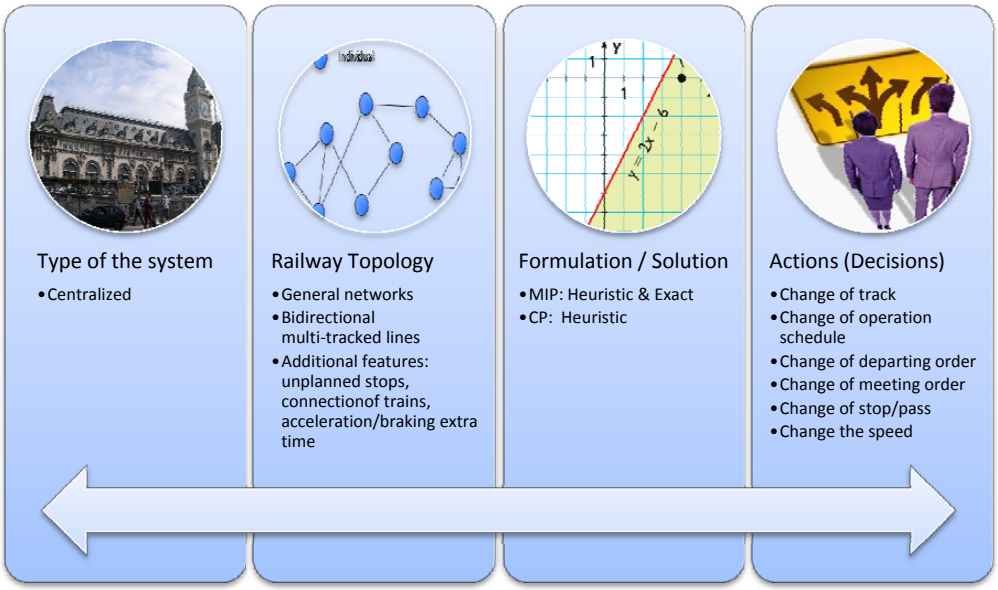


Figure 3.8: *Scope of this Thesis: type, topology, formulation & solution, and actions*

3.6 Conclusions

We began this chapter introducing Disruption Management (DM) as the process of revising the original plan to obtain a new one while minimizing the negative impact of disruptions. We have identified the key aspects to be considered in any DM application that are employed not only in the definition of the problem, but also when designing the models, solution methods, and tests. We have described deeper the railway rescheduling problem (RRP) as the problem of finding a new provisional timetable after disrupted operations considering the operational aspects presented in this chapter.

A literature review was also presented. Several papers dealing with scheduling and rescheduling in rail transportation were found. It is possible to compare the type of systems, the infrastructure and models, but comparing the efficiency of these algorithms is however very difficult, due to the differences in the infrastructure representations and the lack of benchmark instances.

Another important result of this chapter is the definition of the scope of this Thesis considering the type of the system, topology, formulation & solution, and the actions of the problem. This scope is taken into account in the next chapters where different models and solution methods for this problem will be developed.

Title	Year	Reference
On-line timetable re-scheduling in regional train services	1999	(Adenso-Diaz et al., 1999)
Scheduling railway traffic at a construction site	2002	(Brucker et al., 2002)
A study of the expert system for train dispatching	1993	(Cai and Wang, 1993)
From timetabling to train regulation - A new train operation model	2005	(Chang and Chung, 2005)
Hybrid simulation for resolving resource conflicts in train traffic rescheduling	1998	(Cheng, 1998)
A Constraint-Based Interactive Train Rescheduling Tool	2002	(Chiu et al., 2002)
Conflict Resolution and Train Speed Coordination for Solving Real-Time Timetable Perturbations	2007	(D'Ariano et al., 2007)
Assessment of flexible timetables in real-time traffic management of a railway bottleneck	2008	(D'Ariano et al., 2008)
An algorithm for train rescheduling using rescheduling pattern description language R	2006	(Hirai et al., 2006)
Railway junction traffic control by heuristic methods	2001	(Ho and Yeung, 2001)
DARYN, A Distributed Decision-Making Algorithm for Railway Networks: Modeling and Simulation	1995	(Iyer and Ghosh, 1995)
Disruption management in passenger railway transportation	2007	(Jespersen-Groth et al., 2007)
Distributed intelligent railway traffic control based on fuzzy decision making	1993	(Jia and Zhang, 1994)
Autonomous Decentralized Traffic Management System	2000	(Kitahara et al., 2000)
SEPIA: a real-time expert system that automates train route management	1996	(Larroche et al., 1996)
A distributed constraint-based scheduler	1997	(Lamma et al., 1997)
An object-oriented approach to an information and decision support system for railway traffic control	1997	(Missikoff, 1997)
Heuristic approach to train rescheduling	2007	(Mladenović and Čangalović, 2007)
Rescheduling method based on distributed cooperative problem solving model	1996	(Nobuyuki et al., 1996)

Table 3.4: *Papers related to the railway rescheduling problem (I)*

Title	Year	Reference
Train Rescheduling Algorithm Which Minimizes Passengers Dissatisfaction	2005	(Norio et al., 2005)
Study on intelligent train dispatching	2001	(Ping et al., 2001)
Distrain: A simulation tool for train dispatching	2005	(Rebreyend, 2005)
A Dispatching Tool for Railway Transportation	2006	(Rebreyend, 2006)
A constraint programming model for real-time train scheduling at junctions	2007	(Rodriguez, 2007)
Railway traffic control and train scheduling based on inter-train conflict management	1999	(Şahin, 1999)
New approaches for the train dispatching problem	2005	(Sahin et al., 2005)
An expert system for real-time train dispatching	1994	(Schaefer and Pferdmenges, 1994)
An efficient memetic, permutation-based evolutionary algorithm for real-world train timetabling	2005	(Schoenauer and Semet, 2005)
On the benefits of inoculation, an example in train scheduling	2006	(Semet and Schoenauer, 2006)
Fuzzy algorithm for real-time train dispatch and control	2005	(Tazoniero et al., 2005)
A Multi-Agent System Approach to Train Delay Handling	2002	(Törnquist and Davidsson, 2002)
Railway traffic disturbance management - An Experimental analysis of disturbance complexity, management objectives and limitations in planning horizon	2007	(Törnquist, 2007)
N-tracked railway traffic re-scheduling during disturbances	2006	(Törnquist and Persson, 2007)
A distributed intelligence methodology for railway traffic control	1990	(Vernazza and Zunino, 1990)
Railway dispatch planning and control	1999	(Vieira et al., 1999)
Automated dispatching of train operations using genetic algorithms	2004	(Wegele and Schnieder, 2004a)

Table 3.5: *Papers related to the railway rescheduling problem (II)*

Part II

Modeling and Solving the Railway Rescheduling Problem

Chapter 4

Mixed Integer Programming Approaches

Contents

Abstract of the Chapter

In this chapter, some mixed integer linear programming (MIP) approaches for the railway rescheduling problem are presented. The MIP formulation models bidirectional multi-tracked lines for general networks, considering some additional aspects such as acceleration & braking time, combination of trains, and unplanned stops. The main solution method presented in this chapter is based on *local branching* cuts added to the model in order to reduce the feasible region to solutions "near" to the original schedule. This chapter also includes computational experiments in two different networks located in France and Chile. The results show that these methods are viable in practice for real-life instance.

4.1 Introduction

In this chapter we present a mixed integer programming (MIP) formulation for the railway rescheduling problem (RRP). In the previous chapter we have described the RRP as the problem to find a new provisional timetable after disrupted operations respecting several operational and marketing constraints and minimizing the total impact of disruptions (see Chapter 3).

This MIP formulation responds to the scope of the RRP in this Thesis (see Section 3.5). First and foremost, our system is centralized, that is a new provisional schedule is constructed for several stations and trains. Secondly, the formulation works with general networks and lines accepting bidirectional parallel tracks. Third, some additional real-life characteristics are also considered: variable-speed, unplanned stops, extra time for accelerating & braking, and connection of trains.

Lastly, it is also very important to define the real actions (or decisions) that this model helps to take. A complete list of all possible actions taken to reschedule trains after disturbances are exposed in Table 3.3. It may be noticed that only some of them are feasible to be automatized by computer software. In particular, we consider only the decisions that do not require changing the assignment of resources other than tracks. For example, extension of a train, extra trains, and cancelation of trains involve important changes in rolling stock assignment, and so they are not considered in this model. It should be noted also that the actions taken into consideration for this MIP formulation were defined according to the objectives of the MAGES project.

Method	**Description**
Change of track	To change the track of a train in a station or section
Change of operation schedule	To change the operation schedule of a train-set
Change of departing order	To change the departing orders of trains
Change of meeting order	To change the meeting orders of trains (either in single track line or at a station where two lines come together)
Change of stop/pass	To make a train stop at a station which it was originally scheduled to pass
Change the speed	to change the speed of trains

Table 4.1: *Actions to reschedule trains considered in the MIP formulation.*

Table 4.1 presents the actions considered in this MIP formulation. All these actions are associated to the definition of the decision variables. Thus, "changes of track" are assured by binary integer variables modeling the assignment of tracks and platforms. "Changes of departing (meeting) order" are evaluated using binary variables that determine if trains need to be rescheduled or need to keep the same order as the original plan. The special decision of stopping or passing in a station is also modeled by binary variables. These variables permit to adapt arrival and departure times considering a minimal extra time because of acceleration and braking. The last actions, "changes of speed", are assured by (real) decision variables that model the time of arrival and de-

parture events. These variables are constrained in order to guarantee that trains do not overpass the speed limit of sections and locomotives.

The rest of the chapter is organized as follows. Section 4.2 describes the MIP formulation deeply. First, we develop a base model assuming that the composition of the sections and blocks is completely known (see Section 4.2.1). This model is then extended in Section 4.2.2 to the special case when the exact composition of blocks is not available.

Because of the impossibility of solving real-size instances of this formulation using MIP solvers, some solution methods are developed in Section 4.3. The fastest approach is right-shift rescheduling presented in Section 4.3.1. It keeps the same order of trains and does not allow changes in the assignment of tracks. These facts are modeled as fixation of integer variables that permits to solve the problem quickly. Nevertheless, the quality of the solution is not good enough because of the extreme propagation of delays. Because its speed, this approach shows to be appropriate to calculate initial solutions that are exploited by the other solution methods developed in this Thesis.

The second method is described in Section 4.3.2. This approach is based on *local branching* cuts added to the model in order to reduce the feasible region finding solutions "near" to the original schedule. This method is then extended to an iterative approach in Section 4.3.3 where the bounds of local branching cuts are updated in order to increase progressively the search space. Section 4.5 evaluates the methods in different networks located in France and Chile. Finally the conclusions are summed up in the last section of this chapter.

4.2 MIP Formulation for the RRP

In this section we present two MIP formulations: a base model and an extended version. The first one models the fixed block system explicitly. The principle of this kind of system is that trains cannot use a block section when the block is occupied by another train. A more detailed definition of the block system and other operational aspects in this problem were presented in the previous chapter (see Section 3.3).

Unfortunately, the information about the exact composition and position of blocks is not always available. The second MIP formulation presents some modifications of the base model in order to take into account the block system implicitly. Therefore, we consider sections (set of blocks) rather than blocks individually. As a consequence some constraints are adapted to assure the minimal headway[1] in both: at the beginning and at the end of the section.

[1]Headway: The time interval between two successive trains.

4.2.1 Base Model

The mathematical formulation presented in this chapter is an extension of the model presented by (Törnquist and Persson, 2007) and uses basically the same notation. There are two main differences between both models. First, this formulation considers explicitly the fact that unplanned stops will change the minimal and maximal allowed travel time because of acceleration and braking. The model presented by (Törnquist and Persson, 2007) is based on the block system where only one train is admitted in a block at a given time. More information about the block system and other operational aspects of this problem can be found in Section 3.3.

The second difference is the modification of some constraints to admit more than one train in the same section running in the same direction. That is valid when a section is composed of several blocks whose exact number and positions are unknown. In that case some modifications are made to assure the spacing not only at the beginning but also at the ending of the section (see Section 4.2.2).

In order to model the block system, a set of sections F and a set of blocks B are defined (see Figure 3.6). Let indices s and j respectively be associated to sections and blocks. The set of blocks in a section s is denoted by G_s. Without loss of generality, we consider that a station is a particular section with only one block. Every section s has a set of parallel tracks R_s. As a consequence, a block j also has a set of parallel tracks $P_j = \{1, \ldots, p_j\}$.

Let T be the set of trains. A *train* corresponds to one or more items of rolling stock coupled together (at least one is a locomotive) that have to fulfill a predefined set of stations in a given order. Note that some authors rather call trains *services* or *circulations*. Let index i be related to trains. The base of the schedule is a set of events E. An event corresponds to a block request by a train. Let index k be associated to events. The set of events is divided in two subsets $E = E_{st} \bigcup E_{sec}$, where E_{st} is the set of events occurring in stations, and E_{sec} is the set of events occurring in other sections. Let parameter $r_k = 1$, if event k occurs in a station and $r_k = 0$ if the event occurs in another section. Also, let o_k^E be the sense (direction) of the event k, for example: north or south (even or odd).

We define $K_i \subseteq E$ as the ordered set of events of train i. In this definition, *ordered set* means the event order in the original (non-disrupted) schedule. The first event of train i is k_i^0 and the last event is $k_i^{n_i}$. Thus, n_i corresponds to the last event of train i. Additionally, $I_{s,i}$ is the ordered set of events for train i in section s. Two other sets are needed. First, $L_j \subseteq E$ is the ordered set of the events associated with block j. Let m_j be the number of events of set L_j. This set is very important to define an adequate number of schedule variables. Indeed, only trains sharing a block in their schedules have to be safely spaced. Additionally, the fact that this set is *ordered* will help to avoid symmetries in the model. Secondly, H_s corresponds to the set of trains using section s.

To model safety distance for blocks, spacing parameters are introduced. Δ_j^M corresponds to the amount of time between the moment when a train leaves block j and the other one enters it. Δ_j^F is the minimal interval of time separating two trains arriving to block j in the same sense, and using the same track. Without loss of generality, we

assume that these values are constant in a same block and are valid for every kind of trains.

There are two types of tracks: unidirectional and bidirectional. Unidirectional tracks only accept flow in one direction while bidirectional tracks admit it in both senses. For a given track t, we define $c_t = 0$ if track t is unidirectional, and $c_t = 1$ if track t is bidirectional. Additionally, for every unidirectional track t, parameter o_t^T defines the direction.

It is not always possible for a train in a section to change to another track. In that case, parameter $f_t = 1$ indicates that every event using the track t is not allowed to change to another track. In contrast, if $f_t = 0$, some changes are allowed for certain compatible tracks. For a given track t and a particular event k, let $N_{k,t}$ be the set of incompatible tracks. That is, if an event k is using the track t, then the next event of the same train, event $k+1$, cannot use any track belonging to $N_{k,t}$.

The last set to be defined corresponds to connections of trains. This set is denoted C. Thus, $(k, \hat{k})$ belongs to C, if event k has to arrive (to the station associated with k) before $\hat{k}$. Parameters $g_{k,\hat{k}}^{min}$ and $g_{k,\hat{k}}^{max}$ correspond to the minimal and maximal connection times from event k to event $\hat{k}$.

Summarizing, the indices are:

i : Index for trains.
s : Index for sections.
j : Index for blocks.
k : Index for events.
t : Index for tracks.

The data sets are:

T : Set of trains.
F : Set of sections.
B : Set of blocks.
E : Set of events.
E_{sec} : Set of events in sections.
E_{st} : Set of events in stations.
R_s : Set of tracks for section s.
G_s : Set of blocks for section s.
H_s : Set of trains using section s.
$I_{s,i}$: Set of events for train i in section s.
K_i : Set of events for train i.
L_j : Set of m_j events for block j.
P_j : Set of p_j tracks for block j.
$N_{k,t}$: Set of tracks t_{incomp} that are not allowed to be used after event k in track t.
C : Set of pair of events $(k, \hat{k})$, where event k has a connection with event $\hat{k}$.

Other parameters are:

p_j	:	Number of parallel tracks of block j.
r_k	:	=1, if event k occurs in a station and $r_k = 0$ otherwise
o_k^E	:	Sense (direction) of event k.
n_i	:	Last event of train i.
m_j	:	Number of events of set L_j.
Δ_j^M	:	Min. interval between a train leaves block j and the other one enters it
Δ_j^F	:	Min. interval separating two trains arriving to block j (same track, same sense)
c_t	:	=1 if track t is bidirectional and 0 otherwise.
o_t^T	:	Sense (direction) of track t.
f_t	:	=1 if all events using track t are not allowed to change to another track, 0 otherwise.
$b_k^{initial}$	:	Starting time of event k in the original plan.
$e_k^{initial}$	:	Ending time of event k in the original plan.
$b_k^{incident}$	:	Real starting time of event k after incidents.
$e_k^{incident}$	:	Real ending time of event k after incidents.
d_k^{min}	:	Minimal stopping or running time of event k.
d_k^{max}	:	Maximal duration of event k.
d_k^{brake}	:	Minimal extra time for braking taking by the event precedent of k.
d_k^{acc}	:	Minimal extra time for accelerating taking by the next event after k.
M	:	Large constant.
h_k	:	=1 is event k is a planned stop in the original schedule and 0 otherwise

In case of disruptions, the current state of the system is modeled with some additional parameters. First, let $b_k^{initial}$ and $e_k^{initial}$ be the starting and ending times of event k in the original schedule. Second, let parameters $b_k^{incident}$ and $e_k^{incident}$ be the real starting and ending times of an event k, using the information about the incident. All the events strictly before the incident are not considered in the model since they are not relevant.

Decision variables related to time are:

x_k^{begin}	: starting time of event $k \in E$;
x_k^{end}	: ending time of event $k \in E$;
z_k	: lateness of event $k \in E$.

Considering the speed of trains and commercial minimal stopping times in stations, another parameter denoted d_k^{min} is used. In the first place, if event $k \in E_{st}$, d_k^{min} is the minimal stopping time of the train in the corresponding station. Secondly, if event $k \in E_{sec}$, d_k^{min} is the minimal running time depending on the maximal speed of the train. Similarly, we also define d_k^{max} as the maximal duration of event k. If no upper limit is imposed, $d_k^{max} = M$ where M is a large constant.

In the original schedule, some events are planned stops. In that case, d_k^{min} already includes the braking and accelerating times. Nevertheless, for additional (unplanned)

stops, it is necessary to adjust these values. Parameter $h_k = 1$ if event k is a planned stop in the original schedule, $h_k = 0$ otherwise. Let parameter d_k^{brake} be the minimal extra time for braking taken by the previous event before to stop. Similarly, the same train will need also an additional time to accelerate. Let parameter d_k^{acc} be the minimal extra time for accelerating taken by the next event after to stop. Finally, unplanned stops are modeled using the followings decision variables:

$$y_k = \begin{cases} 1, & \text{if an unplanned stop is added during event } k,\ k \in E\ ; \\ 0, & \text{otherwise.} \end{cases}$$

Another important aspect of this model is the assignment of trains to tracks and platforms. Thus, we define the following decision variables:

$$q_{kt} = \begin{cases} 1, & \text{if event } k \text{ uses track } t, \text{ where } k \in L_j, t \in P_j, j \in B; \\ 0, & \text{otherwise.} \end{cases}$$

Using the same notation, some parameters are used to penalize changes of tracks and platforms. Let $q_{kt}^0 = 1$, if event k uses track t in the original schedule, $q_{kt}^0 = 0$ otherwise.

In order to determine the order of events, it is necessary to introduce some disjunctive variables used frequently in scheduling problems:

$$\gamma_{k\hat{k}} = \begin{cases} 1, & \text{if event } k \text{ occurs before event } \hat{k}, \text{ where } k, \hat{k} \in L_j, \\ & j \in B, \text{ and } \hat{k} \text{ is any event following event } k \\ & \text{with respect to the original schedule;} \\ 0, & \text{otherwise.} \end{cases}$$

$$\lambda_{k\hat{k}} = \begin{cases} 1, & \text{if event } k \text{ is rescheduled to occur after event } \hat{k}, \\ & \text{where } k, \hat{k} \in L_j, j \in B, \text{ and } \hat{k} \text{ is any event} \\ & \text{following event } k \text{ with respect to the original schedule;} \\ 0, & \text{otherwise.} \end{cases}$$

It is necessary to clarify the differences between variables γ and λ. On the one hand, variables γ are used to keep the same order of trains than the original schedule. On the other hand, variables λ takes the value 1, if the original order of train is altered, *i.e.*, they are rescheduled. As a consequence, if there are no incidents and the order of trains is kept intact to the original schedule, all variables λ will have the value 0.

Furthermore, we define parameters $\gamma_{k\hat{k}}^0 = 1$, if event k occurs before event $\hat{k}$ in the original schedule and $q_{kt}^0 = q_{\hat{k}t}^0 = 1, t \in P_j$, with $k, \hat{k} \in L_j, j \in B$, $\gamma_{k\hat{k}}^0 = 0$ otherwise.

The objective function in our model is composed of different costs. We define two parameters used for penalizing delays: *CD* (Cost of Delay) and *CFD* (Cost of Final Delay). These correspond to the cost of every time unit of delay at every planned stop and the cost of every time unit of delay in the last station respectively. Also, we utilize two different costs to penalize changes of tracks/platforms: *CCT* (Cost of Change a

Track) and *CCP* (Cost of Change a Platform). These costs are a penalization associated with every change of track/platform compared with the original schedule. Finally, *CUS* is the unitary cost incurred by each unplanned stop.

The complete MIP model is:

$$\text{Minimize}: CD \sum_{\substack{i \in T \\ k \in K_i}} h_k z_k + CFD \sum_{i \in T} z_{k^{n_i}} + CCT \sum_{\substack{t \in P_j, j \in B \\ k \in L_j \cap E_{sec}}} q_{kt}(1 - q^0_{kt}) + CCP \sum_{\substack{t \in P_j, j \in B \\ k \in L_j \cap E_{st}}} q_{kt}(1 - q^0_{kt}) + CUS \sum_{\substack{k \in E_{st} \\ h_k = 0}} y_k \tag{4.1}$$

Subject to:

$$x^{end}_k = x^{begin}_{k+1} \qquad \forall k \in K_i, i \in T : k \neq k^{n_i} \tag{4.2}$$

$$x^{end}_k \geq x^{begin}_k + d^{min}_k + d^{brake}_k y_{k+1} + d^{acc}_k y_{k-1} \qquad \forall k \in K_i, i \in T : r_k = 0 \tag{4.3}$$

$$x^{end}_k \geq x^{begin}_k + d^{min}_k \qquad \forall k \in E : h_k = 1 \wedge r_k = 1 \tag{4.4}$$

$$x^{end}_k \geq x^{begin}_k + y_k d^{min}_k \qquad \forall k \in E : h_k = 0 \wedge r_k = 1 \tag{4.5}$$

$$x^{end}_k \leq x^{begin}_k + y_k d^{max}_k \qquad \forall k \in E : h_k = 0 \wedge r_k = 1 \tag{4.6}$$

$$x^{end}_k \leq x^{begin}_k + d^{max}_k \qquad \forall k \in E : b^{incident}_k = e^{incident}_k = 0 \tag{4.7}$$

$$x^{begin}_k \geq b^{initial}_k \qquad \forall k \in E : h_k = 1 \tag{4.8}$$

$$x^{begin}_k \geq b^{incident}_k \qquad \forall k \in E : b^{incident}_k > 0 \tag{4.9}$$

$$x^{end}_k \geq e^{incident}_k \qquad \forall k \in E : e^{incident}_k > 0 \tag{4.10}$$

$$x^{begin}_k - b^{initial}_k \leq z_k \qquad \forall k \in E \tag{4.11}$$

$$\sum_{t \in P_j} q_{kt} = 1 \qquad \forall k \in L_j, j \in B \tag{4.12}$$

$$q_{kt} + q_{\hat{k}t} - 1 \leq \lambda_{k\hat{k}} + \gamma_{k\hat{k}} \qquad \forall k, \hat{k} \in L_j, t \in P_j, j \in B : k < \hat{k} \tag{4.13}$$

$$\lambda_{k\hat{k}} + \gamma_{k\hat{k}} \leq 1 \qquad \forall k, \hat{k} \in L_j, j \in B : k < \hat{k} \tag{4.14}$$

$$x^{begin}_{\hat{k}} - x^{end}_k \geq \Delta^M_j \gamma_{k\hat{k}} - M(1 - \gamma_{k\hat{k}}) \qquad \forall k, \hat{k} \in L_j, j \in B : k < \hat{k}, o^E_k \neq o^E_{\hat{k}} \tag{4.15}$$

$$x^{begin}_{\hat{k}} - x^{end}_k \geq \Delta^F_j \gamma_{k\hat{k}} - M(1 - \gamma_{k\hat{k}}) \qquad \forall k, \hat{k} \in L_j, j \in B : k < \hat{k}, o^E_k = o^E_{\hat{k}} \tag{4.16}$$

$$x^{begin}_k - x^{end}_{\hat{k}} \geq \Delta^M_j \lambda_{k\hat{k}} - M(1 - \lambda_{k\hat{k}}) \qquad \forall k, \hat{k} \in L_j, j \in B : k < \hat{k}, o^E_k \neq o^E_{\hat{k}} \tag{4.17}$$

$$x^{begin}_k - x^{end}_{\hat{k}} \geq \Delta^F_j \lambda_{k\hat{k}} - M(1 - \lambda_{k\hat{k}}) \qquad \forall k, \hat{k} \in L_j, j \in B : k < \hat{k}, o^E_k = o^E_{\hat{k}} \tag{4.18}$$

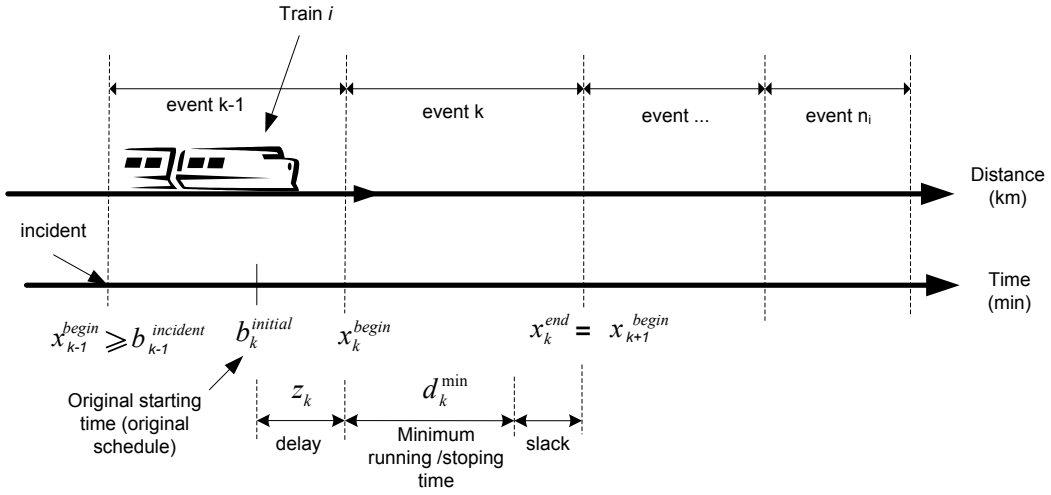


Figure 4.1: *Illustration of time variables, parameters and constraints*

$$q_{kt} = q_{k+1,t} \qquad \forall k \in I_{s,i} : k \neq k_i^n, i \in H_s, t \in R_s : f_t = 1, s \in F \tag{4.19}$$

$$q_{kt} = 0 \qquad \forall k \in L_j, t \in P_j, j \in B : o_t^T \neq o_k^E \wedge c_t = 0 \tag{4.20}$$

$$q_{kt} + \sum_{t_{inc} \in N_{k,t}} q_{k+1,t_{incomp}} \leq 1 \qquad \forall k \in I_{s,i} : k \neq k_i^n, i \in H_s, t \in R_s : f_t = 0, s \in F \tag{4.21}$$

$$x_{\hat{k}}^{end} \geq x_k^{begin} + g_{k\hat{k}}^{min} \qquad \forall (k, \hat{k}) \in C \tag{4.22}$$

$$x_{\hat{k}}^{end} \leq x_k^{begin} + g_{k\hat{k}}^{max} \qquad \forall (k, \hat{k}) \in C \tag{4.23}$$

$$y_k = 0 \qquad \forall k \in E : h_k = 1 \wedge r_k = 1 \tag{4.24}$$

$$y_k = 0 \qquad \forall k \in E : r_k = 0 \tag{4.25}$$

$$x_k^{begin}, x_k^{end}, z_k \geq 0 \qquad \forall k \in E \tag{4.26}$$

$$\gamma_{k\hat{k}}, \lambda_{k\hat{k}} \in \{0, 1\} \qquad \forall k, \hat{k} \in L_j, j \in B : k < \hat{k} \tag{4.27}$$

$$q_{kt} \in \{0, 1\} \qquad \forall k \in L_j, t \in P_j, j \in B \tag{4.28}$$

$$y_k \in \{0, 1\} \qquad \forall k \in E \tag{4.29}$$

Objective function (4.1) corresponds to the total rescheduling cost. This cost is equal to the sum of delays for all planned stops, plus the sum of delays of trains in their last station, plus the total cost of changing tracks/platforms, plus the total cost of unplanned stops.

Set of constraints (4.2) ensures the continuity of events. When a train leaves a block, it immediately enters the next one (see Figure 4.1). For every event in sections, con-

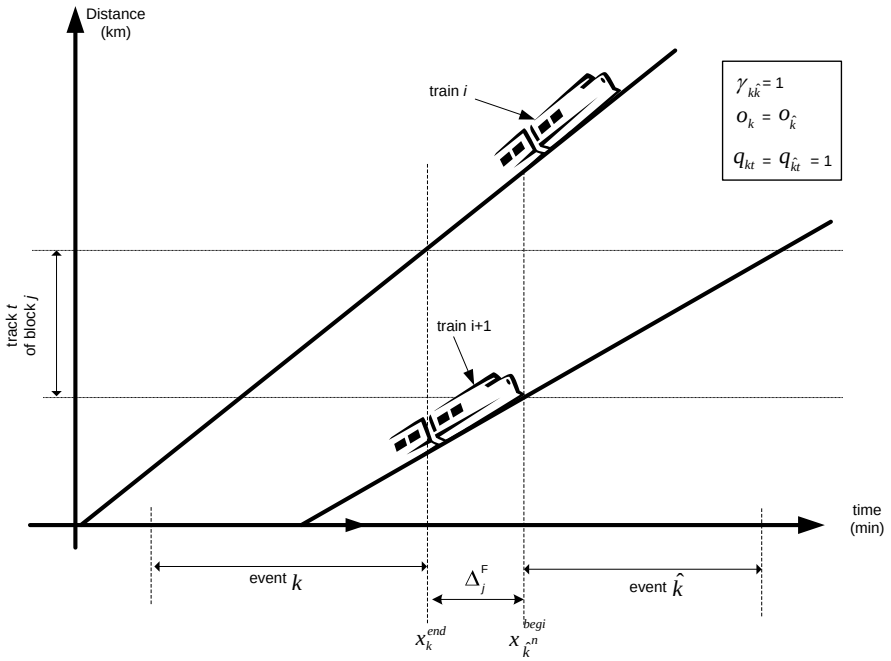


Figure 4.2: *Time-distance diagram: An example for variables and constraints in the MIP model*

straints (4.3) guarantee the minimum traveling time d_k^{min}. If the next (resp. previous) event of a train is an unplanned stop, the minimal travel time has to be increased because of an extra time needed for braking (resp. accelerating).

For a planned stop in a station (*i.e.*, $h_k = 1 \wedge r_k = 1$) the minimal stopping time is guaranteed by constraints (4.4). In contrast, for events that do not have a planned stop in a station (*i.e.*, $h_k = 0 \wedge r_k = 1$), constraints (4.5, 4.6) state that the minimal and maximal stopping time is only valid if they stop.

For events without incidents (*i.e.*, $b_k^{incident} = e_k^{incident} = 0$), a maximal duration is defined using constraints (4.7). Note that these constraints are not applied to other events because the condition could be invalidated by the incident. Constraints (4.8) ensure that trains with a planned stop cannot leave a station before the time originally scheduled.

As it has been previously commented, it is necessary to respect the current status of circulations. Consequently, constrains (4.9, 4.10) inject the incidents into the model. These constraints force new starting (ending) times to be greater or equal to the time of the first indication of the incident (see Figure 4.1).

Set of constraints (4.11) is used to define the delay as the difference between the starting time in the new provisional schedule and the starting time in the initial (non-disrupted) schedule. Actually, the delay for events in stations is calculated using the

difference of the arrival times. In contrast, some authors use the departing times to measure the delay.

Constraints (4.12) are introduced to ensure that one train uses strictly one track in every block. If two events use the same track, constraints (4.13) check that either $\gamma_{k\hat{k}}$ or $\lambda_{k\hat{k}}$ takes value 1. Constraints (4.14) impose that $\gamma_{k\hat{k}}$ and $\lambda_{k\hat{k}}$ cannot be equal to 1 simultaneously. The case when both $\gamma_{k\hat{k}}$ and $\lambda_{k\hat{k}}$ are equal to zero occurs when a block is used simultaneously by two trains on different tracks, and so there is no conflict.

Another basic aspect considered in this model is the safety spacing of trains. Set of constraints (4.15, 4.16, 4.17, 4.18) specify the minimal spacing between two trains that share the same track in the same block depending on the direction they run (see Figure 4.2). For instance, if two trains share the same track in the same block and run in opposite directions, constraints (4.15) ensure a minimal time interval (Δ_j^M) between the entrance and the exit of the trains in the block. In these constraints, constant M is a very large positive number.

This model also includes restrictions about the capability of changes in the allocation of tracks/platforms. Constraints (4.19) state that it is not possible to change to another track if track t does not allow changes inside a section ($f_t = 1$). That is, if a train takes a track entering a section, it has to stay on the same track until arriving to the next station. Constraints (4.20) prohibit using a unidirectional track t for every event k defined in opposite direction. For some particular sections, a train has the possibility of changing to another track using a junction ($f_t = 0$). In that case, constraints (4.21) state that all variables q_{kt} associated to incompatible tracks are fixed to zero.

Constraints (4.22, 4.23) ensure the minimal and maximal time for connections between events k and $\hat{k}$. Constraints (4.24) guarantee that events in stations with planned stops cannot be rescheduled as unplanned stops. Additionally, the fact that the trains are allowed to stop only in stations is ensured by constraints (4.25). Finally, constraints (4.26, 4.27, 4.28, 4.29) define the domain of the decision variables.

4.2.2 Extensions

In this section we indicate how this formulation can be extended when the composition of blocks is unknown. In this special case, we need to remove constraints (4.16, 4.18); because two trains in the same direction can use the same section simultaneously. A minimal headway time is introduced, using the following constraints:

$$x_{\hat{k}}^{begin} - x_k^{begin} \geq SP_{k\hat{k}}\gamma_{k\hat{k}} - M(1-\gamma_{k\hat{k}}) \quad \forall k,\hat{k} \in L_j, j \in B : k < \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 0 \quad (4.30)$$

$$x_k^{begin} - x_{\hat{k}}^{begin} \geq SP_{\hat{k}k}\lambda_{k\hat{k}} - M(1-\lambda_{k\hat{k}}) \quad \forall k,\hat{k} \in L_j, j \in B : k < \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 0 \quad (4.31)$$

$$x_{\hat{k}}^{end} - x_k^{end} \geq SP_{k\hat{k}}\gamma_{k\hat{k}} - M(1-\gamma_{k\hat{k}}) \quad \forall k,\hat{k} \in L_j, j \in B : k < \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 0 \quad (4.32)$$

$$x_k^{end} - x_{\hat{k}}^{end} \geq SP_{\hat{k}k}\lambda_{k\hat{k}} - M(1-\lambda_{k\hat{k}}) \quad \forall k,\hat{k} \in L_j, j \in B : k < \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 0 \quad (4.33)$$

$$x_{\hat{k}}^{begin} - x_k^{end} \geq \Delta_j^F\gamma_{k\hat{k}} - M(1-\gamma_{k\hat{k}}) \quad \forall k,\hat{k} \in L_j, j \in B : k < \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 1 \quad (4.34)$$

$$x_k^{begin} - x_{\hat{k}}^{end} \geq \Delta_j^F\lambda_{k\hat{k}} - M(1-\lambda_{k\hat{k}}) \quad \forall k,\hat{k} \in L_j, j \in B : k < \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 1 \quad (4.35)$$

Two different cases must be considered: events in sections and events in stations. For the first case, constraints (4.30) and (4.31) enforce the minimal time interval, where $SP_{\hat{k}k}$ is a value depending on the characteristics of the trains and lines associated to events k and $\hat{k}$. Constraints (4.32) and (4.33) are added to ensure that a train cannot overpass a precedent train on the same track, and M is a very large positive number.

On the other hand, if the event is performed in a station, we keep the original constraints. Indeed in this special case, we consider that a station is composed exactly of one block. Thus, constraints (4.16, 4.18) become (4.34, 4.35) respectively.

4.3 Solution Methods

In this section we develop three solution methods. The first one is right-shift rescheduling. It is the fastest but its solutions are not good enough because the method is based on a simplification of the problem where order of trains, assignment of tracks, and planned stops cannot be altered. In spite of the quality of the solution, right-shift rescheduling is exploited by other methods for calculating initial solutions.

The second solution method is based on *local branching* cuts. The objective is to search solution "near" to the original schedule by exploring a neighborhood defined by these cuts. The good results obtained by this method can be explained from the fact that the original schedule contains important information that this method exploits appropriately. Finally, the method is extended to an iterative version that tries to improve the current solution by modifying the bounds of the cuts.

4.3.1 Right-shift Rescheduling

The original schedule contains important information that can be used while solving the rescheduling problem. *Right-shift rescheduling* (RS) provides a solution maintaining the original order of trains in sections/stations, keeping the original allocation of tracks/platforms, and forbidding unplanned stops. This procedure is general, in the sense that it can be applied to any rescheduling problem (Ovacik and Uzsoy, 1992). Therefore, some additional constraints are introduced in order to fix all integer variables:

$$\lambda_{k\hat{k}} = 0 \qquad \forall k, \hat{k} \in L_j, j \in B : k < \hat{k} \tag{4.36}$$

$$\gamma_{k\hat{k}} = \gamma^0_{k\hat{k}} \qquad \forall k, \hat{k} \in L_j, j \in B : k < \hat{k} \tag{4.37}$$

$$q_{kt} = q^0_{kt} \qquad \forall k \in L_j, t \in P_j, j \in B \tag{4.38}$$

$$y_k = 0 \qquad \forall k \in E \tag{4.39}$$

These additional constraints imply a drastic reduction of complexity because the remaining problem does not contain any integer variable. In addition, it is to be noted that constraints (4.14, 4.17, 4.18) could be removed of the model because $\lambda_{k\hat{k}} = 0$. As a consequence, the computing time needed for the solution of this problem falls considerably in practice. In spite of this advantage, however, it is expected that the quality of this solution will not be good. Essentially, delays will be propagated until idle time permit to absorb them.

4.3.2 MIP-based Local Search Method

This method is inspired from the *local branching* principle (Fischetti and Lodi, 2003). The solution space explored is composed of the subset of solutions "near" the original (non-disrupted) schedule. The goal of the original *local branching* approach is to improve the performance of a given MIP solver without losing the guarantee of optimality. In contrast, we use this idea as a heuristic: to find good solutions (perhaps suboptimal), in a reasonable time. Therefore, one of the branches derived from the cut is discarded in order to limit the search to a limited neighborhood.

Consider the MIP formulation presented in Section 4.2. Let Q^0 be the vector of the values $q^0_{kt} \forall k \in L_j, t \in P_j, j \in B$ (the original allocation of tracks/platforms). For two given parameters LB_q and LB_λ, neighborhood $N(Q^0, LB_q, LB_\lambda)$ can be defined as the set of feasible solutions of the problem satisfying the following local branching constraints:

$$\sum_{\substack{k \in L_j \\ t \in P_j, j \in B}} (q_{kt} - q^0_{kt})^2 \leq 2LB_q \tag{4.40}$$

$$\sum_{\substack{k,\hat{k} \in L_j : k < \hat{k} \\ j \in B}} {\lambda_{k\hat{k}}}^2 \leq LB_\lambda \tag{4.41}$$

These cuts limit both the number of changes of tracks/platforms and the number of inversion of events in time, compared to the initial planning. Note that variables $\lambda_{k\hat{k}}$ are equal to 0 when the respective order of events k and $\hat{k}$ is conserved.

Constraints (4.40) and (4.41) can be linearized easily, using the following property of binary integers: $x \in \{0,1\} \Rightarrow x^2 = x$. With this property, constraint (4.41) is simply rewritten

$$\sum_{\substack{k,\hat{k} \in L_j : k < \hat{k} \\ j \in B}} \lambda_{k\hat{k}} \leq LB_\lambda \tag{4.42}$$

The linearization of constraint (4.40) implies to define the following sets S' and S'':

$$S' = \{(j,k,t) | k \in L_j, t \in P_j, j \in B \wedge q^0_{kt} = 1\} \tag{4.43}$$

$$S'' = \{(j,k,t) | k \in L_j, t \in P_j, j \in B \wedge q^0_{kt} = 0\} \tag{4.44}$$

Seeing that constraint (4.40) is equivalently

$$\sum_{\substack{k \in L_j \\ t \in P_j, j \in B}} (q_{kt})^2 - 2q_{kt}q^0_{kt} + (q^0_{kt})^2 \leq 2LB_q \tag{4.45}$$

it can be rewritten:

$$\sum_{(j,k,t) \in S'} (1 - q_{kt}) + \sum_{(j,k,t) \in S''} q_{kt} \leq 2LB_q \tag{4.46}$$

The cardinality of the binary support concerning variables q_{kt} of any feasible solution is a constant. In fact, the number of variables q_{kt} taking the value 1 is always equal to $|E|$ (the total number of events). This is true because each event has to be assigned exactly to one track (see constraints (4.12)). As a consequence, if a variable q_{kt} change from 0 to 1, another one must change its value from 1 to 0. As a result, we have:

$$\sum_{(j,k,t)\in S'} (1 - q_{kt}) = \sum_{(j,k,t)\in S''} q_{kt} \tag{4.47}$$

Using (4.47) in (4.46), the constraint can be written in its "asymmetric" form (see (Fischetti and Lodi, 2003)):

$$2 \sum_{(j,k,t)\in S''} q_{kt} \leq 2LB_q$$

that is

$$\sum_{(j,k,t)\in S''} q_{kt} \leq LB_q \tag{4.48}$$

The *MIP-based local search method* (LS) consists in solving the model presented in Section 4.2 adding the local branching cuts (4.42) and (4.48).

We cannot ignore the problem of determining an appropriate value for LB_q and LB_λ. Clearly, there exists a tradeoff between quality of solution and computing time. On one hand, relative small values will limit the search in a small neighborhood, and probably the problem can be solved quickly. On the other hand, a large value for these parameters will increase the computing time but better solutions should be obtained.

4.3.3 Iterative MIP-based Local Search

The *Iterative MIP-based local search algorithm* (I-LS) is developed from the MIP-based local search method described above. First, right-shift rescheduling is used to find an initial feasible solution quickly. Then, an iterative procedure tries to improve the current solution using the MIP-based local search approach, with value LB_q and LB_λ iteratively adjusted.

Algorithm 1 presents the pseudocode of this method. The inputs are P_0, LB_q^0, LB_λ^0, δ^{LB_q}, δ^{LB_λ}. P_0 is the MIP formulation of an instance of the problem, using the model presented in Section 4.2. LB_q^0, LB_λ^0 are the initial values of the right-hand sides of the local branching cuts. Finally, δ^{LB_q} and δ^{LB_λ} denote the changes that are to be applied at each iteration to these values.

To avoid exploring twice the same feasible region during the different iterations, a reformulation of cuts (4.42) and (4.48) is required:

$$LB_q^1 \leq \sum_{(j,k,t)\in S''} q_{kt} \leq LB_q^2 \tag{4.49}$$

$$LB_\lambda^1 \leq \sum_{\substack{k,\hat{k}\in L_j : k<\hat{k} \\ j\in B}} \lambda_{k\hat{k}} \leq LB_\lambda^2 \tag{4.50}$$

Figure 4.3 shows how this algorithm creates a non-overlapped feasible region at every iteration (denoted by numbers 1, 2, 3, 4, 5, ...) using different values of LB_q^1, LB_q^2, LB_λ^1, and LB_λ^2.

Algorithm 1 can be precisely described as follows. The first step is to define the ending criteria. Some possibilities are the number of iterations, a number of iterations without changes in the solution or a time limit. This latter possibility was chosen for our experiments. Line 1 assigns the value *False* to a boolean variable called *EndingCriteria*.

The original schedule is used to compute an initial solution by calling function `RightShift()` (line 2). This function returns a feasible solution using the right-shift rescheduling approach presented in Section 4.3.1. Then, the objective value and the ending criteria are evaluated.

The control structure `while` (lines 10 to 30) defines one iteration of the procedure. At line 11, a subproblem P_1 is created. This subproblem corresponds to the original problem (P_0) plus constraints (4.49) and (4.50). Once the new subproblem is created, line 12 solves P_1 using Z as the cut off value.

The current subproblem could be infeasible because of the cut off value and the changes of the bounds of the local branching cuts. Note that if the subproblem is feasible but not solved optimally (which might happen when computing time is limited) or is solved optimally (line 13), the solution found is necessarily better than the one found in the preceding iteration. The current solution and the cut off value are then updated (lines 14 and 15).

Block `IF` of lines 17 to 22 is executed when the next iteration implies to increase the bounds for local branching cut (4.50), (*e.g.*, during iterations 2 and 4 in Figure 4.3). In the same way, block `IF` of lines 23 to 28 is executed when the next iteration implies to increase the bounds for local branching cut (4.49). The goal of these blocks is to enlarge the search space avoiding the exploration of solutions already evaluated. It is important to remark that this procedure never deteriorates the value of the current solution because of the cut off parameter.

Once the iteration is finished, the ending criteria is evaluated and the block 10 to 30 is repeated until function `EvaluateEndingCriteria()` returns *true*. In that case, line 31 is executed, returning the last solution found.

Algorithm 1 I-LS: Iterative MIP-based local search

Require: $P_0, LB_q, LB_\lambda, \delta^{LB_q}, \delta^{LB_\lambda}$

$EndingCriteria \Leftarrow False$
$X \Leftarrow$ RightShift(P_0)
$Z \Leftarrow$ ObjectiveValue(X)
$EndingCriteria \Leftarrow$ EvaluateEndingCriteria()
$LB_q^1 \Leftarrow 0$
$LB_\lambda^1 \Leftarrow 0$
$LB_q^2 \Leftarrow LB_q$
$LB_\lambda^2 \Leftarrow LB_\lambda$
$NextCase \Leftarrow 1$
while $(EndingCriteria \neq True)$ **do**
  $P_1 \Leftarrow P_0$ adding constraints (4.49) and (4.49)
  $X_1 \Leftarrow$ MipSolve(P_1, CutUp(Z), MaxTime)
  if (GetStatus(P_1)=*Feasible*) OR (GetStatus(P_1)=*Optimal*) **then**
    $X \Leftarrow X_1$
    $Z \Leftarrow$ ObjectiveValue(X)
  end if
  if $NextCase = 1$ **then**
    $LB_\lambda^1 \Leftarrow LB_\lambda^2$
    $LB_\lambda^2 \Leftarrow LB_\lambda^2 + \delta^{LB_\lambda}$
    $LB_q^1 \Leftarrow 0$
    $NextCase \Leftarrow 2$
  end if
  if $NextCase = 2$ **then**
    $LB_\lambda^1 \Leftarrow 0$
    $LB_q^1 \Leftarrow LB_q^2$
    $LB_q^2 \Leftarrow LB_q^2 + \delta^{LB_q}$
    $NextCase \Leftarrow 1$
  end if
  $EndingCriteria \Leftarrow$ EvaluateEndingCriteria()
end while
return X

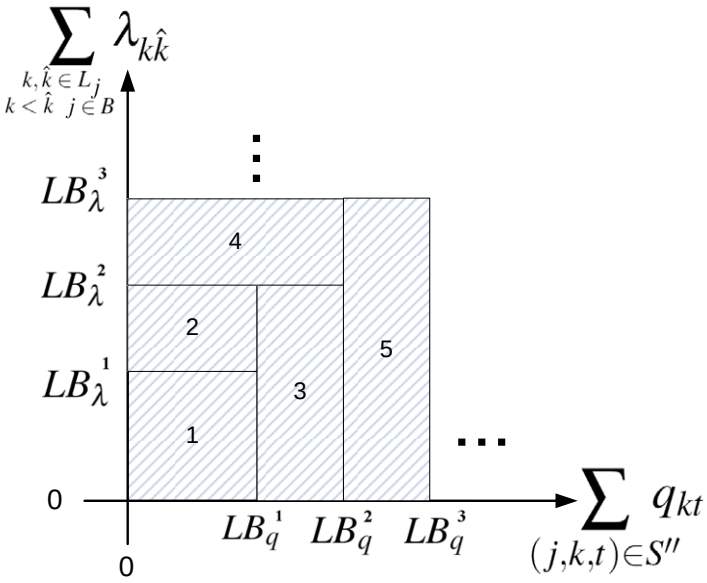


Figure 4.3: *Search space of Iterative MIP-based local search. The numbers in the areas represent the feasible search space at each iteration.*

4.4 Finding optimal solutions

The proposed approaches presented in this chapter are heuristic methods, do not prove optimality and do not give a measure of the quality of the solution. In this section we propose a simple exact algorithm for finding optimal solutions with the objective to evaluate and compare the solution methods.

Once we built the MIP model and had problem instances, we tried to solve it directly in a standard MIP solver. As we expected, this method was not efficient. For example, the solver needed many hours to find a first feasible solution for many of the instances. This fact can be explained by analyzing the nature of the model. Note that the model is composed of binary variables and constraints with large M constants. These both facts imply that the solution of the linear programming relaxation is normally far away from the optimal solution. As a consequence, a branch and bound method may take too much time before to find a feasible solution and so to prove optimality.

We propose an alternative procedure to calculate optimal solutions. The idea is to use the output solution of the proposed methods (*i.e.*, right-shift rescheduling, the MIP-based local search and its iterative version) as an initial solution for a MIP solver. Two important advantage must be noted: the possibility of finding optimal solutions, and the possibility of measuring the quality of any feasible solution. Concerning the second advantage, an exact method based on branch and bound permits to calculate a MIP gap expressed as a percentage of the incumbent solution. It is important to remark that the facts discussed in this section were applied in IBM ILOG CPLEX 11, but it could be perfectly valid for other state-of-the-art software packages.

Algorithm 2 is a generic procedure that employs a standard MIP solver with the functionality of accepting feasible solutions as start parameters. The input of this algorithm is P_0. Let P_0 be the MIP formulation of an instance of the problem, using the model presented in Section 4.2. The first step is to calculate a good solution of the problem instance by using one of the methods proposed in this chapter (line 1). The application of this method could imply that the MIP model change because of the local branching cuts. This is the reason why Line 2 removes all additional cuts in the MIP. As a consequence, the feasible region associated to P_0 corresponds to the search space of the original problem and so an optimal solution for P_0 is also an optimal solution for the original problem.

In Line 3, this solution is then employed as a MIP start data in a MIP solver that allows starting values. The MIP solver processes this current solution before starting branch & cut. After checking feasibility, the solver installs the solution as the **incumbent solution**. Having an incumbent from at the beginning of branch & cut permits to eliminate portions of the search space and, in practice, it result in smaller search trees. The existence of an incumbent solution also allows the solver to use other integrated heuristics, such as relaxation induced neighborhood search (RINS heuristic) or solution polishing (both valid in the last versions of IBM ILOG CPLEX) to improve the current solution.

Assuming that a given problem instance is feasible, the natural output of the MIP solver is an optimal solution. The last line of the algorithm returns this solution to the user. Nevertheless, even if the heuristic gives a very good solution, this procedure could take too much time in practice. It is then possible to stop the search before obtaining a prove of optimality by using an alternative ending criterion such as a time limit. In that case, it should be interesting to know both: the best integer (feasible) solution found so far, and the quality of this solution expressed for example as a MIP gap.

Algorithm 2 Exact algorithm based on a heuristic method

Require: P_0
1: $X^0 \Leftarrow$ HeuristicSolver(P_0)
2: $P_0 \Leftarrow$ RemoveCut (P_0)
3: $X \Leftarrow$ MIPSolver(P_0, X^0)
4: **return** X

4.5 Computational Experiments

The goals of the computational experiments are to evaluate performance of the suggested methods and to study the impact of different objectives on the total delay. To achieve these objectives, we solved a heterogeneous set of instances using different combination of the coefficients in the objective function.

The algorithms were coded in Visual Studio 2005[2] (C♯) using IBM ILOG CPLEX

[2]http://www.microsoft.com/visualstudio

version 11.1 as a black-box MIP solver [3]. Our system runs on a single processor IBM compatible PC Intel Core 2, 1.66 GHz, and 2 GB of main memory.

The code was integrated to Railway Rescheduling Tool, software developed for this Thesis. You will find a description of the software in Chapter 10.

We compare the following methods:

- Method O1, RS: *Right-shift rescheduling* (see Section 4.3.1).
- Method O2, LS: *MIP-based local search method* using RS as initial solution. This algorithm performs the method described in Section 4.3.2.
- Method O3, LS + CPLEX: *MIP-based local search method + CPLEX*. This method uses the output of Method O2 as an initial solution of ILOG CPLEX removing any additional cut. Because all cuts are removed, the feasible region corresponds to the original search space. As a result, the optimal solution of this method coincides with the optimal solution of the original problem. The goal of this method is to obtain an optimal solution to be used as a reference to evaluate the quality of the solutions found with other methods. (see Section 4.4).
- Method O4, I-LS (5 minutes): *Iterative MIP-based local search* algorithm limited to 5 minutes of running time (see Section 4.3.3). We limit the processing time in order to respond to real requirements of railway companies. The algorithm performs only one iteration.

For the tests we used $LB_q = 10$, $LB_\lambda = 100$ and $\delta^{LB_q} = \delta^{LB_\lambda} = 10$. The values LB_q and LB_λ have been chosen following recommendations of Local Branching cuts authors in (Fischetti and Lodi, 2003) and our own numerical experiments.

To evaluate the impact of different objectives on the total delay, three scenarios of instances, each one emphasizing one aspect of the objective function, have been considered. The scenarios are:

- Emphasize minimization of delay: The coefficients in the objective function strongly penalize the delays. The coefficients are $CD = 1000$, $CFD = 1000$, $CCT = 1$, $CCP = 1$, and $CUS = 1$. Every second of delay cost 1000 times more than a simple change of tracks/platforms/stops.
- Emphasize stability: we consider that the number of changes of track/platform/stops is inversely proportional to the stability of the schedule. Thus, the coefficients in the objective function strongly penalize the changes of tracks/platforms/stops. The coefficients are $CD = 1$, $CFD = 1$, $CCT = 1000$, $CCP = 1000$, and $CUS = 1000$.
- Balance stability and minimizing delay: an equilibrated combination of coefficients in the objective function is considered. The coefficients are $CD = 1$, $CFD = 1$, $CCT = 30$, $CCP = 30$, and $CUS = 15$. These values were selected to permit some changes of track/platform/stops if delays become too large.

[3] http://www.ilog.com

Two different networks have been used for the computational experiments. The first one is a line located in France, and the second one a network, with a tree topology, located in Chile.

Network 1. Monts-Ruffec, France

This network corresponds to a real French line connecting two cities: Monts and Ruffec (Figure 4.4). For the experiments a time horizon of 7 hours is considered. In that period of time, the original schedule is composed of 67 trains passing through 43 stations, and corresponding to 3424 events to be rescheduled. This railway line is used by different kind of trains: freight, regional express, and high speed trains.

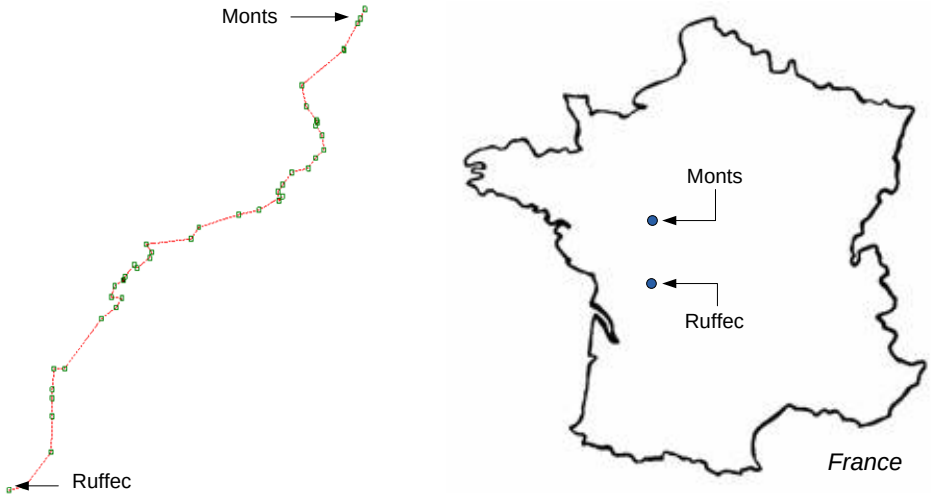


Figure 4.4: *Network 1. Left: Stations and sections of the line. Right: Location of the line in France.*

An original schedule is known (i.e., allocation of tracks/platforms, arrival/departure times for every train in each station/section of the network). This data was obtained from the research department of SNCF (French National Railway Company) in the context of the MAGES project (Acuna-Agost and Gueye, 2006). The composition of blocks in sections is unknown and the extended version of the formulation is used (see Section 4.2.2).

The original schedule is described in Figure 4.5 in a *time-distance diagram*, usually called *traffic diagram* or *string graph* (in North American railways). Each line of this diagram represents a train circulation over time and space.

For the experiments, we disturbed trains on extreme nodes (Monts and Ruffec) in order to affect a large number of trains. Figure 4.6 shows a complete description of the MIP instances constructed using this network. Note that 6 different instances are defined (*france_1* to *france_6*), with different characteristics for the incident. Also, these 6 instances are successively assessed with the three objective functions described above.

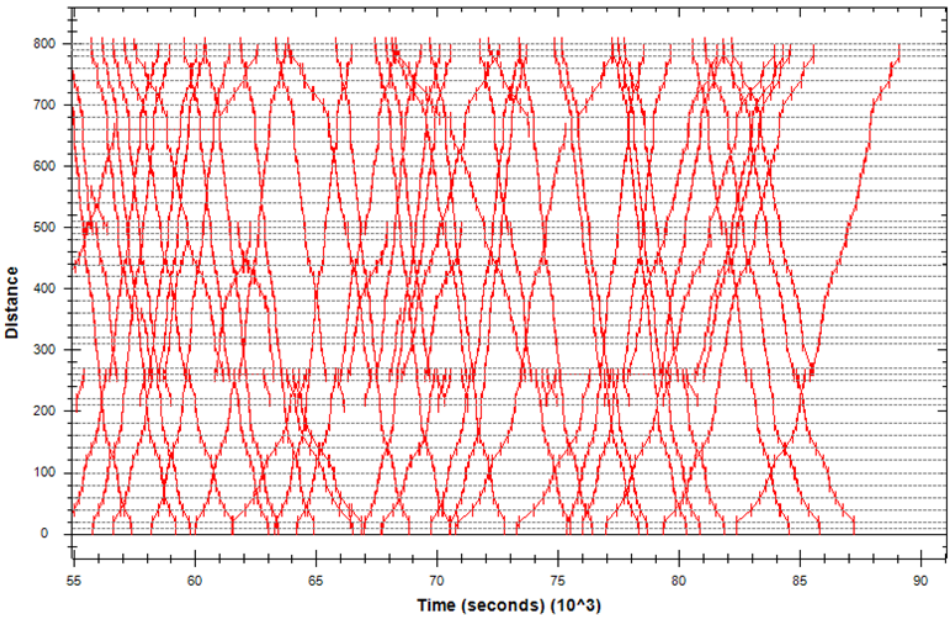


Figure 4.5: *Traffic diagram for Network 1 (France). Every line represents a train, x-axis corresponds to the time in seconds, and y-axis represents a relative distance (without real units).*

One or two trains are directly implicated in the incident. Three delay values have been considered for these trains: 10, 20, and 30 minutes long. We do not take into account shorter delays because of the recovery times introduced to absorb small delays. Indeed, a recovery time is normally added to the running time of trains when constructing the initial schedule, which is defined as a percentage of the normal running time. The typical additional amount of time is 3 to 7% on European railways (Pachl, 2004). It enables a train to recover small delays and prevents against small delay propagations on other trains. On the other hand, longer delays are out of the scope of this Thesis since, they are limited to average delays corresponding to incidents described in Figure 3.1.

	Instance		Incident Delay [min]	Total Perturbed Trains	Time Horizon [hr]	Unitary costs				
	N°	Name				CD	CFD	CCT	CCP	CUS
a) Balance stability and minimizing delay	1	france_1_a_7	10	1	7	1	1	30	15	10
	2	france_2_a_7	20	1	7	1	1	30	15	10
	3	france_3_a_7	30	1	7	1	1	30	15	10
	4	france_4_a_7	10	2	7	1	1	30	15	10
	5	france_5_a_7	20	2	7	1	1	30	15	10
	6	france_6_a_7	30	2	7	1	1	30	15	10
b) Emphasize stability	7	france_1_b_7	10	1	7	1	1	P	P	P
	8	france_2_b_7	20	1	7	1	1	P	P	P
	9	france_3_b_7	30	1	7	1	1	P	P	P
	10	france_4_b_7	10	2	7	1	1	P	P	P
	11	france_5_b_7	20	2	7	1	1	P	P	P
	12	france_6_b_7	30	2	7	1	1	P	P	P
c) Emphasize minimizing delay	13	france_1_c_7	10	1	7	P	P	1	1	1
	14	france_2_c_7	20	1	7	P	P	1	1	1
	15	france_3_c_7	30	1	7	P	P	1	1	1
	16	france_4_c_7	10	2	7	P	P	1	1	1
	17	france_5_c_7	20	2	7	P	P	1	1	1
	18	france_6_c_7	30	2	7	P	P	1	1	1

Figure 4.6: *Set of instances for Network 1 (France).*

Network 2. South of Chile

This network corresponds to a real Chilean network composed of 140 trains, 49 stations, and 1378 events in a horizon of 24 hours (see Figure 4.7). The network is used by different kind of trains: freight, short distance passenger trains, and long distance passenger trains.

The original schedule was taken from public information given by EFE (Chilean State Railways)[4] and FEPASA (the largest freight railroad company in Chile)[5]. For the experiments, we considered that all sections are double-tracked and all stations have four platforms. The reason for this simplification is that the information on the detailed network layout is very difficult to find. The composition of blocks in sections is also unknown and the extended version of the formulation is again used (see Section 4.2.2).

A significant difference with the French network has to be noticed. In Network 2, there is no train circulation covering the whole network. That is the reason why perturbations have been introduced on four cities: Santiago, Chillán, Concepción, and Puerto Montt.

[4]http://www.efe.cl
[5]http://www.fepasa.cl

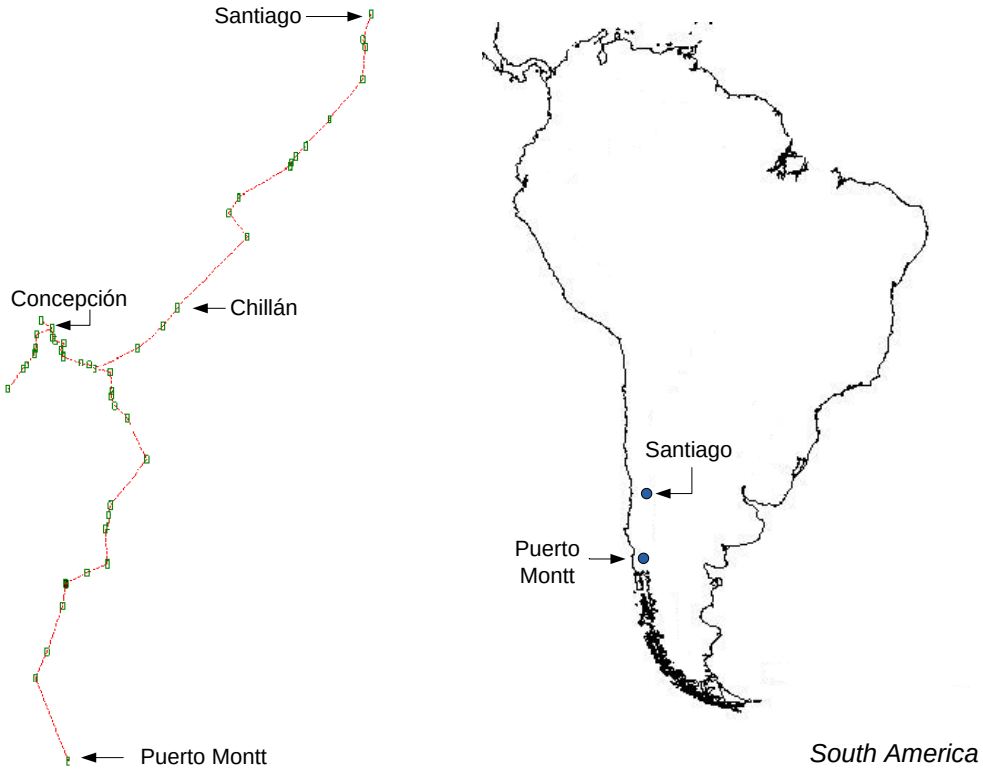


Figure 4.7: *Network 2. Left: Stations and sections of the network. Right: Location of the network in Chile.*

Santiago is the biggest city of the country and a large number of trains depart from its station. Chillán is the final destination for long distance trains going to and coming from Santiago. Concepción has been selected since it is the main city of the south of Chile with the most important train traffic after Santiago. Finally, Puerto Montt is located at the south extremity of our network.

To compare traffics in Network 1 and 2 we have also reported in Figure 4.8 an extract of the traffic diagram for the Chilean network.

For the experiments, delays have been considered on one to four train(s) departing from the selected cities. Delays are again either be set to 10, 20, or 30 minutes. Figure 4.9 shows a complete list of MIP instances for this network. Note that 12 different instances are defined (*chile_1* to *chile_12*).

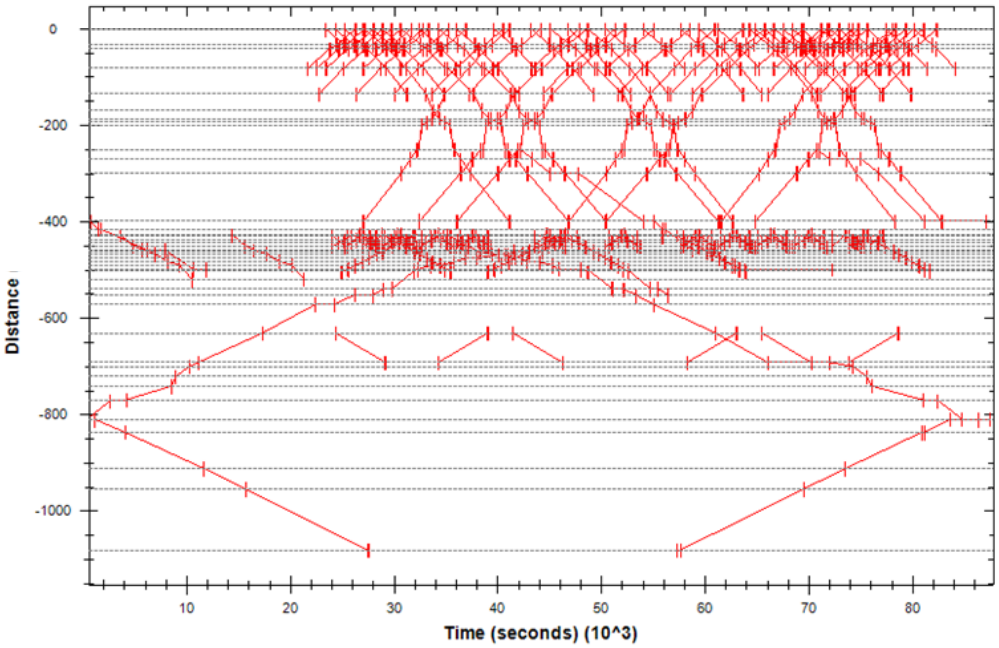


Figure 4.8: *Traffic diagram for Network 2 (Chile). Every line represents a train, x-axis corresponds to the time in seconds, and y-axis represents a relative distance (without real units).*

Results

Performance of the Suggested Methods

In Figures 4.11 and 4.10, comparisons of the different methods for each instance are reported. Column (*Cost [$]*) is for the value of the objective function. Column (*CPU [sec]*) is the total execution time of the methods in seconds. Column *GAP [%]* is the relative deviation between the optimal solution ($y*$) and the heuristic solution (y) for the corresponding method (Method O1, O2 or O4). Let us recall that the optimal solution of our rescheduling problem is found by Method O3. This gap is computed as follows: $GAP_y = (y - y*)/y*$.

This analysis is complemented by Figures 4.12 and 4.13. They show the CPU time required for computing the best solution has been found. As it is possible to observe, in some cases an important amount of time is actually used to prove optimality. In both Networks, the CPU time to the best solution coincide with the CPU time that the algorithm stops for Method O1. In average, Method O2 only needs 37% (85%) of the total time to find the best solution for Network 1 (Network 2). In the same way, we obtain 19% (27%) for O3 and 57% for O4 (85%) for Network 1 (Network 2). It should be noted that Method O4 obtains the best compromise quality/speed if we consider the CPU time to the best solution.

The results show that instances of Network 1 are harder to solve than instances

of Network 2. Such results may be explained by a greater number of events to be rescheduled in Network 1, a higher traffic density, and the correlation between these events. Indeed, Network 1 has 67 trains scheduled in 7 hours while Network 2 has 140 trains scheduled in 24 hours. Moreover, most of the trains of Network 1 go across the entire line and then affect a large number of other trains. In contrast, trains of Network 2 have shorter itineraries and a more local impact (see Figures 4.5 and 4.8).

The first method, RS, is the fastest. Recall that RS provides a solution maintaining the original order of trains in sections/stations and keeping the original allocation of tracks/platforms. In average, RS is able to compute a feasible solution in 3 seconds for instances of Network 1 and 1 second for Network 2. The average gap obtained by RS for all instances in Network 1 and Network 2 are respectively : 141.93% and 17.07% (thus showing that instances of Network 2 are easier to solve for RS). In fact, since incidents on trains of Network 2 have a local impact, keeping the original order and avoiding changes is sufficient for many of them. On the contrary, in Network 1, such approach performs very badly since a greater number of trains are affected simultaneously by an incident.

Method O2 (LS: Heuristic, see Section 4.2) finds the optimal (without proof) for all instances. Its average total running times are 400 seconds for Network 1 and 21 seconds for Network 2.

Method O3 (LS + CPLEX, see Section 4.2) is the exact solution scheme. The procedure is approximately two times slower than Method O2 for Network 1 and three times slower for Network 2.

Finally, the average gap of Method O4 (I-LS (5 minutes), see Section 4.2) is 0.04% for Network 1 and 0.00% for Network 2, with average CPU times of 185 seconds for Network 1 and 21 seconds for Network 2.

In conclusion, RS is the fastest method and can very conveniently be used to generate quickly a solution. However, depending of the network, this solution can really be very bad. Conversely, the MIP-based local search procedure always obtain optimal solutions (on our cases) but with computing times that might sometimes be considered as unacceptable. The best compromise is obtained with the Iterative MIP-based local search procedure that is able to obtain very good solutions (average gap less than 1%) within 5 minutes of computing time and thus proves to be viable in practice.

Impact of Different Objectives on the Total Delay

Figure 4.14 highlights the impact of the different objectives on the total delay. This table presents the optimal results depending on the type of objectives the emphasis is put on.

Column *Optimal Solution - Cost [$]* gives the value of the optimal solution calculated by Method 3. Column *Total Delay [sec]* is the total delay (in seconds) of the optimal solution. In contrast to the solution cost, the total delay can be used to compare two solutions of the same instance but different objective function. Column *Delay GAP [%]* is the relative difference between the optimal total delay, calculated using *Emphasize*

minimizing delay, and the total delay of the current solution. Last column *Average Delay GAP [%]* gives the average of gap per type of objective.

The conclusion that can be drawn from this table is that the total delay is not significantly affected by changes in the coefficients of the objective function. In average strategy *Emphasize stability* only deteriorates by 2.06% and 3.01% the total delay for Network 1 and Network 2 respectively. Practically, it shows that quite stable schedules (in terms of tracks and platforms) can be obtained without deteriorating dramatically the quality of service (delay) for the customers.

Right-shift rescheduling was the first solution method presented in this document. This procedure fixes some integer variables keeping the order of trains and the allocation of tracks/platforms. By fixing integer variables, the problem becomes easy to solve by a MIP solver, but the solutions are often of poor quality.

The second approach is a MIP-based local search method. The difference with traditional local search mechanisms is that neighborhoods are obtained through linear inequalities, called local-branching cuts, added to the MIP model. In addition, information about the original schedule is used to generate these cuts.

4.6 Conclusions

A MIP formulation was presented for this problem. This model includes many practical rules and constraints, which explains its relative complexity compared to other models presented in the literature. Thus, it supports allocation of tracks (and platforms) connection between trains, bidirectional/multi-track lines and extra time for accelerating and braking. As a consequence, the difficulty of resolution increases significantly. In fact, with a relative large number of trains and stations, a standard MIP solver was not able to solve the test instances in a reasonable time.

The problem is addressed with a local search method based on this formulation. The method uses the original (non-disrupted) schedule to compute the new one. This is particularly useful for rescheduling problems, where one expects to limit the impact of the disruption to the schedule, that is, the difference between the schedule computed after disruptions and the initial one.

Right-shift rescheduling was the first solution method presented in this chapter. This procedure fixes some integer variables keeping the order of trains and the allocation of tracks/platforms. By fixing integer variables, the problem becomes easier to solve by a MIP solver, but the solutions are generally far away from the optimum.

The second approach is a MIP-based local search method. The big difference with traditional local search mechanisms is that neighborhoods are obtained through the introduction, in the MIP model, of linear inequalities called local-branching cuts. In addition, information about the original schedule is used to construct these cuts.

An iterative variant of this approach, the iterative MIP-based local search method,

proved to be viable in practice, obtaining near optimal solutions with a computing time compatible with the context of rescheduling.

Finally, it is important to remark that it was not possible to compare these methods with other ones because of the lack of public benchmarks. Nevertheless, the computation experiments show that these methods are more efficient than a standard MIP solver.

Another important limitation of MIP formulations for scheduling (and rescheduling) problems is the exponential number of variables and constraints used to model the order of tasks (trains). As a consequence, it is not possible to construct MIP models for large instances, *i.e.*, the same network with a longer time horizon, because of memory limits. For that reason an alternative formulation is studied in Chapter 6.

	Instance		Incident Delay [min]	Total Perturbed Trains	Time Horizon [hr]	Unitary costs				
	N°	Name				CD	CFD	CCT	CCP	CUS
a) Balance stability and minimizing delay	19	chile_1_a	10	1	24	1	1	30	15	10
	20	chile_2_a	20	1	24	1	1	30	15	10
	21	chile_3_a	30	1	24	1	1	30	15	10
	22	chile_4_a	10	2	24	1	1	30	15	10
	23	chile_5_a	20	2	24	1	1	30	15	10
	24	chile_6_a	30	2	24	1	1	30	15	10
	25	chile_7_a	10	3	24	1	1	30	15	10
	26	chile_8_a	20	3	24	1	1	30	15	10
	27	chile_9_a	30	3	24	1	1	30	15	10
	28	chile_10_a	10	4	24	1	1	30	15	10
	29	chile_11_a	20	4	24	1	1	30	15	10
	30	chile_12_a	30	4	24	1	1	30	15	10
b) Emphasize stability	31	chile_1_b	10	1	24	1	1	P	P	P
	32	chile_2_b	20	1	24	1	1	P	P	P
	33	chile_3_b	30	1	24	1	1	P	P	P
	34	chile_4_b	10	2	24	1	1	P	P	P
	35	chile_5_b	20	2	24	1	1	P	P	P
	36	chile_6_b	30	2	24	1	1	P	P	P
	37	chile_7_b	10	3	24	1	1	P	P	P
	38	chile_8_b	20	3	24	1	1	P	P	P
	39	chile_9_b	30	3	24	1	1	P	P	P
	40	chile_10_b	10	4	24	1	1	P	P	P
	41	chile_11_b	20	4	24	1	1	P	P	P
	42	chile_12_b	30	4	24	1	1	P	P	P
c) Emphasize minimizing delay	43	chile_1_c	10	1	24	P	P	1	1	1
	44	chile_2_c	20	1	24	P	P	1	1	1
	45	chile_3_c	30	1	24	P	P	1	1	1
	46	chile_4_c	10	2	24	P	P	1	1	1
	47	chile_5_c	20	2	24	P	P	1	1	1
	48	chile_6_c	30	2	24	P	P	1	1	1
	49	chile_7_c	10	3	24	P	P	1	1	1
	50	chile_8_c	20	3	24	P	P	1	1	1
	51	chile_9_c	30	3	24	P	P	1	1	1
	52	chile_10_c	10	4	24	P	P	1	1	1
	53	chile_11_c	20	4	24	P	P	1	1	1
	54	chile_12_c	30	4	24	P	P	1	1	1

P=1000

Figure 4.9: *Set of instances for Network 2 (Chile).*

	Instance		Method O1: Right-shift Heuristic			Method O2: LS Heuristic			Method O3: LS + CPLEX Exact			Method O4: I-LS (5 minutes) : Heuristic		
	N°	Name	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]
a) Balance stability and minimizing delay	1	france_1_a_7	6 103	64.32%	3	3 714	0.00%	88	3 714	0.00%	366	3 714	0.00%	88
	2	france_2_a_7	22 405	196.91%	4	7 546	0.00%	64	7 546	0.00%	231	7 546	0.00%	64
	3	france_3_a_7	48 469	272.09%	3	13 026	0.00%	255	13 026	0.00%	714	13 026	0.00%	255
	4	france_4_a_7	6 819	9.45%	3	6 230	0.00%	89	6 230	0.00%	322	6 230	0.00%	89
	5	france_5_a_7	32 692	122.35%	3	14 703	0.00%	335	14 703	0.00%	710	14 703	0.00%	300
	6	france_6_a_7	68 329	175.21%	3	24 828	0.00%	2 339	24 828	0.00%	3 348	24 903	0.30%	300
b) Emphasize stability	7	france_1_b_7	6 103	52.38%	3	4 005	0.00%	91	4 005	0.00%	501	4 005	0.00%	91
	8	france_2_b_7	22 405	196.91%	3	7 546	0.00%	67	7 546	0.00%	210	7 546	0.00%	67
	9	france_3_b_7	48 469	270.61%	3	13 078	0.00%	445	13 078	0.00%	829	13 078	0.00%	300
	10	france_4_b_7	8 619	32.17%	3	6 521	0.00%	101	6 521	0.00%	488	6 521	0.00%	101
	11	france_5_b_7	32 692	122.35%	3	14 703	0.00%	486	14 703	0.00%	820	14 703	0.00%	300
	12	france_6_b_7	68 329	164.18%	4	25 865	0.00%	291	25 865	0.00%	824	25 865	0.00%	291
c) Emphasize minimizing delay	13	france_1_c_7	6 103 000	67.02%	3	3 654 002	0.00%	78	3 654 002	0.00%	366	3 654 002	0.00%	78
	14	france_2_c_7	22 405 000	196.91%	3	7 546 000	0.00%	68	7 546 000	0.00%	248	7 546 000	0.00%	68
	15	france_3_c_7	48 469 000	273.82%	3	12 966 004	0.00%	267	12 966 004	0.00%	828	12 966 004	0.00%	267
	16	france_4_c_7	8 619 000	39.69%	3	6 170 002	0.00%	84	6 170 002	0.00%	365	6 170 002	0.00%	84
	17	france_5_c_7	32 692 000	122.35%	3	14 703 000	0.00%	287	14 703 000	0.00%	682	14 703 000	0.00%	287
	18	france_6_c_7	68 328 900	176.07%	4	24 751 006	0.00%	1 768	24 751 006	0.00%	2 123	24 835 010	0.34%	300
		Average:		**141.93%**	**3**		**0.00%**	**400**		**0.00%**	**776**		**0.04%**	**185**

GAP_Y = (Y -Y)/Y* Y* : Optimal Solution*

Figure 4.10: *Results for instances of Network 1 (France).*

	Instance		Method O1: Right-shift Heuristic			Method O2: LS Heuristic			Method O3: LS + CPLEX Exact			Method O4: I-LS (5 minutes) : Heuristic		
	N°	Name	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]
a) Balance stability and minimizing delay	19	chile_1_a	8 340	2.77%	1	8 115	0.00%	41	8 115	0.00%	51	8 115	0.00%	41
	20	chile_2_a	18 300	15.75%	1	15 810	0.00%	14	15 810	0.00%	65	15 810	0.00%	14
	21	chile_3_a	31 380	24.67%	1	25 170	0.00%	36	25 170	0.00%	85	25 170	0.00%	36
	22	chile_4_a	8 580	4.00%	1	8 250	0.00%	13	8 250	0.00%	58	8 250	0.00%	13
	23	chile_5_a	22 680	20.19%	1	18 870	0.00%	14	18 870	0.00%	74	18 870	0.00%	14
	24	chile_6_a	47 580	52.35%	1	31 230	0.00%	41	31 230	0.00%	101	31 230	0.00%	41
	25	chile_7_a	10 800	3.15%	1	10 470	0.00%	12	10 470	0.00%	59	10 470	0.00%	12
	26	chile_8_a	30 800	14.12%	1	26 990	0.00%	14	26 990	0.00%	73	26 990	0.00%	14
	27	chile_9_a	63 500	34.68%	1	47 150	0.00%	41	47 150	0.00%	106	47 150	0.00%	41
	28	chile_10_a	12 600	2.69%	1	12 270	0.00%	12	12 270	0.00%	63	12 270	0.00%	12
	29	chile_11_a	34 400	12.46%	1	30 590	0.00%	15	30 590	0.00%	70	30 590	0.00%	15
	30	chile_12_a	68 900	31.11%	1	52 550	0.00%	40	52 550	0.00%	93	52 550	0.00%	40
b) Emphasize stability	31	chile_1_b	8 340	2.21%	1	8 160	0.00%	11	8 160	0.00%	51	8 160	0.00%	11
	32	chile_2_b	18 300	9.32%	1	16 740	0.00%	13	16 740	0.00%	67	16 740	0.00%	13
	33	chile_3_b	31 380	19.13%	1	26 340	0.00%	21	26 340	0.00%	75	26 340	0.00%	21
	34	chile_4_b	8 580	2.14%	1	8 400	0.00%	12	8 400	0.00%	50	8 400	0.00%	12
	35	chile_5_b	22 680	14.55%	1	19 800	0.00%	13	19 800	0.00%	65	19 800	0.00%	13
	36	chile_6_b	47 580	46.85%	1	32 400	0.00%	18	32 400	0.00%	64	32 400	0.00%	18
	37	chile_7_b	10 800	1.69%	1	10 620	0.00%	13	10 620	0.00%	51	10 620	0.00%	13
	38	chile_8_b	30 800	10.32%	1	27 920	0.00%	13	27 920	0.00%	52	27 920	0.00%	13
	39	chile_9_b	63 500	31.42%	1	48 320	0.00%	20	48 320	0.00%	73	48 320	0.00%	20
	40	chile_10_b	12 600	1.45%	1	12 420	0.00%	13	12 420	0.00%	53	12 420	0.00%	13
	41	chile_11_b	34 400	9.14%	1	31 520	0.00%	14	31 520	0.00%	53	31 520	0.00%	14
	42	chile_12_b	68 900	28.26%	1	53 720	0.00%	20	53 720	0.00%	72	53 720	0.00%	20
c) Emphasize minimizing delay	43	chile_1_c	8 340 000	2.96%	1	8 100 001	0.00%	11	8 100 001	0.00%	50	8 100 001	0.00%	11
	44	chile_2_c	18 300 000	15.97%	1	15 780 002	0.00%	14	15 780 002	0.00%	64	15 780 002	0.00%	14
	45	chile_3_c	31 380 000	24.82%	1	25 140 002	0.00%	39	25 140 002	0.00%	92	25 140 002	0.00%	39
	46	chile_4_c	8 580 000	4.38%	1	8 220 002	0.00%	12	8 220 002	0.00%	53	8 220 002	0.00%	12
	47	chile_5_c	22 680 000	20.38%	1	18 840 002	0.00%	15	18 840 002	0.00%	69	18 840 002	0.00%	15
	48	chile_6_c	47 580 000	52.50%	1	31 200 002	0.00%	45	31 200 002	0.00%	107	31 200 002	0.00%	45
	49	chile_7_c	10 800 000	3.45%	1	10 440 002	0.00%	13	10 440 002	0.00%	56	10 440 002	0.00%	13
	50	chile_8_c	30 800 000	14.24%	1	26 960 002	0.00%	17	26 960 002	0.00%	88	26 960 002	0.00%	17
	51	chile_9_c	63 500 000	34.76%	1	47 120 003	0.00%	43	47 120 003	0.00%	93	47 120 003	0.00%	43
	52	chile_10_c	12 600 000	2.94%	1	12 240 002	0.00%	13	12 240 002	0.00%	55	12 240 002	0.00%	13
	53	chile_11_c	34 400 000	12.57%	1	30 560 002	0.00%	16	30 560 002	0.00%	69	30 560 002	0.00%	16
	54	chile_12_c	68 900 000	31.19%	1	52 520 003	0.00%	42	52 520 003	0.00%	112	52 520 003	0.00%	42
	Average:			**17.07%**	**1**		**0.00%**	**21**		**0.00%**	**70**		**0.00%**	**21**

GAP_Y = (Y -Y)/Y* Y* : Optimal Solution*

Figure 4.11: *Results for instances of Network 2 (Chile).*

	Instance		Method O1: Right-shift Heuristic			Method O2: LS Heuristic			Method O3: LS + CPLEX Exact			Method O4: I-LS Heuristic		
	N°	Name	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]
a) Balance stability and minimizing delay	1	france_1_a_7	6 103	64.32%	3	3 714	0.00%	78	3 714	0.00%	78	3 714	0.00%	78
	2	france_2_a_7	22 405	196.91%	4	7 546	0.00%	53	7 546	0.00%	53	7 546	0.00%	53
	3	france_3_a_7	48 469	272.09%	3	13 026	0.00%	245	13 026	0.00%	245	13 026	0.00%	245
	4	france_4_a_7	6 819	9.45%	3	6 230	0.00%	81	6 230	0.00%	81	6 230	0.00%	81
	5	france_5_a_7	32 692	122.35%	3	14 703	0.00%	70	14 703	0.00%	70	14 703	0.00%	70
	6	france_6_a_7	68 329	175.21%	3	24 828	0.00%	622	24 828	0.00%	622	24 903	0.30%	256
b) Emphasize stability	7	france_1_b_7	6 103	52.38%	3	4 005	0.00%	55	4 005	0.00%	55	4 005	0.00%	55
	8	france_2_b_7	22 405	196.91%	3	7 546	0.00%	52	7 546	0.00%	52	7 546	0.00%	52
	9	france_3_b_7	48 469	270.61%	3	13 078	0.00%	81	13 078	0.00%	81	13 078	0.00%	81
	10	france_4_b_7	8 619	32.17%	3	6 521	0.00%	58	6 521	0.00%	58	6 521	0.00%	58
	11	france_5_b_7	32 692	122.35%	3	14 703	0.00%	57	14 703	0.00%	57	14 703	0.00%	57
	12	france_6_b_7	68 329	164.18%	4	25 865	0.00%	186	25 865	0.00%	186	25 865	0.00%	186
c) Emphasize minimizing delay	13	france_1_c_7	6 103 000	67.02%	3	3 654 002	0.00%	55	3 654 002	0.00%	55	3 654 002	0.00%	55
	14	france_2_c_7	22 405 000	196.91%	3	7 546 000	0.00%	54	7 546 000	0.00%	54	7 546 000	0.00%	54
	15	france_3_c_7	48 469 000	273.82%	3	12 966 004	0.00%	107	12 966 004	0.00%	107	12 966 004	0.00%	107
	16	france_4_c_7	8 619 000	39.69%	3	6 170 002	0.00%	85	6 170 002	0.00%	85	6 170 002	0.00%	85
	17	france_5_c_7	32 692 000	122.35%	3	14 703 000	0.00%	67	14 703 000	0.00%	67	14 703 000	0.00%	67
	18	france_6_c_7	68 328 900	176.07%	4	24 751 006	0.00%	690	24 751 006	0.00%	690	24 835 010	0.34%	261
		Average:		**141.93%**	**3**		**0.00%**	**150**		**0.00%**	**150**		**0.04%**	**105**

GAP_Y = (Y -Y)/Y* Y* : Optimal Solution*

Figure 4.12: *CPU time required for computing the best solution. Network 1 (France).*

	Instance		Method O1: Right-shift Heuristic			Method O2: LS Heuristic			Method O3: LS + CPLEX Exact			Method O4: I-LS Heuristic		
	N°	Name	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]
a) Balance stability and minimizing delay	19	chile_1_a	8 340	2.77%	1	8 115	0.00%	33.62	8 115	0.00%	33.62	8 115	0.00%	33.62
	20	chile_2_a	18 300	15.75%	1	15 810	0.00%	13.72	15 810	0.00%	13.72	15 810	0.00%	13.72
	21	chile_3_a	31 380	24.67%	1	25 170	0.00%	35.64	25 170	0.00%	35.64	25 170	0.00%	35.64
	22	chile_4_a	8 580	4.00%	1	8 250	0.00%	11.44	8 250	0.00%	11.44	8 250	0.00%	11.44
	23	chile_5_a	22 680	20.19%	1	18 870	0.00%	12.60	18 870	0.00%	12.60	18 870	0.00%	12.60
	24	chile_6_a	47 580	52.35%	1	31 230	0.00%	41.00	31 230	0.00%	41.00	31 230	0.00%	41.00
	25	chile_7_a	10 800	3.15%	1	10 470	0.00%	9.84	10 470	0.00%	9.84	10 470	0.00%	9.84
	26	chile_8_a	30 800	14.12%	1	26 990	0.00%	13.86	26 990	0.00%	13.86	26 990	0.00%	13.86
	27	chile_9_a	63 500	34.68%	1	47 150	0.00%	35.67	47 150	0.00%	35.67	47 150	0.00%	35.67
	28	chile_10_a	12 600	2.69%	1	12 270	0.00%	10.08	12 270	0.00%	10.08	12 270	0.00%	10.08
	29	chile_11_a	34 400	12.46%	1	30 590	0.00%	14.85	30 590	0.00%	14.85	30 590	0.00%	14.85
	30	chile_12_a	68 900	31.11%	1	52 550	0.00%	35.60	52 550	0.00%	35.60	52 550	0.00%	35.60
b) Emphasize stability	31	chile_1_b	8 340	2.21%	1	8 160	0.00%	9.79	8 160	0.00%	9.79	8 160	0.00%	9.79
	32	chile_2_b	18 300	9.32%	1	16 740	0.00%	12.87	16 740	0.00%	12.87	16 740	0.00%	12.87
	33	chile_3_b	31 380	19.13%	1	26 340	0.00%	21.00	26 340	0.00%	21.00	26 340	0.00%	21.00
	34	chile_4_b	8 580	2.14%	1	8 400	0.00%	12.00	8 400	0.00%	12.00	8 400	0.00%	12.00
	35	chile_5_b	22 680	14.55%	1	19 800	0.00%	11.18	19 800	0.00%	11.18	19 800	0.00%	11.18
	36	chile_6_b	47 580	46.85%	1	32 400	0.00%	16.56	32 400	0.00%	16.56	32 400	0.00%	16.56
	37	chile_7_b	10 800	1.69%	1	10 620	0.00%	11.83	10 620	0.00%	11.83	10 620	0.00%	11.83
	38	chile_8_b	30 800	10.32%	1	27 920	0.00%	10.79	27 920	0.00%	10.79	27 920	0.00%	10.79
	39	chile_9_b	63 500	31.42%	1	48 320	0.00%	18.60	48 320	0.00%	18.60	48 320	0.00%	18.60
	40	chile_10_b	12 600	1.45%	1	12 420	0.00%	11.44	12 420	0.00%	11.44	12 420	0.00%	11.44
	41	chile_11_b	34 400	9.14%	1	31 520	0.00%	11.90	31 520	0.00%	11.90	31 520	0.00%	11.90
	42	chile_12_b	68 900	28.26%	1	53 720	0.00%	16.20	53 720	0.00%	16.20	53 720	0.00%	16.20
c) Emphasize minimizing delay	43	chile_1_c	8 340 000	2.96%	1	8 100 001	0.00%	9.68	8 100 001	0.00%	9.68	8 100 001	0.00%	9.68
	44	chile_2_c	18 300 000	15.97%	1	15 780 002	0.00%	11.48	15 780 002	0.00%	11.48	15 780 002	0.00%	11.48
	45	chile_3_c	31 380 000	24.82%	1	25 140 002	0.00%	34.71	25 140 002	0.00%	34.71	25 140 002	0.00%	34.71
	46	chile_4_c	8 580 000	4.38%	1	8 220 002	0.00%	11.04	8 220 002	0.00%	11.04	8 220 002	0.00%	11.04
	47	chile_5_c	22 680 000	20.38%	1	18 840 002	0.00%	13.05	18 840 002	0.00%	13.05	18 840 002	0.00%	13.05
	48	chile_6_c	47 580 000	52.50%	1	31 200 002	0.00%	43.20	31 200 002	0.00%	43.20	31 200 002	0.00%	43.20
	49	chile_7_c	10 800 000	3.45%	1	10 440 002	0.00%	12.48	10 440 002	0.00%	12.48	10 440 002	0.00%	12.48
	50	chile_8_c	30 800 000	14.24%	1	26 960 002	0.00%	16.32	26 960 002	0.00%	16.32	26 960 002	0.00%	16.32
	51	chile_9_c	63 500 000	34.76%	1	47 120 003	0.00%	36.98	47 120 003	0.00%	36.98	47 120 003	0.00%	36.98
	52	chile_10_c	12 600 000	2.94%	1	12 240 002	0.00%	11.44	12 240 002	0.00%	11.44	12 240 002	0.00%	11.44
	53	chile_11_c	34 400 000	12.57%	1	30 560 002	0.00%	15.04	30 560 002	0.00%	15.04	30 560 002	0.00%	15.04
	54	chile_12_c	68 900 000	31.19%	1	52 520 003	0.00%	34.02	52 520 003	0.00%	34.02	52 520 003	0.00%	34.02
		Average:		**17.07%**	**1**		**0.00%**	**18.93**		**0.00%**	**18.93**		**0.00%**	**18.93**

Figure 4.13: *CPU time required for computing the best solution. Network 2 (Chile).*

	Network 1				
	Instance	Optimal Solution (Cost [$])	Total Delay [sec]	Delay GAP [%]	Average Delay GAP [%]
a) Balance stability and minimizing delay	france_1_a_7	3714	3077	0.00%	0.00%
	france_2_a_7	7546	6589	0.00%	
	france_3_a_7	13026	11268	0.00%	
	france_4_a_7	6230	5336	0.00%	
	france_5_a_7	14703	12828	0.00%	
	france_6_a_7	24828	21551	0.01%	
b) Emphasize stability	france_1_b_7	4005	3292	6.99%	2.06%
	france_2_b_7	7546	6589	0.00%	
	france_3_b_7	13078	11365	0.86%	
	france_4_b_7	6521	5553	4.07%	
	france_5_b_7	14703	12828	0.00%	
	france_6_b_7	25865	21648	0.46%	
c) Emphasize minimizing delay	france_1_c_7	3654002	3077	0.00%	0.00%
	france_2_c_7	7546000	6589	0.00%	
	france_3_c_7	12966004	11268	0.00%	
	france_4_c_7	6170002	5336	0.00%	
	france_5_c_7	14703000	12828	0.00%	
	france_6_c_7	24751006	21549	0.00%	

	Network 2				
	Instance	Optimal Solution (Cost [$])	Total Delay [sec]	Delay GAP [%]	Average Delay GAP [%]
a) Balance stability and minimizing delay	chile_1_a	8115	7375	0.00%	0.00%
	chile_2_a	15810	14515	0.00%	
	chile_3_a	25170	23160	0.00%	
	chile_4_a	8250	7495	0.00%	
	chile_5_a	18870	16995	0.00%	
	chile_6_a	31230	28038	0.00%	
	chile_7_a	10470	9714	0.00%	
	chile_8_a	26990	24970	0.00%	
	chile_9_a	47150	43218	0.00%	
	chile_10_a	12270	10914	0.00%	
	chile_11_a	30590	27370	0.00%	
	chile_12_a	52550	46818	0.00%	
b) Emphasize stability	chile_1_b	8160	7435	0.81%	3.01%
	chile_2_b	16740	15475	6.61%	
	chile_3_b	26340	24111	4.11%	
	chile_4_b	8400	7615	1.60%	
	chile_5_b	19800	17954	5.64%	
	chile_6_b	32400	29000	3.43%	
	chile_7_b	10620	9834	1.24%	
	chile_8_b	27920	25929	3.84%	
	chile_9_b	48320	44173	2.21%	
	chile_10_b	12420	11034	1.10%	
	chile_11_b	31520	28329	3.50%	
	chile_12_b	53720	47775	2.04%	
c) Emphasize minimizing delay	chile_1_c	8100001	7375	0.00%	0.00%
	chile_2_c	15780002	14515	0.00%	
	chile_3_c	25140002	23160	0.00%	
	chile_4_c	8220002	7495	0.00%	
	chile_5_c	18840002	16995	0.00%	
	chile_6_c	31200002	28038	0.00%	
	chile_7_c	10440002	9714	0.00%	
	chile_8_c	26960002	24970	0.00%	
	chile_9_c	47120003	43218	0.00%	
	chile_10_c	12240002	10914	0.00%	
	chile_11_c	30560002	27370	0.00%	
	chile_12_c	52520003	46818	0.00%	

Delay GAP = (Delay of Solution - D)/D**
D = Optimal Delay*

Figure 4.14: *Impact of different objective functions on the total delay. Left: Results for Network 1. Right: Results for Network 2*

Chapter 5

SAPI: Statistical Analysis of Propagation of Incidents

Contents

Abstract of the Chapter

In this chapter you will discover SAPI (Statistical Analysis of Propagation of Incidents), an original method to solve the railway rescheduling problem. This method analyzes several factors for predicting the consequences of disruptions and uses this information to reduce the search space of the inherent optimization problem. We applied this method using the MIP formulation presented in Chapter 4. SAPI is tested and compared with the other methods presented in this Thesis, showing that it is faster than the other MIP-based methods.

5.1 Description of the Method

We propose a method called SAPI (Statistical Analysis of Propagation of Incidents) that performs a statistical analysis of possible propagation of the incidents before solving

the MIP presented in Chapter 4. This analysis helps to reduce the problem and to concentrate the effort in concerned trains. A sequence of the steps of this method is showed in Figure 5.1.

The thesis of this method is that events could be affected for a given incident with a certain probability. SAPI calculates these probabilities using a logistic regression model where the predictor variables are easy to calculate. With these probabilities, the method reduces the problem by hard and soft fixing integer variables in the MIP, without losing much of the quality of the solution.

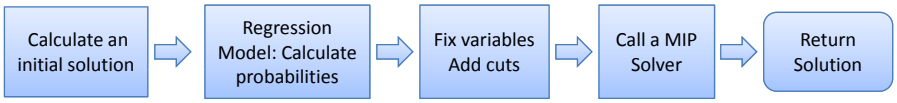


Figure 5.1: *Sequential steps of SAPI method*

5.1.1 Initial Solution

The first step is the construction of an initial solution. This solution is needed to calculate some of the predictor variables of the regression model. Also, we use the objective function value of this solution as an upper bound in the next call of the MIP Solver.

The fastest way to have a feasible (initial) solution is *right-shift rescheduling* (RS) (see Section 4.3.1). RS gives a solution maintaining the original order of trains, keeping the current assignment of tracks/platforms, and forbidding unplanned stops. To achieve this objective, some additional constraints are added to the MIP model presented in Chapter 4:

$$\begin{aligned} &\lambda_{k\hat{k}} = 0 && \forall k, \hat{k} \in L_j, j \in B : k < \hat{k} \\ &\gamma_{k\hat{k}} = \gamma^0_{k\hat{k}} && \forall k, \hat{k} \in L_j, j \in B : k < \hat{k} \\ &q_{kt} = q^0_{kt} && \forall k \in L_j, t \in P_j, j \in B \\ &y_k = 0 && \forall k \in E \end{aligned}$$

Where $k < \hat{k}$ means that $\hat{k}$ is any event following event k, in relation to the original schedule.

Parameters $\gamma^0_{k\hat{k}} = 1$, if event k occurs before event $\hat{k}$ in the original schedule and $q^0_{kt} = q^0_{\hat{k}t} = 1, t \in P_j$, with $k, \hat{k} \in L_j, j \in B$, $\gamma^0_{k\hat{k}} = 0$ otherwise.

After these modifications all integer variables are fixed and solving this problem is easy. Nevertheless, the quality of this solution is not good enough, because tracks having idle capacity are not correctly exploited. Moreover, delays are easily propagate since changing the order of trains is not allowed, *i.e.*, a delayed train will propagate it to all the following trains along its itinerary.

5.1.2 Logistic Regression

An interesting aspect of SAPI is how to calculate the probabilities. We propose to use a logistic regression model that has been used in many fields from medicine to social sciences.

Logistic regression is part of a category of statistical models called generalized linear models. General linear models analyze one or more continuous dependent variables and one or more independent variables. In particular, logistic regression analysis allows estimating multiple regression models when the response can be scored 0 or 1.

Here we consider two possible outcomes: 1, if the variable associated to an event is affected by the incidents, and 0 otherwise. Thus, the dependent variable is dichotomous, that is, it can take the value 1 with a probability of success θ, or the value 0 with probability of failure $1 - \theta$. On the other hand, independent (predictor) variables in logistic regression can take any form. That is, logistic regression makes no assumption about the distribution of the independent variables, *i.e.*, they do not have to be normally distributed, linearly related or of equal variance within each group.

The procedure uses the probabilities estimated by the logistic regression model to reduce the number of integer variables and the size of the feasible region. In particular, we reduce the combinatory of rescheduling variables (λ). For the rest of the integer variables (γ, y, q) it was not possible to find predictor variables statistically significant.

The regression model is defined by the following equation:

$$\theta^{\lambda}_{k\hat{k}} = \frac{exp(\eta)}{1 + exp(\eta)} \quad \forall k, \hat{k} \in L_j, j \in B : k < \hat{k} \tag{5.1}$$

Where:

$$\eta = \alpha_\lambda + \beta^1_\lambda \rho^1_k + \beta^2_\lambda \rho^2_k + \beta^3_\lambda \rho^3_k + \beta^4_\lambda \rho^4_{k\hat{k}} + \varepsilon_{k\hat{k}}$$

$\theta^{\lambda}_{k\hat{k}}$	:	Probability that variable $\lambda_{k\hat{k}} \neq 0$ in the optimal solution, where $k, \hat{k} \in L_j, j \in B : k < \hat{k}$
α_λ	:	Constant of the regression equation.
β^i_λ	:	Regression coefficient of the predictor variable i.
$\varepsilon_{k\hat{k}}$	:	Error term.

$k < \hat{k}$ means that $\hat{k}$ is any event following event k, in relation to the original schedule.

Section 5.1.4 details the process of estimating the coefficients for the regression model and also discuss the study of the statistical significance of the model.

The predictor variables are:

ρ_k^1 Measure of the density near to event k. We use the average time between trains divided by the number of trains in the block associated with the current event. To calculate this, we just considerate the trains in the interval two times longer than the largest incident. Finally, this value is divided by the duration of the longer delay. We expect that variables associated with *dense blocks* are prone to be affected by an incident.

ρ_k^2 The relative difference of arrival times between event k to the last known incident. This value is calculated as the time interval, using the original schedule, between the event k and the last incident, only if this value is a positive number, otherwise it is zero. In order to obtain a relative value, this value is then divided by the duration of the longest delay.

ρ_k^3 Measure of the impact of the incidents in event k. To compute this value, we calculate the difference of the arrival time between the initial solution (calculated by right-shift rescheduling) and the original plan. This value corresponds to the delay of the event after right-shift rescheduling. We expected that events with a large impact are more concerned by a possible reassignment of track/platform or to be rescheduled.

$\rho_{k\hat{k}}^4$ The relative difference of arrival times between event $\hat{k}$ and event k. When a train is late, it could be interesting to change the order of trains to minimize the impact of incidents. Then, we expect to have a negative correlation between this distance and the possibility of changing the order of the trains in a block. For example, if a train has a delay in the morning, the optimal solution of this problem might require changing the order of this train with other trains in the morning. However, there is a very low probability that the same train must change its order with trains late in the afternoon. Obviously, the number of probable trains affected is proportional to the length of the delay, which explains why we use a relative measure of this distance, *i.e.*, we divide the time interval by the length of the largest incident.

It should be noted that we preferred to use relative values rather than absolute ones. The explication is quite simple: the goal is that the values of the regression parameters remain valid for any kind of instance and not only for the one used as statistical sample.

Once the probabilities are calculated, it is possible to define three different subsets for the variables λ. The first subset U_1 corresponds to the set of indices of variables with the lower probability of being affected by incidents. For selecting these variables, we choose the variables with a probability in the interval $[0, \phi_\lambda^1]$. Because they have a very low probability of being affected, we fix them using their value in the original schedule. A second subset U_2 is composed by the indices of the variables that have a probability in the interval $]\phi_\lambda^1, \phi_\lambda^2]$. As these variables have a higher probability to have a different value in the optimal solution, we should leave them free. However, not all the variables in this subset are going to change, then we limited the number of changes by a local branching cut, originally proposed in (Fischetti and Lodi, 2003) and applied

to this problem in Chapter 4. The last subset to be defined is U_3 with the indices of the rest of the variables (i.e., with probabilities in $]\phi_\lambda^2, 1]$) and a local branching cut is also utilized for these variables. The decision of adding local branching cuts also in this last subset is explained by empirical results. The experiments have shown that not all the variables in U_3 are already affected by the incidents and adding local branching cuts in this subset has a marginal (but positive) impact in the processing time without degrading the quality of the solution. Thus, we have:

$$U_1 \triangleq \{(k,\hat{k}) | k,\hat{k} \in L_j, j \in B : k < \hat{k} \wedge \theta^\lambda_{k\hat{k}} \in [0, \phi^1_\lambda]\}$$
$$U_2 \triangleq \{(k,\hat{k}) | k,\hat{k} \in L_j, j \in B : k < \hat{k} \wedge \theta^\lambda_{k\hat{k}} \in]\phi^1_\lambda, \phi^2_\lambda]\}$$
$$U_3 \triangleq \{(k,\hat{k}) | k,\hat{k} \in L_j, j \in B : k < \hat{k} \wedge \theta^\lambda_{k\hat{k}} \in]\phi^2_\lambda, 1]\}$$

Recall that $k < \hat{k}$ means that $\hat{k}$ is any event following event k, in relation to the original schedule.

Finally, with this information, we could add the next constraints into the original model:

$$\lambda_{k\hat{k}} = 0 \qquad \forall (k,\hat{k}) \in U_1 \tag{5.2}$$
$$\sum_{(k,\hat{k}) \in U_2} \lambda_{k\hat{k}} \leq LB^1_\lambda \tag{5.3}$$
$$\sum_{(k,\hat{k}) \in U_3} \lambda_{k\hat{k}} \leq LB^2_\lambda \tag{5.4}$$

Where LB^1_λ and LB^2_λ are parameters of the method. They represent the maximal number of events that are going to change their initial schedule.

5.1.3 Creating and Solving the Reduced Subproblem

The last step is to add constraints (5.2, 5.3, 5.4) to the original MIP model. Thanks to them, the solution of this reduced MIP model is much easier than the original one and these simple changes in the original MIP model have a strong impact on the processing time.

Note that many integer variables are directly fixed because of Constraints (5.2). Moreover, adding these constraints may imply that the preprocessing procedure of the MIP solver will eliminate not only variables λ, but also many other variables and constrains because of the nature of the model. Consider, for example, Constraints (4.17) and (4.18). For these constraint, $\lambda_{k\hat{k}} = 0$ implies that they become inactive and the preprocessing could eliminate them.

In addition, a limited number of changes are allowed for the rest of the variables because of Constraints (5.3) and (5.4). We have shown in Chapter 4 that this kind of cuts helps to solve this problem quickly.

5.1.4 Estimating Parameters and Regression Coefficients of SAPI

To use SAPI, it is necessary to evaluate all parameters of the method. There are three types of parameters: cuts parameters, regression coefficients, and limits of the intervals (cutoffs)

Cuts parameters

The first are the parameters for the cuts: LB^1_λ, LB^2_λ. Relative small values will limit the search in a small neighborhood, and probably the problem can be solved quickly. In contrast, large values for these parameters will increase the computing time but better solutions should be obtained. In practice, a good combination of these parameter can be found empirically.

Regression coefficients

Other important parameters are the coefficients for the regression model: α_λ, β^1_λ, β^2_λ, β^3_λ and β^4_λ. A statistical sample is needed to make inference of these coefficients. To achieve this objective we use a train network and solve the original MIP model optimally with different kinds of incidents using the exact method presented in Chapter 4. Because it is a training process, the processing time has not the same importance as solving real instances, with real incidents, where a real-time solution is required. It is important to remark that these estimations (of parameters) can be improved gradually using continuously this method, says daily. Indeed, we increase the size of the sample, gaining more and more information each time we use SAPI.

In logistic regression, as other regression models, the coefficients could be estimated by maximum likelihood (interested readers can find in (Aldrich, 1997) a good introduction to this method). The objective of maximum likelihood parameter estimation is to find the values for the regression coefficients that maximize the probability or likelihood of the statistical sample data. This non-constrained optimization problem is solved by finding the values where the derivative of the function is equal to zero. In practice, approximative numerical algorithms are used to estimate the coefficients for logistic regression model. One example is Newton-Raphson that chooses arbitrary initial estimates of the regression coefficients, such as $\alpha_\lambda = 0$. Then, at each iteration, the value of the coefficients are updated until the difference between the values of two consecutive iterations is less than a tolerance limit.

Nevertheless, estimating the values of the coefficients is only half part of the whole process. Thus, it is very important to study the statistical significance for the regression model in general and for every coefficient in particular.

i) Significance of the regression model (Analysis of Deviance):

An Analysis of Deviance table summarizes information about the variation in the response for the set of data. For an example, see Table 5.1. In this case we have two

source of variability: the model and the residual. The residual (*i.e.*, error) is the remainder that is not systematically explained by the model.

In order to validate the results of the estimation, we expect that a large proportion of the total variance is explained by the model. Statistically, it is possible to determine this proportion using the "p-value" associated to the model (see for example the first line of Table 5.1).

To calculate the p-value, it is necessary to determine the degree-of-freedom (Df) of each component of variance. For the model, this value corresponds to the total number of regression coefficients less 1 (*e.g.*, in this case we have 5 regression coefficients less 1 $\implies Df = 4$). The total Df is calculated by the total number of elements in the sample less one. Therefore, the Df for the residual is just the difference between these two values.

A correct interpretation of this analysis is: if the p-value for the model is less than α, there is a statistically significance relationship between all the explicative variables and the response variable at the $(1-\alpha) * 100$ % confidence level. On the other hand, if the p-value for the residuals is greater than δ, the model is not significantly worse than the best possible model for this data at the $(1-\delta) * 100$ % or higher confidence level.

We are interested to have higher confidence level for the model. As a consequence, the smaller p-value for the model, the more sure we are to believe that the regression model fits adequately to data *i.e.*, there is no evidence to think the contrary.

ii) Significance of regression coefficients (Likelihood-ratio test):

We use this test to determine if the coefficients of the regression model are significative individually. For an example see Table 5.2. In this table, the column "Estimate" corresponds to the estimated values for the coefficients. Additionally, the "p-value" is used to determine the significance of each coefficient.

The correct interpretation is: if the p-value for the regression coefficient is less than α, the estimate is statistically significance at the $(1-\alpha) * 100$ % confidence level. As a consequence, the smaller the likelihood ratio (p-value), the stronger the relationship.

It should be noted that there are other interesting statistical tests, tables and graphics that can be used in logistic regression models. Some examples are: Chi-Square goodness of fit test, correlation matrix for coefficient estimates, influential points for value, analysis of residuals, prediction histograms, and many others. We consider that the tests presented in this section are enough to achieve the objectives for our application.

Limits of the intervals (cutoffs)

The last parameters are the limits of the intervals of probabilities (cutoffs): $\phi_\lambda^1, \phi_\lambda^2$. There is a tradeoff between performance and quality of the solution. Thus, the more the value of ϕ_λ^1 is close to 1, the more we are going to have a solution quickly. To estimate these

values, we study the prediction performance of the regression model defining the interval to the desired percentage of events correctly predicted (for an example see Section 5.3).

5.2 Iterative SAPI

In our first experiments we noted that SAPI obtained solution faster than we expected. We were then immediately interested in using more processor time to improve the current solution. In this section we present an iterative SAPI method that improves the current solution increasing the search space with two strategies: decrease the number of fixed variables and increase the size of the feasible region by increasing the upper bound of the added local branching cuts. Figure 5.2 shows the basic structure of this method.

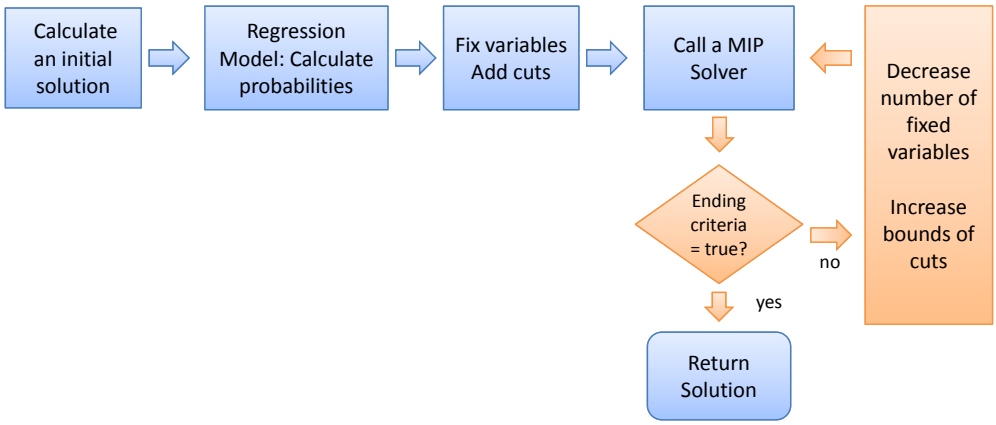


Figure 5.2: *Sequential steps of Iterative SAPI method*

Algorithm 3 presents the pseudocode of this method. We could divide the parameters of this algorithm in five: mathematical problem, cuts parameters, SAPI limit probabilities, iteration parameters and regression coefficients. The mathematical problem P_0 is an instance using the formulation presented in the Chapter 4. Cuts parameters are LB^1_λ and LB^2_λ. They are integer values corresponding to the initial value of the right side for the constraints (5.3) and (5.4). The cutoff parameters are ϕ^1_λ, ϕ^2_λ used to delimit the subsets of indices U_1, U_2, U_3. The parameters used to increase the search space are δ^{LB_λ}, δ^{ϕ^1}, δ^{ϕ^2}. The first one, (δ^{LB_λ}), denotes the change, at each iteration, of the right side for constraints (5.3) and (5.4). In contrast, δ^{ϕ^1}, δ^{ϕ^2} establish the changes for the construction of the subsets U_1, U_2, and U_3 that define a stratification of the variables. Finally, the last parameters of this algorithm correspond to the coefficients of the regression models α_λ, β^1_λ, β^2_λ, β^3_λ. These parameters have to be estimated before using this algorithm and could be fixed for all instances in the same network.

The first step is to define the ending criteria for this procedure. Some possibilities are the number of iterations, a time limit or a number of iterations without changes in the solution. Thus, Line 1 assigns the value *False* to a boolean variable called *EndingCriteria*. We employ the knowledge of this problem to compute quickly an initial solution (Line 2) using a function called `RightShift(`P_0`)`. This function gives a feasible solution using the approach presented in Section 5.1.1. This solution helps also to calculate the values of the probabilities.

Lines 3 to 9 are used to calculate the probability of each variable λ to be affected by the incidents in the current instance. Then, for every event k it is necessary to calculate the predictor factors with function `EvaluateFactor(k)`. With the value of these factors and the value of the regression coefficients, the probability is calculated on Line 8. After this step, the method evaluate a boolean function called `EvaluateEndingCriteria()`. The answer is *True* or *False* depending on the current status (for example the elapsed time) and the ending criteria used.

The control structure `while` (Lines 11 to 20) defines one iteration of this procedure. The condition to execute the iteration is the ending criteria. To create (update) the stratification of the variables depending on their probabilities for being affected by the incidents a function called `GenerateStratification(`$\phi^1_\lambda, \phi^2_\lambda$`)` is used on Line 12. This function modifies the composition of the subsets of indices U_1, U_2, U_3. Then, Line 13 creates a subproblem, denoted by P_1, that corresponds the original problem (P_0) plus the local branching cuts defined by Constraints (5.2), (5.3), and (5.4). Once the new subproblem is created, Line 14 solves P_1 using the previous solution X as starting values and limiting the processing to the remaining time.

Lines 15 to 18 change the values of LB^1_λ, LB^2_λ,ϕ^1_λ, ϕ^2_λ for the next iteration. The goal is to enlarge the search space, allowing more changes of tracks and changes of orders of trains with respect to the original schedule and also fixing fewer variables. It is important to remark that this procedure always improves (or conserves) the current solution since it defines a relaxation of the previous subproblem. Finally, the ending criteria is reevaluated on Line 19 and Line 11 is performed again. This procedure continues until the function `EvaluateEndingCriteria()` returns *true*. In that case, Line 21 is executed returning the last (and best) solution found so far.

5.3 Computational Experiments

The goal of the computational experiments is to evaluate the performance of the suggested methods. To achieve this objective, we considered a heterogeneous set of instances using different combination of the coefficients in the objective function. The results are compared with several methods proposed in Chapter 4, under the same environment: algorithm coded in Visual Studio 2005 (C♯) using IBM ILOG CPLEX version 11.1[3] as a black-box MIP solver and running on a single processor IBM compatible PC Intel Core 2, 1.66 GHz, and 2 GB of main memory.

[3]http://www.ilog.com

Algorithm 3 Iterative SAPI Method

Require: P_0, LB^1_λ, LB^2_λ, ϕ^1_λ, ϕ^2_λ, δ^{LB_λ}, δ^{ϕ^1}, δ^{ϕ^2}, α_λ, β^1_λ, β^2_λ, β^3_λ , β^4_λ

$EndingCriteria \Leftarrow False$
$X \Leftarrow$ RightShift(P_0)
for all $\lambda_{k\hat{k}}$ with $k, \hat{k} \in L_j, j \in B : k < \hat{k}$ **do**
  for all $PredictorVariable_i$ $i = 1, 2, 3$ **do**
    $\rho^i_k \Leftarrow EvaluateFactor_i(k)$
  end for
  $\rho^4_{k\hat{k}} \Leftarrow EvaluateFactor_4(k\hat{k})$
  $\theta^\lambda_{k\hat{k}} \Leftarrow EvaluateRegression_\lambda(\rho^1_k, \rho^2_k, \rho^3_k, \rho^4_{k\hat{k}})$
end for
$EndingCriteria \Leftarrow$ EvaluateEndingCriteria()
while $(EndingCriteria \neq True)$ **do**
  $(U_1, U_2, U_3) \Leftarrow$ GenerateStratification($\phi^1_\lambda, \phi^2_\lambda$)
  $P_1 \Leftarrow P_0$ adding constraints (5.2), (5.3), and (5.4)
  $X \Leftarrow$ MipSolve(P_1,StartSolution(X))
  $LB^1_\lambda \Leftarrow LB^1_\lambda + \delta^{LB_\lambda}$
  $LB^2_\lambda \Leftarrow LB^2_\lambda + \delta^{LB_\lambda}$
  $\phi^1_\lambda \Leftarrow \phi^1_\lambda - \delta^{\phi^1}$
  $\phi^2_\lambda \Leftarrow \phi^2_\lambda - \delta^{\phi^2}$
  $EndingCriteria \Leftarrow$ EvaluateEndingCriteria()
end while
return X

We tested the following methods:

- Method S1 (SAPI Heuristic, stop criterion: 5 minutes): *SAPI* algorithm limited to 300 seconds of running time.
- Method S2 (SAPI Heuristic, stop criterion: 5 iterations): *SAPI* algorithm limited to 5 iterations.
- Method S3 (SAPI Heuristic, stop criteria: 1 iteration (or) 5 minutes): *SAPI* algorithm limited to 5 minutes of running time or one complete iteration. The limit of the processing time responds to real requirements of railway companies. Additionally, the algorithm performs only one iteration, i.e., if the first iteration finishes before 5 minutes, the algorithm stops.
- Method S4 (SAPI Exact, 1 iteration of SAPI + CPLEX) This method uses the output of Method S3 as an initial solution for IBM ILOG CPLEX removing any additional cut and letting all variable be free. As a consequence, the feasible region corresponds to the original search space and the optimal solution of this method coincides with the optimal solution of the original problem. This method is described in Section 4.4 using SAPI as the heuristic method.

They are compared with the following methods proposed in Chapter 4:

- Method O1 (Right-shift Heuristic), *Right-shift rescheduling* The initial solution of SAPI.
- Method O4 (I-LS Heuristic. Stop criteria: 1 iteration), *MIP-based local search method* using right-shift as initial solution. It limits the search space around the original non-disrupted schedule with local-branching-type cuts added to the model.
- Method O3 (LS + CPLEX Exact): *MIP-based local search method + CPLEX*. This is a variation of Method O4 which, in a second phase, solves the original (complete) MIP formulation with an improved upper bound. (see Section 4.4)

In order to compare, we use exactly the same scenarios for the objective function coefficients considered in Chapter 4:

- Emphasize minimization of delay: the coefficients in the objective function strongly penalize the delays.
- Emphasize stability: we consider that the number of changes of track/platform/stops is inversely proportional to the stability of the schedule. Thus, the coefficients in the objective function strongly penalize the changes of tracks/platforms/stops.
- Balance stability and minimizing delay: an equilibrated combination of coefficients in the objective function is considered.

Two different networks have been used for the computational experiments. The first is a line, consecutive set of stations and sections, located in France. The second one is a network, with a topology of a tree where a station can be connected by more than two sections, located in Chile. Both were described in Chapter 4. The values used in the experiments for the parameters associates to cuts were $LB_\lambda^1 = 100$, $LB_\lambda^2 = 100$. These

values have been chosen following recommendations of Local Branching cuts authors in (Fischetti and Lodi, 2003) and our own numerical experiments.

Estimating Regression Coefficients

In this subsection we estimate the coefficients of the regression model used in the experiments. First, we explain the statistical sample used for the estimation. Then, we study the significance of the regression model in general and of each regression coefficient in particular.

The sample

A statistical sample is needed to estimate the regression coefficients. We considered one additional instance of Network 1 (i.e., the French railway network presented in Section 4.5) to be used as a sample. This instance is composed of 26 trains running over 43 stations by limiting the recovery horizon to 3 hours after the first incident. A total of 1431 events are rescheduled after 2 incidents located in the both extreme of the network. Each incident causes an initial delay of 20 minutes.

This is a very little instance in relation to those used in the experiments. Then, it was easy to find an optimal solution executing the algorithm described in Section 4.4. Using this optimal solution, we prepared a statistical sample with the value of the decision variables (λ) and the value of the predictor variables (ρ). This sample does not contain the values for all variables $\lambda_{k\hat{k}}$ because of the large number of them. We select the first 4999 variables ordered by the original arrival time of event k. Note that the first events are the more affected by the incidents. This sample was then processed by a statistical software namely Statgraphics 4.

It is very important to understand that this instance is only used for this estimation and not for the rest of the experiments. In the same way, we expressly did not choose the same perturbed trains for the rest of the experiments. In order to demonstrate that the quality of the coefficients does not depend on the instance, we decide to use the same coefficients for all problem instances in both networks.

Significance of the regression model

Table 5.1 presents an Analysis of Deviance used to evaluate the significance of the regression model. For the reason that the P-value for the model in the analysis of deviance is also less than 0.01, there is a statistically significant relationship between the variables at the 99% confidence level. In addition, the P-value for the residuals is greater than 0.10, indicating that the model is not significantly worse than the best possible model for this data at the 90% or higher confidence level.

The percentage of deviance explained by the model is equal to 86.56 (see Table 5.1). This statistic is similar to the usual R^2 statistic. That is, 86.56% of the variability of the

Source	Deviance	Df	P-Value
Model	1381.31	4	0
Residual	214.53	4994	1
Total (corr.)	1595.84	4998	

Table 5.1: *Analysis of Deviance: Percentage of deviance explained by model = 86.56*

value of variables λ in the optimal solution can be explained by this regression model without solving the problem. In order to compare this regression model, we evaluate also the R^2 for a linear (multiple) regression model, using the same sample and the same predictor variables. The result was $R^2_{linear} = 29.88\%$. In conclusion, this logistic regression is more adequate to predict the optimal value of variables λ than a linear (multiple) regression model using the same predictor variables.

Significance of each regression coefficient individually

Table 5.2 shows estimated values for the regression parameters and the Likelihood-ratio test. Notice that the highest P-value (likelihood ratio) is 0.0020, for parameter β^2_λ (see Table 5.2). Because the P-values are less than 0.01, the estimates are statistically significant at the 99% confidence level. Consequently, we do not consider removing any variables from the model.

Parameter	Estimate	P-Value
α_λ	-117.60	-
β^1_λ	25.90	0.0000
β^2_λ	8.33	0.0020
β^3_λ	117.54	0.0000
β^4_λ	-5.49	0.0000

Table 5.2: *Estimated Regression Model (Maximum Likelihood and Likelihood-ratio test)*

Prediction capability

Figure 5.3 shows a summary of the prediction capability of the fitted regression model. If the predicted value is larger than the cutoff (ϕ^1_λ), the response is predicted to be TRUE. In our application, TRUE means that variable $\lambda_{k\hat{k}} \neq 0$ (in the optimal solution) and this variable is not fixed. If the predicted value is less than or equal to the cutoff, the response is predicted to be FALSE. In that case, variable $\lambda_{k\hat{k}} = 0$. The figure shows the percent of the observed data correctly predicted for various cutoff values in the sample. For example, using a cutoff equal to 0.4, 98.93% of all TRUE responses were correctly predicted, while 99.31% of all FALSE responses were correctly predicted, for a total of 99.30%. This cutoff value maximizes the total percentage of correct TRUE and FALSE responses and thus may provide a good value for predicting additional individuals.

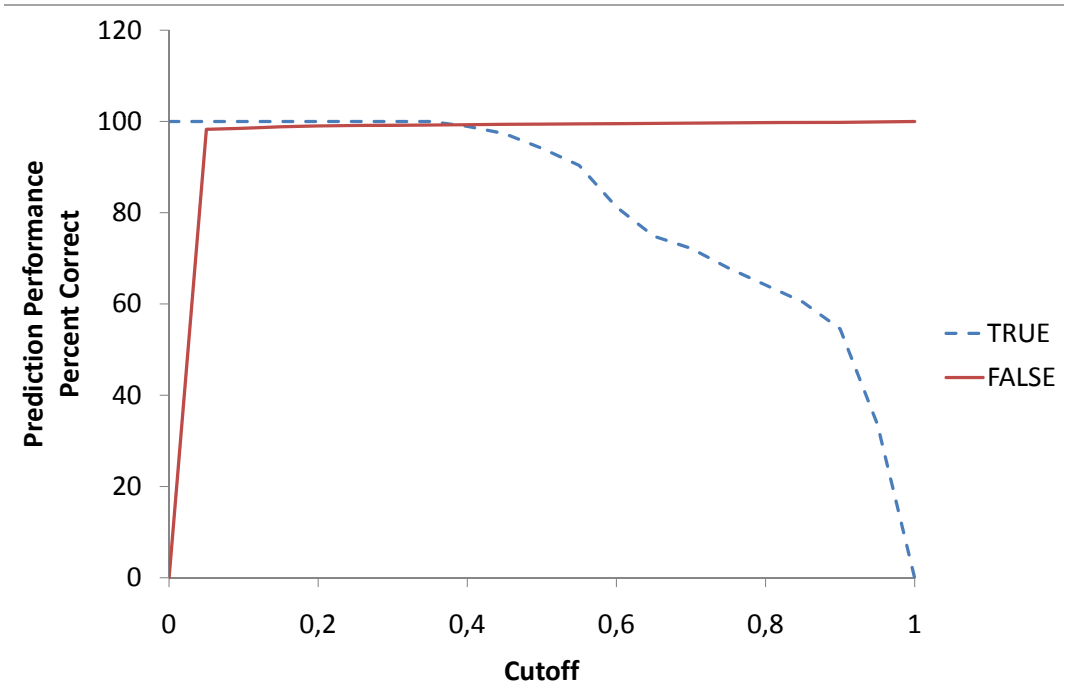


Figure 5.3: *Prediction performance of the regression model.*

Nevertheless, we do not use this cutoff value for the experiments, because fixing less variables should help calculating a better solution. In particular we define the cutoff value $\phi_\lambda^1 = 0.01$, where 100% of all TRUE responses in the sample were correctly predicted, while 97.87% of all FALSE responses were correctly predicted, for a total of 97.99%. On the other hand, we choose arbitrarily the value of $\phi_\lambda^2 = 0.5$. Some experiments showed that varying this value does not affect significantly the final results.

5.4 Results and Conclusions

Figure 5.4 shows the results of the experiments carried out on the two networks with regression coefficients defined as explained above. Column (*Cost [$]*) is for the value of the objective function. Column (*CPU [sec]*) is the total execution time of the methods in seconds. Column *GAP [%]* is the relative deviation between the optimal solution ($y*$) and the heuristic solution (y) for the corresponding method. Let us recall that the optimal solution of our rescheduling problem is found by Method S4 or O3. This gap is computed as follows: $GAP_y = (y - y*)/y*$.

The results show that instances of Network 1 are harder to solve than instances of Network 2. Such results may be explained by a greater number of events to be rescheduled on Network 1, a higher traffic density, and the correlation between these events.

Four different variations of SAPI are presented in Figure 5.4. In addition, Figure 5.5 and 5.6 present the results of SAPI compared with those obtained in Chapter 4. The results obtained by the different methods are:

Method S1: (SAPI Heuristic, stop criteria: 5 minutes). The method iterative performs as many iterations as possible in 300 seconds. The average gap is 0.49% for Network 1 and 0.05% for Network 2.

Method S2: (SAPI Heuristic, stop criteria: 5 iterations). This variation performs 5 iterations of the algorithm presented in Section 5.2. It is possible to appreciate that it is able to find the optimum for all instances of Network 1 but the average for the processing time was 1031 seconds. The results for Network 2 are quite different. The quality of solutions is similar than Method S1, but in average is faster.

Method S3: (SAPI Heuristic, stop criteria: 1 iteration or 5 minutes). The average gap is 0.49% for Network 1 and 0.05% for Network 2. Its average total running times are 128 seconds for Network 1 and 15 seconds for Network 2. This method is selected to be compared with Method O4 because both are heuristic and consider the same end criteria (see Figure 5.5). Analyzing these results it is possible to conclude that SAPI is faster finding good solutions, but with a small possibility to return a suboptimal solution. That is the case for 6 of 54 instances. Probably, some instances require making non-evident changes in the order of trains, and these changes have a very low probability in the regression model. SAPI assures good forecasting for most of the cases with an average gap less than 1%. There are two instances where we calculate the result corresponding to 1 iteration without the limit of 300 seconds: Instances 6 and 12 of Network 1. For Instance 6 we have a cost of \$24828 (gap = 0%) using a CPU time of 645 seconds (not reported in the Figure). For Instance 12 we have a cost of \$25865 (gap = 0%) using a CPU time of 961 seconds (not reported in the Figure). The CPU time to calculate the best solution using S3 is 86 and 14 seconds for Network 1 and Network 2 respectively (Figure 5.6). An important observation is the fact that Method S2 needs more than 5 times the time of S3 when the time limit is not reached. We estimate that this is due to the different size of the remained MIP solved by each iteration. Indeed, the size of iteration 2 could more than twice the size of iteration 1 because of the reduction of the cutoff parameters ϕ_λ^1 and ϕ_λ^2 (see lines 17 and 18 of Algorithm 3).

Method S4: (SAPI Exact, 1 iteration of SAPI + CPLEX). It finds the optimal solution for all instances. Its average total running times are 498 seconds for Network 1 and 33 seconds for Network 2. Because both methods are exact, Method S4 is comparable with Method O3 (see Figure 5.5). The results show that in average SAPI (Exact) is 35.82% and 52.86% faster than Method O3 for Networks 1 and 2 respectively. Actually, only 4 of 54 instances are solved faster by Method O4. The CPU time to calculate the best solution using S4 is 135 and 15 seconds for Network 1 and Network 2 respectively (Figure 5.6).

Method O1: (Right-shift Heuristic) Right-shift rescheduling is the fastest. In average, RS is able to compute a feasible solution in 3 seconds for instances of Network 1 and 1 second for Network 2. The average gap obtained by RS for all instances on Network 1 and Network 2 are respectively: 141.93% and 17.07% (thus showing that instances of Network 2 are easier to solve for RS). In fact, since incidents on trains of

Network 2 have a local impact, keeping the original order and avoiding changes is sufficient for many of them. On the contrary, in Network 1, such approach performs very badly since a greater number of trains are affected simultaneously by an incident.

Method O4: (I-LS Heuristic. Stop criteria: 1 iteration or 5 minutes). This is the heuristic solution scheme presented in Chapter 4. Its average total running times are 400 seconds for Network 1 and 21 seconds for Network 2. Note that for these experiments the method is able to find an optimal solution for all instances of both networks, but without proof of optimality. This algorithm is limited to 5 minutes of running time and performs only one iteration. The CPU time to calculate the best solution using O4 is 105 and 19 seconds for Network 1 and Network 2 respectively (Figure 5.6).

Method O3: (LS + CPLEX Exact). This method is the exact solution scheme proposed in Chapter 4. Its average total running times are 776 seconds for Network 1 and 70 seconds for Network 2. The CPU time to calculate the best solution using O3 is 150 and 19 seconds for Network 1 and Network 2 respectively (Figure 5.6).

In conclusion, right-shift rescheduling shown to be the fastest method and is convenient to generate initial solutions. There is no a clear dominance between Method S3 and O4, while one is faster the other one obtains better solutions in average. Considering that operators could be more interesting in having good solutions quickly rather than optimal solutions, the best compromise is obtained with Method S3 that is able to obtain very good solutions (average GAP less than 1%) within 5 minutes of computing time and thus proves to be viable in practice. On the other hand, Method O4 obtains better solutions values with slightly larger CPU times. Finally, regarding exact methods, SAPI (Method S4) is in average faster than Method O3.

We address the RRP with a new solution scheme called SAPI. The method uses the original (non-disrupted) schedule to compute the new one calculating the probability that train are affected given a set of incidents expressed as delays of some trains. This procedure fixes some integer variables and adds linear inequalities (cuts) to keep the order of trains when the probability to be affected is not significant.

An iterative variant of this approach is used to improve the solution when some extra processing time is viable. Using this method we obtain near optimal solutions with a computing time compatible with the context of rescheduling.

It should be noted that the proposed approach is based on the assumption that all incidents are known, i.e., deterministic. However, it is evident that upcoming operations, during the recovery horizon, can also be affected by future incidents. A way of dealing with this problem and mitigating the effects of these possible unknown disruptions is developing a robust optimization procedure where some coefficients in the MIP model are known within certain bounds. Definitely, the study of the robust railway rescheduling problem constitutes a line for future research.

	Instance		Method S1: SAPI (5 min) - Heuristic			Method S2: SAPI (5 Iterations) - Heuristic			Method S3: SAPI (1 Iteration or 5 min) - Heuristic			Method S4: SAPI (1 Iteration + CPLEX) - Exact		
	N°	Name	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]
a) Balance stability and minimizing delay	1	france_1_a_7	3 714	0.00%	300	3 714	0.00%	761	3 714	0.00%	77	3 714	0.00%	213
	2	france_2_a_7	7 546	0.00%	300	7 546	0.00%	715	7 546	0.00%	60	7 546	0.00%	263
	3	france_3_a_7	13 026	0.00%	300	13 026	0.00%	1 105	13 026	0.00%	139	13 026	0.00%	372
	4	france_4_a_7	6 230	0.00%	300	6 230	0.00%	673	6 230	0.00%	75	6 230	0.00%	194
	5	france_5_a_7	14 703	0.00%	300	14 703	0.00%	739	14 703	0.00%	78	14 703	0.00%	285
	6	france_6_a_7	26 460	6.57%	300	24 828	0.00%	2 653	26 460	6.57%	300	24 828	0.00%	1 739
b) Emphasize stability	7	france_1_b_7	4 005	0.00%	300	4 005	0.00%	793	4 005	0.00%	83	4 005	0.00%	239
	8	france_2_b_7	7 546	0.00%	300	7 546	0.00%	745	7 546	0.00%	74	7 546	0.00%	229
	9	france_3_b_7	13 078	0.00%	300	13 078	0.00%	1 037	13 078	0.00%	122	13 078	0.00%	329
	10	france_4_b_7	6 521	0.00%	300	6 521	0.00%	752	6 521	0.00%	96	6 521	0.00%	241
	11	france_5_b_7	14 703	0.00%	300	14 703	0.00%	699	14 703	0.00%	88	14 703	0.00%	291
	12	france_6_b_7	26 461	2.30%	300	25 865	0.00%	2 166	26 461	2.30%	300	25 865	0.00%	1 870
c) Emphasize minimizing delay	13	france_1_c_7	3 654 002	0.00%	300	3 654 002	0.00%	667	3 654 002	0.00%	82	3 654 002	0.00%	192
	14	france_2_c_7	7 546 000	0.00%	300	7 546 000	0.00%	760	7 546 000	0.00%	75	7 546 000	0.00%	258
	15	france_3_c_7	12 966 004	0.00%	300	12 966 004	0.00%	1 117	12 966 004	0.00%	199	12 966 004	0.00%	413
	16	france_4_c_7	6 170 002	0.00%	300	6 170 002	0.00%	697	6 170 002	0.00%	92	6 170 002	0.00%	202
	17	france_5_c_7	14 703 000	0.00%	300	14 703 000	0.00%	851	14 703 000	0.00%	90	14 703 000	0.00%	338
	18	france_6_c_7	24 751 006	0.00%	300	24 751 006	0.00%	1 625	24 751 006	0.00%	277	24 751 006	0.00%	1 292
		Average:		**0.49%**	**300**		**0.00%**	**1 031**		**0.49%**	**128**		**0.00%**	**498**

	Instance		Method S1: SAPI (5 min) - Heuristic			Method S2: SAPI (5 Iterations) - Heuristic			Method S3: SAPI (1 Iteration or 5 min) - Heuristic			Method S4: SAPI (1 Iteration + CPLEX) : Exact		
	N°	Name	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]
a) Balance stability and minimizing delay	19	chile_1_a	8 115	0.00%	300	8 115	0.00%	129	8 115	0.00%	12	8 115	0.00%	21
	20	chile_2_a	15 810	0.00%	300	15 810	0.00%	139	15 810	0.00%	15	15 810	0.00%	24
	21	chile_3_a	25 170	0.00%	300	25 170	0.00%	128	25 170	0.00%	19	25 170	0.00%	34
	22	chile_4_a	8 250	0.00%	300	8 250	0.00%	112	8 250	0.00%	13	8 250	0.00%	21
	23	chile_5_a	18 870	0.00%	300	18 870	0.00%	118	18 870	0.00%	14	18 870	0.00%	27
	24	chile_6_a	31 230	0.00%	300	31 230	0.00%	130	31 230	0.00%	19	31 230	0.00%	49
	25	chile_7_a	10 470	0.00%	300	10 470	0.00%	106	10 470	0.00%	14	10 470	0.00%	23
	26	chile_8_a	26 990	0.00%	300	26 990	0.00%	133	26 990	0.00%	14	26 990	0.00%	27
	27	chile_9_a	47 150	0.00%	300	47 150	0.00%	152	47 150	0.00%	16	47 150	0.00%	48
	28	chile_10_a	12 270	0.00%	300	12 270	0.00%	128	12 270	0.00%	13	12 270	0.00%	23
	29	chile_11_a	30 590	0.00%	300	30 590	0.00%	133	30 590	0.00%	13	30 590	0.00%	28
	30	chile_12_a	52 550	0.00%	300	52 550	0.00%	151	52 550	0.00%	16	52 550	0.00%	50
b) Emphasize stability	31	chile_1_b	8 160	0.00%	300	8 160	0.00%	122	8 160	0.00%	12	8 160	0.00%	20
	32	chile_2_b	16 740	0.00%	300	16 740	0.00%	140	16 740	0.00%	13	16 740	0.00%	27
	33	chile_3_b	26 520	0.68%	300	26 520	0.68%	173	26 520	0.68%	14	26 340	0.00%	47
	34	chile_4_b	8 400	0.00%	300	8 400	0.00%	127	8 400	0.00%	11	8 400	0.00%	22
	35	chile_5_b	19 800	0.00%	300	19 800	0.00%	143	19 800	0.00%	14	19 800	0.00%	28
	36	chile_6_b	32 580	0.56%	300	32 580	0.56%	182	32 580	0.56%	19	32 400	0.00%	53
	37	chile_7_b	10 620	0.00%	300	10 620	0.00%	135	10 620	0.00%	15	10 620	0.00%	23
	38	chile_8_b	27 920	0.00%	300	27 920	0.00%	115	27 920	0.00%	13	27 920	0.00%	31
	39	chile_9_b	48 500	0.37%	300	48 500	0.37%	153	48 500	0.37%	24	48 320	0.00%	66
	40	chile_10_b	12 420	0.00%	300	12 420	0.00%	137	12 420	0.00%	13	12 420	0.00%	28
	41	chile_11_b	31 520	0.00%	300	31 520	0.00%	144	31 520	0.00%	13	31 520	0.00%	33
	42	chile_12_b	53 900	0.34%	300	53 900	0.34%	194	53 900	0.34%	21	53 720	0.00%	68
c) Emphasize minimizing delay	43	chile_1_c	8 100 001	0.00%	300	8 100 001	0.00%	141	8 100 001	0.00%	15	8 100 001	0.00%	20
	44	chile_2_c	15 780 002	0.00%	300	15 780 002	0.00%	144	15 780 002	0.00%	13	15 780 002	0.00%	25
	45	chile_3_c	25 140 002	0.00%	300	25 140 002	0.00%	172	25 140 002	0.00%	16	25 140 002	0.00%	36
	46	chile_4_c	8 220 002	0.00%	300	8 220 002	0.00%	130	8 220 002	0.00%	13	8 220 002	0.00%	23
	47	chile_5_c	18 840 002	0.00%	300	18 840 002	0.00%	173	18 840 002	0.00%	14	18 840 002	0.00%	27
	48	chile_6_c	31 200 002	0.00%	300	31 200 002	0.00%	170	31 200 002	0.00%	21	31 200 002	0.00%	47
	49	chile_7_c	10 440 002	0.00%	300	10 440 002	0.00%	119	10 440 002	0.00%	13	10 440 002	0.00%	23
	50	chile_8_c	26 960 002	0.00%	300	26 960 002	0.00%	151	26 960 002	0.00%	20	26 960 002	0.00%	32
	51	chile_9_c	47 120 003	0.00%	300	47 120 003	0.00%	155	47 120 003	0.00%	23	47 120 003	0.00%	50
	52	chile_10_c	12 240 002	0.00%	300	12 240 002	0.00%	132	12 240 002	0.00%	13	12 240 002	0.00%	22
	53	chile_11_c	30 560 002	0.00%	300	30 560 002	0.00%	119	30 560 002	0.00%	15	30 560 002	0.00%	33
	54	chile_12_c	52 520 003	0.00%	300	52 520 003	0.00%	158	52 520 003	0.00%	20	52 520 003	0.00%	45
		Average:		**0.05%**	**300**		**0.05%**	**141**		**0.05%**	**15**		**0.00%**	**33**

GAP Y = (Y -Y)/Y**

Figure 5.4: *Results of SAPI on Network 1 (France) and Network 2 (Chile)*

	Instance		Method O1: Right-shift Heuristic			Method O4: I-LS (1 iteration OR 5 min) - Heuristic			Method O3: LS + CPLEX Exact			Method S3: SAPI (1 iteration OR 5 min) - Heuristic			Method S4: SAPI (1 iteration + CPLEX) - Exact		
	N°	Name	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]
a) Balance stability and minimizing delay	1	france_1_a_7	6 103	64.32%	3	3 714	0.00%	88	3 714	0.00%	366	3 714	0.00%	77	3 714	0.00%	213
	2	france_2_a_7	22 405	196.91%	4	7 546	0.00%	64	7 546	0.00%	231	7 546	0.00%	60	7 546	0.00%	263
	3	france_3_a_7	48 469	272.09%	3	13 026	0.00%	255	13 026	0.00%	714	13 026	0.00%	139	13 026	0.00%	372
	4	france_4_a_7	6 819	9.45%	3	6 230	0.00%	89	6 230	0.00%	322	6 230	0.00%	75	6 230	0.00%	194
	5	france_5_a_7	32 692	122.35%	3	14 703	0.00%	300	14 703	0.00%	710	14 703	0.00%	78	14 703	0.00%	285
	6	france_6_a_7	68 329	175.21%	3	24 903	0.30%	300	24 828	0.00%	3 348	26 460	6.57%	300	24 828	0.00%	1 739
b) Emphasize stability	7	france_1_b_7	6 103	52.38%	3	4 005	0.00%	91	4 005	0.00%	501	4 005	0.00%	83	4 005	0.00%	239
	8	france_2_b_7	22 405	196.91%	3	7 546	0.00%	67	7 546	0.00%	210	7 546	0.00%	74	7 546	0.00%	229
	9	france_3_b_7	48 469	270.61%	3	13 078	0.00%	300	13 078	0.00%	829	13 078	0.00%	122	13 078	0.00%	329
	10	france_4_b_7	8 619	32.17%	3	6 521	0.00%	101	6 521	0.00%	488	6 521	0.00%	96	6 521	0.00%	241
	11	france_5_b_7	32 692	122.35%	3	14 703	0.00%	300	14 703	0.00%	820	14 703	0.00%	88	14 703	0.00%	291
	12	france_6_b_7	68 329	164.18%	4	25 865	0.00%	291	25 865	0.00%	824	26 461	2.30%	300	25 865	0.00%	1 870
c) Emphasize minimizing delay	13	france_1_c_7	6 103 000	67.02%	3	3 654 002	0.00%	78	3 654 002	0.00%	366	3 654 002	0.00%	82	3 654 002	0.00%	192
	14	france_2_c_7	22 405 000	196.91%	3	7 546 000	0.00%	68	7 546 000	0.00%	248	7 546 000	0.00%	75	7 546 000	0.00%	258
	15	france_3_c_7	48 469 000	273.82%	3	12 966 004	0.00%	267	12 966 004	0.00%	828	12 966 004	0.00%	199	12 966 004	0.00%	413
	16	france_4_c_7	8 619 000	39.69%	3	6 170 002	0.00%	84	6 170 002	0.00%	365	6 170 002	0.00%	92	6 170 002	0.00%	202
	17	france_5_c_7	32 692 000	122.35%	3	14 703 000	0.00%	287	14 703 000	0.00%	682	14 703 000	0.00%	90	14 703 000	0.00%	338
	18	france_6_c_7	68 328 900	176.07%	4	24 835 010	0.34%	300	24 751 006	0.00%	2 123	24 751 006	0.00%	277	24 751 006	0.00%	1 292
	Average:			**141.93%**	**3**		**0.04%**	**185**		**0.00%**	**776**		**0.49%**	**128**		**0.00%**	**498**

	Instance		Method O1: Right-shift Heuristic			Method O4: I-LS (1 iteration OR 5 min) - Heuristic			Method O3: LS + CPLEX Exact			Method S3: SAPI (1 iteration OR 5 min) - Heuristic			Method S4: SAPI (1 iteration + CPLEX) - Exact		
	N°	Name	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]
a) Balance stability and minimizing delay	19	chile_1_a	8 340	2.77%	1	8 115	0.00%	41	8 115	0.00%	51	8 115	0.00%	12	8 115	0.00%	21
	20	chile_2_a	18 300	15.75%	1	15 810	0.00%	14	15 810	0.00%	65	15 810	0.00%	15	15 810	0.00%	24
	21	chile_3_a	31 380	24.67%	1	25 170	0.00%	36	25 170	0.00%	85	25 170	0.00%	19	25 170	0.00%	34
	22	chile_4_a	8 580	4.00%	1	8 250	0.00%	13	8 250	0.00%	58	8 250	0.00%	13	8 250	0.00%	21
	23	chile_5_a	22 680	20.19%	1	18 870	0.00%	14	18 870	0.00%	74	18 870	0.00%	14	18 870	0.00%	27
	24	chile_6_a	47 580	52.35%	1	31 230	0.00%	41	31 230	0.00%	101	31 230	0.00%	19	31 230	0.00%	49
	25	chile_7_a	10 800	3.15%	1	10 470	0.00%	12	10 470	0.00%	59	10 470	0.00%	14	10 470	0.00%	23
	26	chile_8_a	30 800	14.12%	1	26 990	0.00%	14	26 990	0.00%	73	26 990	0.00%	14	26 990	0.00%	27
	27	chile_9_a	63 500	34.68%	1	47 150	0.00%	41	47 150	0.00%	106	47 150	0.00%	16	47 150	0.00%	48
	28	chile_10_a	12 600	2.69%	1	12 270	0.00%	12	12 270	0.00%	63	12 270	0.00%	13	12 270	0.00%	23
	29	chile_11_a	34 400	12.46%	1	30 590	0.00%	15	30 590	0.00%	70	30 590	0.00%	13	30 590	0.00%	28
	30	chile_12_a	68 900	31.11%	1	52 550	0.00%	40	52 550	0.00%	93	52 550	0.00%	16	52 550	0.00%	50
b) Emphasize stability	31	chile_1_b	8 340	2.21%	1	8 160	0.00%	11	8 160	0.00%	51	8 160	0.00%	12	8 160	0.00%	20
	32	chile_2_b	18 300	9.32%	1	16 740	0.00%	13	16 740	0.00%	67	16 740	0.00%	13	16 740	0.00%	27
	33	chile_3_b	31 380	19.13%	1	26 340	0.00%	21	26 340	0.00%	75	26 520	0.68%	14	26 340	0.00%	47
	34	chile_4_b	8 580	2.14%	1	8 400	0.00%	12	8 400	0.00%	50	8 400	0.00%	11	8 400	0.00%	22
	35	chile_5_b	22 680	14.55%	1	19 800	0.00%	13	19 800	0.00%	65	19 800	0.00%	14	19 800	0.00%	28
	36	chile_6_b	47 580	46.85%	1	32 400	0.00%	18	32 400	0.00%	64	32 580	0.56%	19	32 400	0.00%	53
	37	chile_7_b	10 800	1.69%	1	10 620	0.00%	13	10 620	0.00%	51	10 620	0.00%	15	10 620	0.00%	23
	38	chile_8_b	30 800	10.32%	1	27 920	0.00%	13	27 920	0.00%	52	27 920	0.00%	13	27 920	0.00%	31
	39	chile_9_b	63 500	31.42%	1	48 320	0.00%	20	48 320	0.00%	73	48 500	0.37%	24	48 320	0.00%	66
	40	chile_10_b	12 600	1.45%	1	12 420	0.00%	13	12 420	0.00%	53	12 420	0.00%	13	12 420	0.00%	28
	41	chile_11_b	34 400	9.14%	1	31 520	0.00%	14	31 520	0.00%	53	31 520	0.00%	13	31 520	0.00%	33
	42	chile_12_b	68 900	28.26%	1	53 720	0.00%	20	53 720	0.00%	72	53 900	0.34%	21	53 720	0.00%	68
c) Emphasize minimizing delay	43	chile_1_c	8 340 000	2.96%	1	8 100 001	0.00%	11	8 100 001	0.00%	50	8 100 001	0.00%	15	8 100 001	0.00%	20
	44	chile_2_c	18 300 000	15.97%	1	15 780 002	0.00%	14	15 780 002	0.00%	64	15 780 002	0.00%	13	15 780 002	0.00%	25
	45	chile_3_c	31 380 000	24.82%	1	25 140 002	0.00%	39	25 140 002	0.00%	92	25 140 002	0.00%	16	25 140 002	0.00%	36
	46	chile_4_c	8 580 000	4.38%	1	8 220 002	0.00%	12	8 220 002	0.00%	53	8 220 002	0.00%	13	8 220 002	0.00%	23
	47	chile_5_c	22 680 000	20.38%	1	18 840 002	0.00%	15	18 840 002	0.00%	69	18 840 002	0.00%	14	18 840 002	0.00%	27
	48	chile_6_c	47 580 000	52.50%	1	31 200 002	0.00%	45	31 200 002	0.00%	107	31 200 002	0.00%	21	31 200 002	0.00%	47
	49	chile_7_c	10 800 000	3.45%	1	10 440 002	0.00%	13	10 440 002	0.00%	56	10 440 002	0.00%	13	10 440 002	0.00%	23
	50	chile_8_c	30 800 000	14.24%	1	26 960 002	0.00%	17	26 960 002	0.00%	88	26 960 002	0.00%	20	26 960 002	0.00%	32
	51	chile_9_c	63 500 000	34.76%	1	47 120 003	0.00%	43	47 120 003	0.00%	93	47 120 003	0.00%	23	47 120 003	0.00%	50
	52	chile_10_c	12 600 000	2.94%	1	12 240 002	0.00%	13	12 240 002	0.00%	55	12 240 002	0.00%	13	12 240 002	0.00%	22
	53	chile_11_c	34 400 000	12.57%	1	30 560 002	0.00%	16	30 560 002	0.00%	69	30 560 002	0.00%	15	30 560 002	0.00%	33
	54	chile_12_c	68 900 000	31.19%	1	52 520 003	0.00%	42	52 520 003	0.00%	112	52 520 003	0.00%	20	52 520 003	0.00%	45
	Average:			**17.07%**	**1**		**0.00%**	**21**		**0.00%**	**70**		**0.05%**	**15**		**0.00%**	**33**

GAP_Y = (Y -Y)/Y**

Figure 5.5: *Results of SAPI compared to other methods.*

	Instance		Method O4: I-LS Heuristic			Method O3: LS + CPLEX Exact			Method S3: SAPI Heuristic			Method S4: SAPI Exact		
	N°	Name	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]
a) Balance stability and minimizing delay	1	france_1_a_7	3 714	0.00%	78	3 714	0.00%	78	3 714	0.00%	72	3 714	0.00%	72
	2	france_2_a_7	7 546	0.00%	53	7 546	0.00%	53	7 546	0.00%	74	7 546	0.00%	74
	3	france_3_a_7	13 026	0.00%	245	13 026	0.00%	245	13 026	0.00%	122	13 026	0.00%	122
	4	france_4_a_7	6 230	0.00%	81	6 230	0.00%	81	6 230	0.00%	71	6 230	0.00%	71
	5	france_5_a_7	14 703	0.00%	70	14 703	0.00%	70	14 703	0.00%	61	14 703	0.00%	61
	6	france_6_a_7	24 903	0.30%	256	24 828	0.00%	622	26 460	6.57%	78	24 828	0.00%	525
b) Emphasize stability	7	france_1_b_7	4 005	0.00%	55	4 005	0.00%	55	4 005	0.00%	58	4 005	0.00%	58
	8	france_2_b_7	7 546	0.00%	52	7 546	0.00%	52	7 546	0.00%	66	7 546	0.00%	66
	9	france_3_b_7	13 078	0.00%	81	13 078	0.00%	81	13 078	0.00%	82	13 078	0.00%	82
	10	france_4_b_7	6 521	0.00%	58	6 521	0.00%	58	6 521	0.00%	63	6 521	0.00%	63
	11	france_5_b_7	14 703	0.00%	57	14 703	0.00%	57	14 703	0.00%	59	14 703	0.00%	59
	12	france_6_b_7	25 865	0.00%	186	25 865	0.00%	186	26 461	2.30%	210	25 865	0.00%	210
c) Emphasize minimizing delay	13	france_1_c_7	3 654 002	0.00%	55	3 654 002	0.00%	55	3 654 002	0.00%	76	3 654 002	0.00%	76
	14	france_2_c_7	7 546 000	0.00%	54	7 546 000	0.00%	54	7 546 000	0.00%	95	7 546 000	0.00%	95
	15	france_3_c_7	12 966 004	0.00%	107	12 966 004	0.00%	107	12 966 004	0.00%	138	12 966 004	0.00%	138
	16	france_4_c_7	6 170 002	0.00%	85	6 170 002	0.00%	85	6 170 002	0.00%	73	6 170 002	0.00%	73
	17	france_5_c_7	14 703 000	0.00%	67	14 703 000	0.00%	67	14 703 000	0.00%	68	14 703 000	0.00%	68
	18	france_6_c_7	24 835 010	0.34%	261	24 751 006	0.00%	690	24 751 006	0.00%	89	24 751 006	0.00%	523
		Average:		**0.04%**	**105**		**0.00%**	**150**		**0.49%**	**86**		**0.00%**	**135**

	Instance		Method O4: I-LS Heuristic			Method O3: LS + CPLEX Exact			Method S3: SAPI Heuristic			Method S4: SAPI Exact		
	N°	Name	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]
a) Balance stability and minimizing delay	19	chile_1_a	8 115	0.00%	33.62	8 115	0.00%	33.62	8 115	0.00%	11.75	8 115	0.00%	11.75
	20	chile_2_a	15 810	0.00%	13.72	15 810	0.00%	13.72	15 810	0.00%	11.63	15 810	0.00%	11.63
	21	chile_3_a	25 170	0.00%	35.64	25 170	0.00%	35.64	25 170	0.00%	17.53	25 170	0.00%	17.53
	22	chile_4_a	8 250	0.00%	11.44	8 250	0.00%	11.44	8 250	0.00%	13.24	8 250	0.00%	13.24
	23	chile_5_a	18 870	0.00%	12.60	18 870	0.00%	12.60	18 870	0.00%	13.57	18 870	0.00%	13.57
	24	chile_6_a	31 230	0.00%	41.00	31 230	0.00%	41.00	31 230	0.00%	16.04	31 230	0.00%	16.04
	25	chile_7_a	10 470	0.00%	9.84	10 470	0.00%	9.84	10 470	0.00%	11.93	10 470	0.00%	11.93
	26	chile_8_a	26 990	0.00%	13.86	26 990	0.00%	13.86	26 990	0.00%	14.09	26 990	0.00%	14.09
	27	chile_9_a	47 150	0.00%	35.67	47 150	0.00%	35.67	47 150	0.00%	16.21	47 150	0.00%	16.21
	28	chile_10_a	12 270	0.00%	10.08	12 270	0.00%	10.08	12 270	0.00%	13.46	12 270	0.00%	13.46
	29	chile_11_a	30 590	0.00%	14.85	30 590	0.00%	14.85	30 590	0.00%	13.21	30 590	0.00%	13.21
	30	chile_12_a	52 550	0.00%	35.60	52 550	0.00%	35.60	52 550	0.00%	13.77	52 550	0.00%	13.77
b) Emphasize stability	31	chile_1_b	8 160	0.00%	9.79	8 160	0.00%	9.79	8 160	0.00%	9.66	8 160	0.00%	9.66
	32	chile_2_b	16 740	0.00%	12.87	16 740	0.00%	12.87	16 740	0.00%	11.18	16 740	0.00%	11.18
	33	chile_3_b	26 340	0.00%	21.00	26 340	0.00%	21.00	26 520	0.68%	13.10	26 340	0.00%	35.14
	34	chile_4_b	8 400	0.00%	12.00	8 400	0.00%	12.00	8 400	0.00%	11.32	8 400	0.00%	11.32
	35	chile_5_b	19 800	0.00%	11.18	19 800	0.00%	11.18	19 800	0.00%	11.83	19 800	0.00%	11.83
	36	chile_6_b	32 400	0.00%	16.56	32 400	0.00%	16.56	32 580	0.56%	13.72	32 400	0.00%	29.85
	37	chile_7_b	10 620	0.00%	11.83	10 620	0.00%	11.83	10 620	0.00%	11.74	10 620	0.00%	11.74
	38	chile_8_b	27 920	0.00%	10.79	27 920	0.00%	10.79	27 920	0.00%	11.76	27 920	0.00%	11.76
	39	chile_9_b	48 320	0.00%	18.60	48 320	0.00%	18.60	48 500	0.37%	15.55	48 320	0.00%	28.68
	40	chile_10_b	12 420	0.00%	11.44	12 420	0.00%	11.44	12 420	0.00%	11.98	12 420	0.00%	11.98
	41	chile_11_b	31 520	0.00%	11.90	31 520	0.00%	11.90	31 520	0.00%	13.27	31 520	0.00%	13.27
	42	chile_12_b	53 720	0.00%	16.20	53 720	0.00%	16.20	53 900	0.34%	16.04	53 720	0.00%	33.87
c) Emphasize minimizing delay	43	chile_1_c	8 100 001	0.00%	9.68	8 100 001	0.00%	9.68	8 100 001	0.00%	10.74	8 100 001	0.00%	10.74
	44	chile_2_c	15 780 002	0.00%	11.48	15 780 002	0.00%	11.48	15 780 002	0.00%	11.86	15 780 002	0.00%	11.86
	45	chile_3_c	25 140 002	0.00%	34.71	25 140 002	0.00%	34.71	25 140 002	0.00%	16.32	25 140 002	0.00%	16.32
	46	chile_4_c	8 220 002	0.00%	11.04	8 220 002	0.00%	11.04	8 220 002	0.00%	11.57	8 220 002	0.00%	11.57
	47	chile_5_c	18 840 002	0.00%	13.05	18 840 002	0.00%	13.05	18 840 002	0.00%	11.69	18 840 002	0.00%	11.69
	48	chile_6_c	31 200 002	0.00%	43.20	31 200 002	0.00%	43.20	31 200 002	0.00%	18.55	31 200 002	0.00%	18.55
	49	chile_7_c	10 440 002	0.00%	12.48	10 440 002	0.00%	12.48	10 440 002	0.00%	11.70	10 440 002	0.00%	11.70
	50	chile_8_c	26 960 002	0.00%	16.32	26 960 002	0.00%	16.32	26 960 002	0.00%	11.87	26 960 002	0.00%	11.87
	51	chile_9_c	47 120 003	0.00%	36.98	47 120 003	0.00%	36.98	47 120 003	0.00%	20.69	47 120 003	0.00%	20.69
	52	chile_10_c	12 240 002	0.00%	11.44	12 240 002	0.00%	11.44	12 240 002	0.00%	11.28	12 240 002	0.00%	11.28
	53	chile_11_c	30 560 002	0.00%	15.04	30 560 002	0.00%	15.04	30 560 002	0.00%	13.68	30 560 002	0.00%	13.68
	54	chile_12_c	52 520 003	0.00%	34.02	52 520 003	0.00%	34.02	52 520 003	0.00%	19.81	52 520 003	0.00%	19.81
		Average:		**0.00%**	**19**		**0.00%**	**19**		**0.05%**	**14**		**0.00%**	**15**

Figure 5.6: *CPU time for computing the best solution of SAPI compared to other methods.*

Chapter 6

Constraint Programming Approach

Contents

Abstract of the Chapter

The MIP methods presented in Chapters 4 and 5 are efficient but limited to small and mid-size instances. It is not possible to solve large instances using MIP methods because of the large number of variables and constraints. In this chapter you will discover an alternative way to model the railway rescheduling problem using Constraint Programming (CP). This technique permits to have less variables and constraints compared to the MIP. A cooperative approach mixing MIP and CP is developed to take into account the advantages of both paradigms and solving large instances quickly. The results show that this cooperative procedure can obtain good solutions in a reasonable amount of time for instances larger than those presented in the previous chapters and showed to be much more efficient than a state-of-the-art CP solver.

6.1 Introduction

It is important to remark the difference of the word *programming* in both *Mathematical Programming* and *Constraint Programming*. The first one comes from George Dantzing, inventor of the simplex and considered as one of the fathers of Operations Research. In the beginning, Dantzing use the term *program* to refers to a plan of activities, mainly explained by the applications in solving "programming problems" of the United States Defense Department (Lustig and Puget, 2001). On the other hand, Constraint Programming, also called constraint logic programming, originates in the artificial intelligence literature. For this reason, the term *programming* refers directly to a computer program. Consequently, Constraint Programming is a computer programming technique like object-oriented, structured, and others. Thus, a *constraint program* is not a formulation of a problem, but is rather a computer program (code) that indicates a method for solving a particular problem. In order to utilize Constraint Programming, we introduce some basic notation.

Definition Constraint Satisfaction Problem (CSP) is a triple $P =< X, C, D >$, where X is a set of variables: $x_1, x_2, ..., x_n$, C is a set of constraints: $c_1, c_2, ..., c_m$, and D correspond to the domain of variables: $d_1, d_2, ..., d_n$, *i.e.*, the set of possible values for each variable.

Definition Constraint Optimization Problem (COP) is a couple $Q =< P, f(X) >$, where, P is a CSP and $f(X)$ is an objective function defined in terms of some or all of variables X.

The decision variables could be of any type like integer, boolean, real, or anything else, nevertheless they could be grouped in two types: discrete or continuous. Discrete variables only accept some "disconnected" values for the variables, for example integer numbers. For discrete domains we have two different CSPs (COPs): finite-domains (bounded) and infinite-domains (unbounded). On the other hand, the best-known category of continuous-domain CSPs (COPs) corresponds to linear programming problems, where constraints must be linear inequalities forming a convex region. It is known that linear programming problems can be solved in time polynomial in the number of variables (Hillier and Lieberman, 2001a). Moreover, special solution algorithms exist for linear constraints on integer variables, *i.e.*, Branch-and-Bound. Nevertheless, no algorithm exists for solving general nonlinear constraints on integer variables (Hillier and Lieberman, 2001b). In some cases, we can reduce integer constraint problems to finite-domain problems simply by bounding the values of all the variables.

Each constraint c_i involves some subset of the variables and specifies the allowable combinations of values for that subset. One of the most interesting features in CP is that constraints are logical expressions and not only mathematical expressions. A state of the problem is defined by an assignment of values to some or all of the variables, for example $x_1 = v_1, x_2 = v_2, ..., x_n = v_n$. An assignment that does not violate any constraints is called a consistent or legal assignment and also commonly called a *feasible solution* of the CSP.

We study a CP formulation for the railway rescheduling problem with the aim to deal with larger instances than those treated with MIP formulations. However, this CP formulation plus a standard CP solver are not enough to solve this problem efficiently. This is the reason why an effective algorithm is also proposed in order to find good solutions for real-size instances in a reasonable processing time.

The application of Constraint Programming in rail transportation is not new. The first articles using CP in railway scheduling applications are (Fukumori, 1980; Fukumori and Sano, 1987). They were focused in the construction of a train schedule subject to a set of constraints using an algorithm DFS (depth-first search). Another related work was presented by (Silva de Oliveira, 2001) and (Oliveira and Smith, 2001). The authors solve the single-track scheduling problem as a job-shop scheduling problem proposing a hybrid algorithm based on CP. For the same problem, (Tormos et al., 2006) and (Abril et al., 2006) present a COP. Because of the large number of variables and constraints of the proposed model the authors present diverse approaches to divide the problem into a set of subproblems easier to solve.

More relevant to the topic of this paper, (Chiu et al., 2002) was the first work that incorporated rescheduling. The objective function is similar to the one used in MIP formulations: minimize the modifications of the original schedule and minimize the largest delay of trains. Recently, (Rodriguez, 2007) uses CP for routing and scheduling of trains running through a junction, minimizing conflicts and delays.

The rest of the chapter is organized as follows. Section 6.2 presents a CP formulation for the railway rescheduling problem. Section 6.3 studies the similarities and difference between the two modeling approaches. Section 6.4 is dedicated to explain the proposed algorithm. Finally, computational experiments and conclusions are presented in Section 6.5. The method was implemented over Railway Rescheduling Tool, a software developed in the context of MAGES project (Acuna-Agost and Gueye, 2006). The experiments simulate different kind of incidents over a French railway line using a recovery window of 10 hours. The results show that this methodology can deal efficiently with larger instances than previous chapters and improves significantly the performance of a state-of-the-art CP solver.

6.2 CP Formulation for the RRP

In this section we present a CP model for the railway rescheduling problem. This model supports allocation of tracks and platforms, connection between trains, bidirectional lines, multi-track lines and extra time for accelerating and braking.

In this formulation we identify three main resources: trains, nodes, and tracks. A *train*, associated to index i, corresponds to one or more coupled items of *rolling stock* that have to serve a predefined set of nodes in a given order. *Nodes*, associated to index k, represent physical elements where the train pass and/or stop, *i.e.*, sections, stations and bifurcations (junctions). The last elements are *tracks* that are associated to index t. They correspond to the rails on nodes. Particularly, if a node is a station, tracks are also called *platforms*. In general one node is composed of several parallel tracks. A priori, a train can use any track if the direction of the train and the sense of the track are compatible. Thus, some tracks allow traffic in only one direction while others allow traffic in both.

The components of this model described in this section are: indices, data sets, defined functions, optimization parameters, decision variables, the constraints, and the objective function.

6.2.1 Indices, Data sets and Parameters

Table 6.1 presents the indices and data sets used in this model while Figure 6.2 shows the parameters associated to trains.

Element	Description
i	Index for trains
n	Index for nodes
t	Index for tracks
T	Set of trains
E	Set of nodes ($E = S \cup L$)
S	Set of stations
L	Set of sections and junctions
P	Set of tracks
N_n	Vector of the tracks of node n

Table 6.1: *Indices and data sets used in the CP model.*

6.2.2 Decision Variables

$Delay_{in}$: Delay of train i at node n. $Delay_{in} \geq 0; \forall i \in T, n \in Seq_i$
$Start_{in}$: Arrival time of train i at node n. $Start_{in} \geq 0; \forall i \in T, n \in Seq_i$
$Track_{in}$: Track used by train i at node n. $Track_{in} \in Comp_{in}; \forall i \in T, n \in Seq_i$
UP_{in} : 1, if train i performs an unplanned stop in the node n; 0 otherwise. $UP_{in} \in \{0, 1\}; \forall i \in T, n \in Seq_i$

Element	Description
Seq_i	Vector of sequence of nodes visited by train i
$Stop_i$	Vector of sequence of nodes visited by train i where it has to stop
$Comp_{in}$	Vector of all compatible tracks for train i in the node n
Sen_{in}	Sense of the train i in the node n
$Sep^{min}_{i\hat{i}n}$	Minimal headway between i and $\hat{i}$ at node n
D^{min}_{in}	Minimal duration of train i at node n (planned stop or trip time)
D^{max}_{in}	Maximal duration of train i at node n (planned stop or trip time)
D^{brake}_{in}	Extra time of train i used to brake at node n (only for unplanned stops)
D^{acc}_{in}	Extra time taken train i to accelerate after brake at node n (only for unplanned stops)
$D^{minstop}_{in}$	Minimal stop time of train i for an unplanned stop at node n
$D^{maxstop}_{in}$	Maximal stop time of train i for an unplanned stop at node n
$Start^0_{in}$	Arrival time of train i at node n in the original planning
$Track^0_{in}$	Track used by the train i at node n in the original planning
Inc_{in}	Known delay of train i before to arrive to node n
$Con^{min}_{i\hat{i}n}$	Minimal connection time from train i to train $\hat{i}$ at node n 0 indicates there is no connection between them
$Con^{max}_{i\hat{i}n}$	Maximal connection time from train i to train $\hat{i}$ at node n
CD_{in}	Cost of every time unit of delay for train i at node n
CT_{in}	Cost of change the planned track for train i at node n
CU_{in}	Cost of performing an unplanned stop for train i at node n
$N(i,n)$	Function that returns the next node after n in the sequence of the train i
$P(i,n)$	Function that returns the previous node before n in the sequence of the train i
$L(i)$	Function that returns the last node in the sequence of the train i

Table 6.2: *Parameters used in the CP model associated to train i.*

6.2.3 Constraints

Traveling and Stopping Time

Constraints (6.1) and (6.2) assure the minimal and maximal traveling or stopping time required for train i to pass node n. If the decision is to perform an unplanned stop in that node, it is necessary to add an extra time taken to brake and accelerate given by D_{in}^{brake} and D_{in}^{acc} respectively. It should be noted that it is not necessary to add any extra time for events before or after planned stops, because it is already considered in the minimal running time, D_{in}^{min}. For this reason Constraints (6.3) assure variable UP_{in} to be equal to zero when train i has a planned stop at node n, *i.e.*, events in stations with planned stops cannot be rescheduled as unplanned stops. Finally, Constraints (6.4) are added to guarantee that trains cannot arrive before its original arrival time for all planned stops.

$$\begin{aligned} &Start_{in} + D_{in}^{min} + D_{in}^{minstop} UP_{in} + \\ &D_{in}^{brake} y_{i,N(i,n)} + D_{in}^{acc} y_{i,P(i,n)} \leq Start_{i,N(i,n)} \qquad \forall i \in T; n \in Seq_i \end{aligned} \tag{6.1}$$

$$\begin{aligned} &Start_{in} + D_{in}^{max} + D_{in}^{maxstop} UP_{in} + \\ &D_{in}^{brake} y_{i,N(i,n)} + D_{in}^{acc} UP_{i,P(i,n)} \geq Start_{i,N(i,n)} \qquad \forall i \in T; n \in Seq_i \end{aligned} \tag{6.2}$$

$$UP_{in} = 0 \qquad \forall i \in T, n \in Stop_i \tag{6.3}$$

$$Start_{in} \geq Start_{in}^{0} \qquad \forall i \in T; n \in Stop_i \tag{6.4}$$

Safety Spacing of Trains

To avoid that two train collide it is necessary a minimal headway at the beginning and at the end of the section. Constraints (6.5) and (6.6) guarantee a minimal headway of trains when they use the same track and run in the same direction. There are two possibilities: train i passes before $\hat{i}$ or train i passes after $\hat{i}$, both of them separated by $Sep_{\hat{i}in}^{min}$. These constraints are a clear example of one of the biggest difference between MIP formulations and constraint programming. In this case we present a logic expression composed by three "classical" constraints and linked using logical connectors and "IF THEN" special constraint.

Similarly, headway is also required when two trains use the same track and are running in opposite directions. Constraints (6.7) guarantee that only one of them is using the track. There is a minimal time interval, $Sep_{\hat{i}in}^{min}$ ($Sep_{i\hat{i}n}^{min}$), that has to be assured between train $\hat{i}$ (i) exits and train i ($\hat{i}$) enters node n.

$$(Track_{in} = Track_{\hat{i}n}) \Rightarrow$$
$$|Start_{in} - Start_{\hat{i}n}| \geq Sep^{min}_{i\hat{i}n} \qquad \forall i, \hat{i} \in T; n \in E; Sen_{in} = Sen_{\hat{i}n} \qquad (6.5)$$

$$(Track_{in} = Track_{\hat{i}n}) \wedge (Start_{in} \geq Start_{\hat{i}n}) \Rightarrow$$
$$(Start_{i,N(i,n)} \geq Start_{\hat{i},N(\hat{i},n)} + Sep^{min}_{i\hat{i}n}) \qquad \forall i, \hat{i} \in T; n \in E; Sen_{in} = Sen_{\hat{i}n} \qquad (6.6)$$

$$(Track_{in} = Track_{\hat{i}n}) \vee$$
$$(Start_{in} \geq Start_{\hat{i},N(\hat{i},n)} + Sep^{min}_{i\hat{i}n})$$
$$\Rightarrow (Start_{\hat{i}n} \geq Start_{i,N(i,n)} + Sep^{min}_{\hat{i}in}) \qquad \forall i, \hat{i} \in T; n \in E; Sen_{in} \neq Sen_{\hat{i}n} \qquad (6.7)$$

Delay and Incidents

The lateness of each event is calculated as the difference between the arrival time of the solution and the original arrival time. This condition is assured by set of constraints (6.8). Additionally, constrains (6.9) inject the incidents into the model. These constraints force new starting (ending) times to be greater or equal to the time of the first indication of the incident.

$$Delay_{in} = Start_{in} - Start^{0}_{in} \qquad \forall i \in T; n \in Seq_i \qquad (6.8)$$
$$Start_{in} \geq Start^{0}_{in} + Inc_{in} \qquad \forall i \in T; n \in Seq_i; Inc_{in} \neq 0 \qquad (6.9)$$

Connections

When connections of trains are required, they are assured by constraints (6.10) and (6.11). Thus, $Con^{min}_{i\hat{i}n}$ and $Con^{max}_{i\hat{i}n}$ are the minimal and maximal time for connections between trains i and $\hat{i}$ at node n.

$$Start_{\hat{i},N(i,n)} \geq Start_{in} + Con^{min}_{i\hat{i}n} \qquad \forall i, \hat{i} \in T; n \in Seq_i \cap Seq_{\hat{i}}; Con^{min}_{i\hat{i}n} \neq 0 \qquad (6.10)$$
$$Start_{\hat{i},N(i,n)} \leq Start_{in} + Con^{max}_{i\hat{i}n} \qquad \forall i, \hat{i} \in T; n \in Seq_i \cap Seq_{\hat{i}}; Con^{max}_{i\hat{i}n} \neq 0 \qquad (6.11)$$

6.2.4 Objective Function

Objective function (6.12) corresponds to the total rescheduling cost. This cost is equal to the sum of the cost of delays for all planned stops, plus the total cost of changing

tracks/platforms, plus the total cost of unplanned stops. These costs are calculated by constraints (6.14), (6.15) and (6.16) respectively.

$$\text{Minimize} : TotalCost \tag{6.12}$$

where:

$$TotalCost = CostDelay + CostCTracks + CostUPStops \tag{6.13}$$

$$CostDelay = \sum_{\substack{i \in T \\ n \in Stop_i}} CD_{in} Delay_{in} \tag{6.14}$$

$$CostCTracks = \sum_{\substack{i \in T \\ n \in Seq_i \\ Track_{in} \neq Track^0_{in}}} CT_{in} \tag{6.15}$$

$$CostUPStops = \sum_{\substack{i \in T \\ n \in Seq_i}} CU_{in} UP_{in} \tag{6.16}$$

6.3 Similarities and Differences: MIP vs. CP

Both formulations model the same problem and have the same objective function. Nevertheless, the definition of variables and constraints differ significantly because of the nature of both methodologies. Some aspects can be directly associated, thus Table 6.3 presents the equivalences between CP and MIP formulations. It should be noted that the equivalence is valid only if event k, used in the MIP formulation, is the same event of train i at node n, used in the CP formulation.

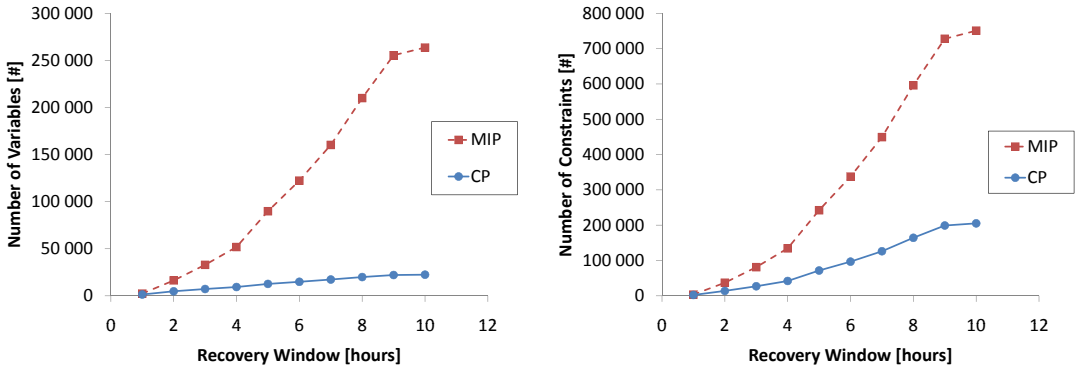
Element	MIP	CP
Original Track	$(q_{kt} = q^0_{kt} = 1)$	$(Track_{in} = Track^0_{in})$
Decision variable - start time (arrival)	x^{begin}_k	$Start_{in}$
Decision variable - delay	z_k	$Delay_{in}$
Decision variable - unplanned stop	y_k	UP_{in}

Table 6.3: *Similarities between CP and MIP models.*

The main difference between both formulations is the way they model the order of trains and the use of tracks. On the one hand, the MIP formulation includes binary variables that are created as the combinatory of all events that could share a track. These variables are $\gamma_{k\hat{k}} \in \{0,1\}$ and $\lambda_{k\hat{k}} \in \{0,1\}$. Note that the number of these variables increases drastically with the number of trains and stations. The constraints to assure the headway of trains are also a complication for the MIP formulation. These constraints include a large constant, denoted by M, that may complicate the numerical aspect of the algorithm. More details about the MIP can be found in Chapter 4.

On the other hand, the CP formulation is clearly more natural to model this problem. First, the decision variables define directly the arrival time of trains and the assigned track. The headway and other aspects are modeled by the constraints. The advantage is that the constraints are not required to be linear, *e.g.*, constraint (6.5) is a logic "IF THEN" constraint and includes an absolute value function.

Because of the modeling differences, CP formulation requires much fewer variables and constraints than MIP formulation. Figures 6.1(a) and 6.1(b) illustrate this fact using an instance of the test. All instances have the same behavior because they differ just in the nature of the incidents and the coefficients of the objective value. These graphs plot the number of variables and constraints when the recovery window is increased. The longer the recovery window, the more number of trains and arrival/departure events. As it was expected, the number of variables for the MIP increases drastically. This is a real memory issue, in fact our test using the MIP on the instance showed in Figures 6.1(a) and 6.1(b) where limited to a recovery window of 7 hours because of lack of RAM memory.



(a) Number of variables

(b) Number of constraints

Figure 6.1: *Differences of the size of the models size: MIP vs. CP*

6.4 A Cooperative CP/MIP Approach

The objective of this algorithm is to solve large instances of the railway rescheduling problem. For this reason, we propose to use the CP model because it requires less memory than the MIP. Additionally, we want to take advantage of some of the efficient MIP-based methods in order to improve performance of a standard CP solver. This is carried out through two main strategies: domain filtering and variable/values ordering.

Considering any CP formulation, the problem can be solved via *backtracking*. The base method attempts to obtain solutions by choosing a non-fixed variable and a value of the domain for that variable. The chosen variable is then fixed to the chosen value triggering constraint propagation. *Constraint propagation* and *filtering* decrease the do-

mains of variables during this process. The method continues with another non-fixed variable, if one exists, and repeated until all decision variables have only one possible value, *i.e.*, they are all fixed. However, if a fixing fails because it cannot lead to a solution or the solution is "worse" than the current incumbent solution, the method backtracks and chooses another value for the variable. When all values for the current variable have been tried, the method backtracks to a previously assigned variable and reassigns it.

This process can be improved in three generic ways: **adapted search strategy**, **domain filtering**, and **variable/values ordering** (Grandoni and Italiano, 2006). First, there are different strategies to backtrack in case of a fails such as *backjumping*, *backchecking* and *backmarking*. The effectiveness of a particular search strategy depends on the nature of the problem. Secondly, domain filtering has the objective to reduce the domain of variables in order to speed up the search. It is possible to use the knowledge of the problem to design an improved domain reduction and constraint propagation procedures. Finally, the order of variables and their values considering in the search affects the performance of the system considerably. An ordered list of the decision variables is passed to the search algorithm and the first unassigned variable in the list will be chosen next.

The CP/MIP approach uses the two last strategies, domain filtering and variable and values ordering. The main aspect in this algorithm is the use of an auxiliary MIP model to solve a subproblem by limiting the recovery window. The solution of this subproblem is employed to reduce the domain of variables and to define an appropriate variable/value ordering sets to lead the search algorithm.

Domain Filtering A MIP model is created for a subproblem defined as the railway rescheduling problem for the events (arrival/departure of trains) before a time parameter denoted by $EndTimeMIP$. Figure 6.2 illustrates the relation between the complete CP model and the MIP subproblem. This subproblem is then solved by any MIP-based method (see Chapter 4). It should be noted that the performance and the quality of this method depends on the solution of this subproblem.

Variable and values ordering Using the result obtained by RS, it is possible to identify the events in the subproblem which are not affected by the incidents. The decision variables modeling delays, *i.e.*, $Delay_{in}$, for non-affected events are added to a dynamic array denoted as $List$. These variables are very important because once these variables are fixed, it is easier to extend the partial solution to the remaining variables because of constraint propagation of constraints 6.8. In addition, for the selected events, it is probable that $Delay_{in} = 0$. Thus, when instantiating variables, the first value chosen is the smallest possible value for the current variable.

All steps of this procedure are detailed in Algorithm 4. Several requirements (inputs) are needed. The first one, CP, corresponds to an instance of the CP model described in Sec. 6.2. The next parameter is $EndTimeMIP$ that limits the size of the MIP subproblem. The subproblem contains all events comprised before $EndTimeMIP$.

It seems to be reasonable that non-affected trains in the solution of RS will not be

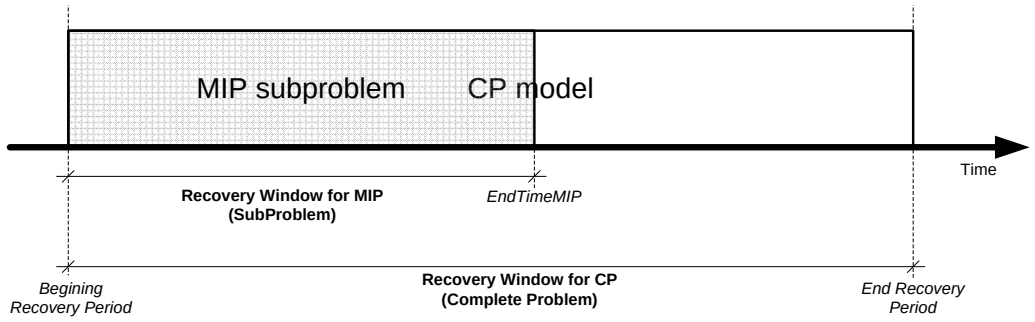


Figure 6.2: *Relation between the complete CP model and the MIP subproblem.*

delayed in the optimal solution of the complete problem. Therefore, the search can be improved if we force an extra domain reduction of all these variables. Let *Tolerance* be a vector composed by the elements $tolerance_{in}$ with the maximal allowed time to increase the delay obtained in the solution of the subproblem (MIP model). In particular, if we have $tolerance_{in} = 0$ for $Event_{in}$, the event will be fixed using the solution of the subproblem. It should be noted that this approach is heuristic because of parameter *Tolerance*, *i.e.*, some feasible solutions, including optimal solutions, could be discarded.

The two final parameters are used as stop criteria; *TimeLimit* is the maximal processing time for the algorithm and *SolLimit* is the maximal number of feasible solutions that has been reached.

The key aspect of this algorithm is to create and solve the subproblem. Line 1 creates a MIP for the railway rescheduling problem limited to events before *EndTimeMIP* (see Figure 6.2). It should be noted that solving this subproblem can be as hard as the original problem. For this reason we propose to use one of the method described in the previous chapters, for example SAPI that shows to obtain a good balance between quality of the solution and speed. In the solution of the subproblem some of trains will be not affected because of the robustness of the original plan. This is especially useful to identify affected (and non-affected trains) and use this information to improve the performance of a CP solver. Line 2 solves the subproblem by any of the MIP-based methods described in the previous chapters and returns its solution, values of decision variables, to the vector X^{MIP}. In order to respect the CPU time imposed with *TimeLimit*, the solution of the subproblem is limited to *TimeLimit*/2. Line 3 initializes the list (*List*) of variables that will be used to define the search strategy.

The "for loops" between Lines 4 to 23 read the solution of the subproblem returned by the MIP method and evaluate which arrival events are not affected with the incidents. The first condition is to check if the event is comprised also in the subproblem, that is $Start^0_{in} \leq EndTimeMIP$. For all these events the variables $Start_{in}$ and $Delay_{in}$ are bounded to the value of the subproblem solution xb^{RS}_{in} plus a tolerance parameter *Tolerance* (Lines 9-10).

Lines 12 to 14 are executed only when the tolerance for this event event (combination of i and n) is zero, *i.e.*, $tolerance_{in} = 0$ (Line 13). In that case, we keep the same

Algorithm 4 CP-based algorithm

Require: CP, $EndTimeMIP$, $Tolerance$, $TimeLimit$, $SolLimit$

$MIP \leftarrow$ CreateMIPModel($EndTimeMIP$)
$X^{MIP} \leftarrow$ MIPMethod(MIP, $TimeLimit/2$)
$List \leftarrow \emptyset$
for all $i \in T$ **do**
  for all $n \in Seq_i$ **do**
    if ($Start^0_{in} \leq EndTimeMIP$ **then**
      for all $k \in Events$ (events in MIP) **do**
        if (Event k (MIP) is the same event of train i at node n (CP)) **then**
          CP.AddConstraint ($Start_{in} \leq x_k^{begin} + tolerance_{in}$)
          CP.AddConstraint ($Delay_{in} \leq x_k^{begin} - Start^0_{in} + tolerance_{in}$)
          if ($tolerance_{in} = 0$) **then**
            CP.AddConstraint ($Track_{in} =$ track of event k in X^{MIP})
            CP.AddConstraint ($UP_{in} = y_k$)
            Add variable $Delay_{in}$ to $List$
          end if
          if ($x_k^{begin} = Start^0_{in}$) **then**
            Add variable $Delay_{in}$ to $List$
          end if
        end if
      end for
    end if
  end for
end for
$SearchStrategy \leftarrow$ Strategy($List$, LargestIndex($List$), SmallestValue)
$ElapsedTime \leftarrow$ EvaluateElapsedTime()
$TimeLimit \leftarrow TimeLimit - ElapsedTime$
$X^{CP} \leftarrow$ SolveCP(CP,$SearchStrategy$,$TimeLimit$, $SolLimit$)
return X^{CP}

track and the decision of performing an unplanned stop given by the solution of the subproblem.

Line 17 is executed only when the event (combination of i and n) is not affected in the solution of the subproblem, *i.e.*, $x_k^{begin} = Start_{in}^0$ (Line 16). In that case, variable $Delay_{in}$ is added to *List* to be used in the conception of the search strategy.

Line 24 defines a search strategy ("SearchStrategy"). This search strategy is based on *constructive search*, which builds a solution by fixing decision variables to values. The construction of this strategy is assured by function "Strategy()" which requires three parameters. First, variables belonging to *List* compose a group of key decision variables, $Delay_{in}$ who are not affected by incidents, such that once these variables are fixed, it is easy to extend the partial solution to the remaining variables, e.g, $Start_{in}$ and $CostDelay$. The objective is to force the search to fix the decision variables from *List* before instantiating any other variable in the model. Secondly, the parameter "LargestIndex(*List*)" indicates that variables in *List* are selected with a LIFO (*last in first out*) rule, *i.e.*, later events are fixed first. The last parameter is the strategy to choose values. Thus "SmallestValue" assures that the first value tried is the smallest value in the domain of variables and, if *fails* it continues for the successor and so on. Note that in *List* we include variables of delays for non-affected trains; they will probably have an optimal value of zero.

The last part of the algorithm is to solve the CP model. Lines 25 and 26 calculate the elapsed processing time (*ElapsedTime*) and the remaining time(*TimeLimit*). Let X^{CP} be the vector of the values for the decision variables obtained by solving the problem. The function "SolveCP()" calls a CP solver taking four parameters: CP is the original CP model, *SearchStrategy* is the custom defined search strategy, *TimeLimit* sets a time limit on the time spent in search, and *SolLimit* is the limit of the number of feasible solutions found. In particular we use IBM CP Optimizer detailed in the next section. Finally, Line 28 returns the best solution found.

6.5 Experimental Results

We test this method over three different set of instances. The first two are the same used in the last chapters (see Section 4.5). The last set of instances corresponds to a real French line between the stations of Ruffec and Monts. In order to organize this section we call this last set of instances "Network 3". They use the same French network presented in Chapter 4 but extended to a time horizon of 10 hours, *i.e.*, in the previous chapters the problem instances were limited to 7 hours (time horizon). The original schedule is thus composed of 76 trains (freight, regional express, and high speed trains) passing through 43 stations in a time horizon of 10 hours. This corresponds to 4451 events to be rescheduled. All data were obtained from the research department of SNCF (French National Railway Company) in the context of the MAGES project (Acuna-Agost and Gueye, 2006).

For the incidents of Network 3, we disturbed trains on extreme nodes to affect a large number of trains. Thus, one or two trains are directly implicated in the incident by an initial delay of 10, 20, or 30 minutes. We also consider different combination of unitary cost for the objective function: (a) emphasizing minimization of delay (instance 13-18), (b) emphasizing stability (instances 7-12), and (c) balancing stability and minimizing delay (instances 1-6). Where the coefficients in the objective function strongly penalize the delays, penalize the changes of tracks/platforms/stops, and a balance between them respectively.

It is very important to remark that we tried to solve the instances of Network 3 using the methods presented in the previous chapters. All previous methods are based on the same MIP formulation and this is the reason why we could not load the MIP in memory (i.e., *out of memory error*) because of the large number of variable and constraints. For this reason we compare the solution of the method proposed in this chapter to the other methods using the first two set of instances. The result on the last set of instances are compared to CP Optimizer 2.11, a state-of-the-art CP solver.

This algorithm was coded in Visual Studio 2005 (C♯) using IBM ILOG CPLEX version 11.21 as a black-box solver for the MIP model and IBM ILOG CP Optimizer 2.11 for the CP model. Our system runs on a single processor IBM compatible PC Intel Core 2, 1.66 GHz, and 2 GB of main memory. This algorithm was implemented as a component of Railway Rescheduling Tool (RRT) developed in the context of project MAGES (Acuna-Agost and Gueye, 2006). More details about this software are presented in Chapter 10.

For all methods we report both: the quality of the solution using the value of objective function (column "Cost [$]") and the processing time needed to calculate this solution (column "CPU [s]"), that is the global CPU time of the corresponding method in seconds. In some methods we also include the CPU time required for computing the best solution (column "CPU to best [s]") in seconds. Additionally, we include a GAP that is utilized to compare the results of the solution methods. Note that for Network 1 and Network 2, this GAP is calculated in relation to the optimal solution, while for Network 3 we use the best solution known that does not necessarily correspond to the

optimal one.

We tested the following method:

- CP-based algorithm (CP-based algorithm 4): We use the parameters $TimeLimit =$ 5 minutes, $tolerance_{in} = 0; \forall i \in T, n \in Seq_i$ and $SolLimit = 1000000$. It uses Method S3 (SAPI Heuristic) to solve the subproblem that is created by limiting the time horizon to the half of the time horizon of the current instance (parameter $EndTimeMIP$).

The CP-based algorithm is compared to the following methods for Network 3:

- CP Optimizer (First Solution), IBM ILOG CP Optimizer 2.11 with default parameters. It stops when it found the first feasible solution, i.e., it solves the CSP associated to this problem.
- CP Optimizer (5 minutes), IBM ILOG CP Optimizer 2.11 with default parameters. It stops after 5 minutes of search. It could stop before the CPU time limit only if the optimal solution is reached.
- CP Optimizer (30 minutes), IBM ILOG CP Optimizer 2.11 with default parameters. It stops after 30 minutes of search. It could stop before the CPU time limit only if the optimal solution is reached.

It is compared to the following methods for Network 1 and Network 2:

- Method O1 (Right-shift Heuristic), *Right-shift rescheduling*. (Section 4.3.1)
- Method O4 (I-LS Heuristic. Stop criteria: 1 iteration), *MIP-based local search method* using right-shift as initial solution. It limits the search space around the original non-disrupted schedule with local-branching-type cuts added to the model. (Section 4.3.3.
- Method S3 (SAPI Heuristic, stop criteria: 1 iteration (or) 5 minutes): *SAPI* algorithm limited to 5 minutes of running time or one complete iteration. The limit of the processing time responds to real requirements of railway companies. Additionally, the algorithm performs only one iteration, i.e., if the first iteration finishes before 5 minutes, the algorithm stops. (Chapter 5)
- Method S4 (SAPI Exact, 1 iteration of SAPI + CPLEX) This method uses the output of Method S3 as an initial solution for IBM ILOG CPLEX removing any additional cut and letting all variable be free. As a consequence, the feasible region corresponds to the original search space and the optimal solution of this method coincides with the optimal solution of the original problem. This method is described in Section 4.4 using SAPI as the heuristic method. This methods has been showed to be the best exact method presented in this Thesis considering the both dimensions: quality of the solution and the speed. (Chapter 5)

Figure 6.3 show the results obtained by IBM ILOG CP Optimizer 2.11 with default parameters. We include these results as a reference to be compared with those obtained by Algorithm 4 on the instances of Network 3. Three versions are evaluated changing

	N°	Name	Best [$]	CP Optimizer 2.11 (default) (Stop Criteria: First Solution)			CP Optimizer 2.11 (default) (Stop Criteria: 5 minutes)			CP Optimizer 2.11 (default) (Stop Criteria: 30 minutes)		
				Cost [$]	GAP [%]	CPU [s]	Cost [$]	GAP [%]	CPU [s]	Cost [$]	GAP [%]	CPU [s]
a) Balance stability and minimizing delay	1	france_1_a_10	3 761	19 699	423.77%	70	19 699	423.77%	300	3 890	3.43%	1 800
	2	france_2_a_10	7 593	34 109	349.22%	72	22 062	190.56%	300	8 565	12.80%	1 800
	3	france_3_a_10	13 073	58 251	345.58%	72	58 248	345.56%	300	13 866	6.07%	1 800
	4	france_4_a_10	6 277	24 427	289.15%	73	10 943	74.33%	300	6 661	6.12%	1 800
	5	france_5_a_10	16 296	46 768	186.99%	71	35 447	117.52%	300	18 712	14.83%	1 800
	6	france_6_a_10	29 018	83 107	186.40%	72	82 977	185.95%	300	33 174	14.32%	1 800
b) Emphasize stability	7	france_1_b_10	4 052	19 699	386.15%	71	8 172	101.68%	300	6 080	50.05%	1 800
	8	france_2_b_10	7 593	34 109	349.22%	70	21 212	179.36%	300	9 550	25.77%	1 800
	9	france_3_b_10	13 125	57 927	341.35%	114	57 927	341.35%	300	15 085	14.93%	1 800
	10	france_4_b_10	6 568	23 144	252.38%	184	23 144	252.38%	300	8 562	30.36%	1 800
	11	france_5_b_10	16 296	45 472	179.04%	116	35 447	117.52%	300	20 775	27.49%	1 800
	12	france_6_b_10	29 018	85 064	193.14%	121	85 064	193.14%	300	32 296	11.30%	1 800
c) Emphasize minimizing delay	13	france_1_c_10	3 701 002	20 049 000	441.72%	74	9 824 240	165.45%	300	9 634 145	160.31%	1 800
	14	france_2_c_10	7 593 000	33 785 000	344.95%	74	22 062 000	190.56%	300	9 840 028	29.59%	1 800
	15	france_3_c_10	13 013 004	58 251 000	347.64%	72	53 494 895	311.09%	300	13 259 034	1.89%	1 800
	16	france_4_c_10	6 217 002	25 418 000	308.85%	72	13 430 000	116.02%	300	6 419 012	3.25%	1 800
	17	france_5_c_10	16 296 000	47 572 000	191.92%	75	35 771 000	119.51%	300	18 893 005	15.94%	1 800
	18	france_6_c_10	26 508 000	84 740 000	219.68%	74	84 560 069	219.00%	300	26 799 730	1.10%	1 800
			Average:		296.51%	86		202.49%	300		23.86%	1 800

Figure 6.3: *Results of CP Optimizer 2.11 with defaults parameters (Network 3).*

the ending criteria. The first version stops when the first feasible solution is found. The second (the third) stops either when the optimum is found limiting to 5 minutes (30 minutes) of processing time. On the other hand, the results of Algorithm 4 are presented in Figure 6.4. Note that all the best solution for this set of instances are found with this method.

The results show that Algorithm 4 is better than using a standard CP solver in both dimensions. First, this method is able to find better solutions than CP optimizer, indeed limiting the processing time to 5 minutes. The best solutions obtained with CP Optimizer 2.11 have an average GAP of 23.86% after 30 minutes of CPU time in our machine. In contrast, the proposed method obtains an averga GAP of 0%; that is, it found the best solution for all instances in Network 3. The second conclusion is that Algorithm 4 is faster obtaining good solutions. This fact is shown by the average CPU time: 131 versus 1800 seconds if we compare Algorithm 4 vs CP Optimizer 2.11 (30 minutes).

After analyzing the results for all instances, we realized that once Algorithm 4 finishes the first phase using the MIP model the CP solver finds a very good solution quickly. In fact, in most of the cases, it immediately ends after finding the first feasible solution, because it is optimal in relation to the remained CP model. It is possible to appreciate this fact comparing the quality of the first solutions calculated by Algorithm 4 and CO Optimizer 2.11. Figure 6.5 illustrates this behavior. The graph plots the evolution of the objective function for both solution methods (Algorithm 4 and CP Optimizer 2.11) on instance "france-1-a-10". It should be noted that Algorithm 4 can find quickly a good solution (in this case an optimal solution) whereas CP Optimizer 2.11 spends a

	N°	Name	Best [$]	CP-based algorithm (subproblem: SAPI [5 h])			
				Cost [$]	GAP [%]	CPU [s]	CPU to best [s]
a) Balance stability and minimizing delay	1	france_1_a_10	3 761	3 761	0.0000%	75	74
	2	france_2_a_10	7 593	7 593	0.0000%	72	70
	3	france_3_a_10	13 073	13 073	0.0000%	109	107
	4	france_4_a_10	6 277	6 277	0.0000%	81	80
	5	france_5_a_10	16 296	16 296	0.0000%	223	221
	6	france_6_a_10	29 018	29 018	0.0000%	225	223
b) Emphasize stability	7	france_1_b_10	4 052	4 052	0.0000%	76	75
	8	france_2_b_10	7 593	7 593	0.0000%	70	70
	9	france_3_b_10	13 125	13 125	0.0000%	99	98
	10	france_4_b_10	6 568	6 568	0.0000%	78	78
	11	france_5_b_10	16 296	16 296	0.0000%	189	188
	12	france_6_b_10	29 018	29 018	0.0000%	232	231
c) Emphasize minimizing delay	13	france_1_c_10	3 701 002	3 701 002	0.0000%	84	83
	14	france_2_c_10	7 593 000	7 593 000	0.0000%	75	74
	15	france_3_c_10	13 013 004	13 013 004	0.0000%	119	118
	16	france_4_c_10	6 217 002	6 217 002	0.0000%	82	81
	17	france_5_c_10	16 296 000	16 296 000	0.0000%	236	235
	18	france_6_c_10	26 508 000	26 508 000	0.0000%	237	237
			Average:		0.0000%	131	130

Figure 6.4: *Results of Algorithm 4 (Network 3).*

considerable amount of time trying to improve the current solution. As a consequence, for this example given a limitation of the processing time, for example 10 minutes, the solution obtained by Algorithm 4 is much better than those returned by CP Optimizer 2.11 (default parameters).

We also present the results in Network 1 and Network 2 that were used in Chapter 4 and Chapter 5 with the aim to compare this new approach to those described previously. Figure 6.5 shows the results of Algorithm 4 compared to Method O1, O4, S3, and S4. It should be noted that this CP-based algorithm is faster in average than O4, S3, and S4; however, the quality of the solution is worse. It is also possible to observe that the variability of the solutions obtained by Algorithm 4 is higher than other methods. As a consequence, the previous methods show to be more stable for these kind of instances. Additionally, we compare the performance of the methods in terms of the speed to calculate the best solution. Figure 6.6 presents the CPU time required for computing the best solution (Network 1 and 2) comparing Algorithm 4 to some solution methods described in the previous chapters. Concerning the heuristic methods, Method O3 is the approach with the best solutions in average; while Method S3 (SAPI Heuristic) is the faster to find solutions with reasonable quality of solution (GAP less than 1%). A good characteristic of Algorithm 4 is that it is able to identify quickly that the current solution is good; actually, the algorithm stops immediately when it finds the best solution.

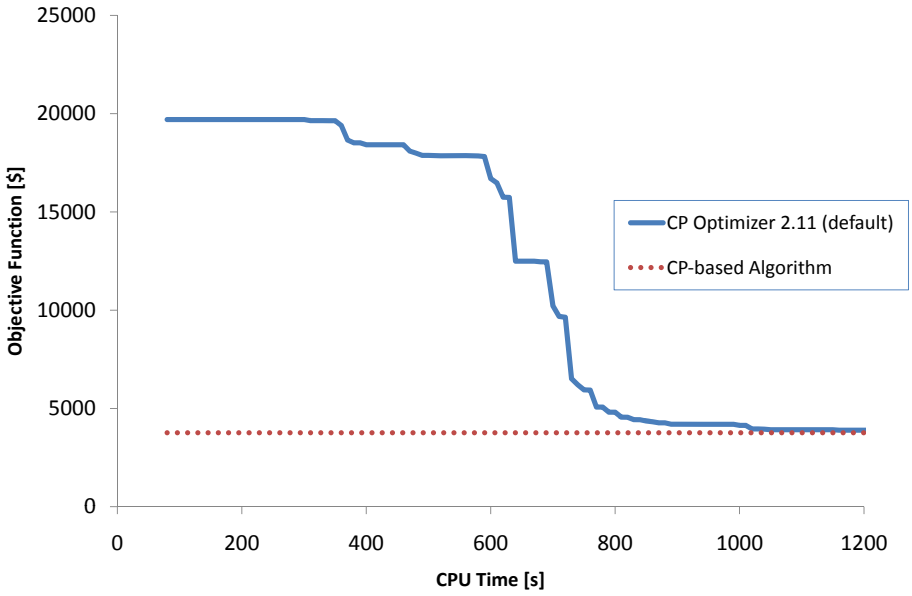


Figure 6.5: *Comparative analysis for instance* `france-1-a-10`*. CP Optimizer 2.11 versus Algorithm 4*

6.6 Concluding Remarks

Two alternatives formulation for rescheduling trains after incidents have been compared: CP and MIP. We proposed a cooperative algorithm that take advantage of both, that is, use CP to model large problems without memory problems and a MIP-based method to improve the performance of the search. More specifically, it is improved in two ways: domain filtering and an appropriate variable/values ordering. These two strategies are considering solving a MIP subproblem with one of the MIP-based methods presented in the previous chapters.

The solution of the subproblem is then used to fix decision variables. Non-affected events are imposed to be similar to the original schedule reducing the domain of variables that model delays. Additionally, depending on some parameters of the method these events are allowed neither to change their planned track nor to perform unplanned stops.

The identification of non-affected events is also utilized to forces the search to fix the decision variables associated to non-affected trains before any other variable in the model. Then, it is coded that the solver selects the smallest value in the domain of the selected variable. These strategies guarantees that the first solutions found will be not composed of trains extremely delayed.

We tested this method using three different set of instances: the two ones presented in the previous chapters (Network 1 - France and Network 2 - Chile) and a new larger one (Network 3 - France).

Concerning the first two networks, the results have shown that previous methods

based on the MIP formulation seem to be more appropriate for little and mid-size instances.

The experimental results over large instances (Network 3) showed that the proposed cooperative CP/MIP algorithm is satisfactory to increase significantly the performance of a state-of-the-art CP solver like CP Optimizer 2.11. It is very important to remark also that all instances of Network 3 were solved without memory errors with a CPU time of 130 seconds in average. In contrast, we tried to solve the same instances using the MIP-based methods using the same machine and it was not possible to load any instance in memory.

	Instance		Method O1: Right-shift Heuristic			Method O4: I-LS Heuristic			Method S3: SAPI Heuristic			Method S4: SAPI Exact			CP-based algorithm (subproblem: SAPI [3.5 h])			
	N°	Name	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [s]	CPU to best [s]
a) Balance stability and minimizing delay	1	france_1_a_7	6 103	64.32%	3	3 714	0.00%	88	3 714	0.00%	77	3 714	0.00%	213	3 714	0.00%	44	44
	2	france_2_a_7	22 405	196.91%	4	7 546	0.00%	64	7 546	0.00%	60	7 546	0.00%	263	7 546	0.00%	44	44
	3	france_3_a_7	48 469	272.09%	3	13 026	0.00%	255	13 026	0.00%	139	13 026	0.00%	372	13 071	0.35%	190	190
	4	france_4_a_7	6 819	9.45%	3	6 230	0.00%	89	6 230	0.00%	75	6 230	0.00%	194	6 230	0.00%	45	45
	5	france_5_a_7	32 692	122.35%	3	14 703	0.00%	300	14 703	0.00%	78	14 703	0.00%	285	15 438	5.00%	211	211
	6	france_6_a_7	68 329	175.21%	3	24 903	0.30%	300	26 460	6.57%	300	24 828	0.00%	1 739	24 828	0.00%	91	91
b) Emphasize stability	7	france_1_b_7	6 103	52.38%	3	4 005	0.00%	91	4 005	0.00%	83	4 005	0.00%	239	4 005	0.00%	45	45
	8	france_2_b_7	22 405	196.91%	3	7 546	0.00%	67	7 546	0.00%	74	7 546	0.00%	229	7 546	0.00%	45	45
	9	france_3_b_7	48 469	270.61%	3	13 078	0.00%	300	13 078	0.00%	122	13 078	0.00%	329	13 078	0.00%	57	57
	10	france_4_b_7	8 619	32.17%	3	6 521	0.00%	101	6 521	0.00%	96	6 521	0.00%	241	6 521	0.00%	47	47
	11	france_5_b_7	32 692	122.35%	3	14 703	0.00%	300	14 703	0.00%	88	14 703	0.00%	291	14 703	0.00%	54	54
	12	france_6_b_7	68 329	164.18%	4	25 865	0.00%	291	26 461	2.30%	300	25 865	0.00%	1 870	28 971	12.01%	224	224
c) Emphasize minimizing delay	13	france_1_c_7	6 103 000	67.02%	3	3 654 002	0.00%	78	3 654 002	0.00%	82	3 654 002	0.00%	192	3 654 002	0.00%	45	45
	14	france_2_c_7	22 405 000	196.91%	3	7 546 000	0.00%	68	7 546 000	0.00%	75	7 546 000	0.00%	258	7 546 000	0.00%	46	46
	15	france_3_c_7	48 469 000	273.82%	3	12 966 004	0.00%	267	12 966 004	0.00%	199	12 966 004	0.00%	413	12 966 004	0.00%	70	70
	16	france_4_c_7	8 619 000	39.69%	3	6 170 002	0.00%	84	6 170 002	0.00%	92	6 170 002	0.00%	202	6 170 002	0.00%	45	45
	17	france_5_c_7	32 692 000	122.35%	3	14 703 000	0.00%	287	14 703 000	0.00%	90	14 703 000	0.00%	338	14 703 000	0.00%	46	46
	18	france_6_c_7	68 328 900	176.07%	4	24 835 010	0.34%	300	24 751 006	0.00%	277	24 751 006	0.00%	1 292	26 433 005	6.80%	213	213
	Average:			**141.93%**	**3**		**0.04%**	**185**		**0.49%**	**128**		**0.00%**	**498**		**1.34%**	**87**	**87**

	Instance		Method O1: Right-shift Heuristic			Method O4: I-LS Heuristic			Method S3: SAPI Heuristic			Method S4: SAPI Exact			CP-based algorithm (subproblem: SAPI [12 h]) - Heuristic			
	N°	Name	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	CPU to best [s]
a) Balance stability and minimizing delay	19	chile_1_a	8 340	2.77%	1	8 115	0.00%	41	8 115	0.00%	12	8 115	0.00%	21	8 130	0.18%	24	24
	20	chile_2_a	18 300	15.75%	1	15 810	0.00%	14	15 810	0.00%	15	15 810	0.00%	24	15 825	0.09%	24	24
	21	chile_3_a	31 380	24.67%	1	25 170	0.00%	36	25 170	0.00%	19	25 170	0.00%	34	25 200	0.12%	32	32
	22	chile_4_a	8 580	4.00%	1	8 250	0.00%	13	8 250	0.00%	13	8 250	0.00%	21	8 265	0.18%	23	23
	23	chile_5_a	22 680	20.19%	1	18 870	0.00%	14	18 870	0.00%	14	18 870	0.00%	27	18 885	0.08%	24	24
	24	chile_6_a	47 580	52.35%	1	31 230	0.00%	41	31 230	0.00%	19	31 230	0.00%	49	31 275	0.14%	28	28
	25	chile_7_a	10 800	3.15%	1	10 470	0.00%	12	10 470	0.00%	14	10 470	0.00%	23	10 485	0.14%	23	23
	26	chile_8_a	30 800	14.12%	1	26 990	0.00%	14	26 990	0.00%	14	26 990	0.00%	27	27 005	0.06%	24	24
	27	chile_9_a	63 500	34.68%	1	47 150	0.00%	41	47 150	0.00%	16	47 150	0.00%	48	47 195	0.10%	32	32
	28	chile_10_a	12 600	2.69%	1	12 270	0.00%	12	12 270	0.00%	13	12 270	0.00%	23	12 285	0.12%	23	23
	29	chile_11_a	34 400	12.46%	1	30 590	0.00%	15	30 590	0.00%	13	30 590	0.00%	28	30 605	0.05%	24	24
	30	chile_12_a	68 900	31.11%	1	52 550	0.00%	40	52 550	0.00%	16	52 550	0.00%	50	52 595	0.09%	31	31
b) Emphasize stability	31	chile_1_b	8 340	2.21%	1	8 160	0.00%	11	8 160	0.00%	12	8 160	0.00%	20	8 400	2.94%	23	23
	32	chile_2_b	18 300	9.32%	1	16 740	0.00%	13	16 740	0.00%	13	16 740	0.00%	27	16 980	1.43%	24	24
	33	chile_3_b	31 380	19.13%	1	26 340	0.00%	21	26 520	0.68%	14	26 340	0.00%	47	26 760	1.59%	29	29
	34	chile_4_b	8 580	2.14%	1	8 400	0.00%	12	8 400	0.00%	11	8 400	0.00%	22	8 640	2.86%	23	23
	35	chile_5_b	22 680	14.55%	1	19 800	0.00%	13	19 800	0.00%	14	19 800	0.00%	28	20 040	1.21%	24	24
	36	chile_6_b	47 580	46.85%	1	32 400	0.00%	18	32 580	0.56%	19	32 400	0.00%	53	32 820	1.30%	31	31
	37	chile_7_b	10 800	1.69%	1	10 620	0.00%	13	10 620	0.00%	15	10 620	0.00%	23	10 860	2.26%	24	24
	38	chile_8_b	30 800	10.32%	1	27 920	0.00%	13	27 920	0.00%	13	27 920	0.00%	31	28 160	0.86%	24	24
	39	chile_9_b	63 500	31.42%	1	48 320	0.00%	20	48 500	0.37%	24	48 320	0.00%	66	48 740	0.87%	32	32
	40	chile_10_b	12 600	1.45%	1	12 420	0.00%	13	12 420	0.00%	13	12 420	0.00%	28	12 660	1.93%	24	24
	41	chile_11_b	34 400	9.14%	1	31 520	0.00%	14	31 520	0.00%	13	31 520	0.00%	33	31 760	0.76%	26	26
	42	chile_12_b	68 900	28.26%	1	53 720	0.00%	20	53 900	0.34%	21	53 720	0.00%	68	54 140	0.78%	30	30
c) Emphasize minimizing delay	43	chile_1_c	8 340 000	2.96%	1	8 100 001	0.00%	11	8 100 001	0.00%	15	8 100 001	0.00%	20	8 100 002	0.00%	24	24
	44	chile_2_c	18 300 000	15.97%	1	15 780 002	0.00%	14	15 780 002	0.00%	13	15 780 002	0.00%	25	15 780 003	0.00%	25	25
	45	chile_3_c	31 380 000	24.82%	1	25 140 002	0.00%	39	25 140 002	0.00%	16	25 140 002	0.00%	36	25 140 004	0.00%	28	28
	46	chile_4_c	8 580 000	4.38%	1	8 220 002	0.00%	12	8 220 002	0.00%	13	8 220 002	0.00%	23	8 220 003	0.00%	24	24
	47	chile_5_c	22 680 000	20.38%	1	18 840 002	0.00%	15	18 840 002	0.00%	14	18 840 002	0.00%	27	18 840 003	0.00%	24	24
	48	chile_6_c	47 580 000	52.50%	1	31 200 002	0.00%	45	31 200 002	0.00%	21	31 200 002	0.00%	47	31 200 005	0.00%	30	30
	49	chile_7_c	10 800 000	3.45%	1	10 440 002	0.00%	13	10 440 002	0.00%	13	10 440 002	0.00%	23	10 440 003	0.00%	23	23
	50	chile_8_c	30 800 000	14.24%	1	26 960 002	0.00%	17	26 960 002	0.00%	20	26 960 002	0.00%	32	26 960 003	0.00%	25	25
	51	chile_9_c	63 500 000	34.76%	1	47 120 003	0.00%	43	47 120 003	0.00%	23	47 120 003	0.00%	50	47 120 005	0.00%	30	30
	52	chile_10_c	12 600 000	2.94%	1	12 240 002	0.00%	13	12 240 002	0.00%	13	12 240 002	0.00%	22	12 240 003	0.00%	23	23
	53	chile_11_c	34 400 000	12.57%	1	30 560 002	0.00%	16	30 560 002	0.00%	15	30 560 002	0.00%	33	30 560 003	0.00%	26	26
	54	chile_12_c	68 900 000	31.19%	1	52 520 003	0.00%	42	52 520 003	0.00%	20	52 520 003	0.00%	45	52 520 005	0.00%	30	30
	Average:			**17.07%**	**1**		**0.00%**	**21**		**0.05%**	**15**		**0.00%**	**33**		**0.56%**	**26**	**26**

GAP_Y = (Y -Y)/Y**

Figure 6.6: *Results of Algorithm 4 (Network 1 and 2) compared to several methods presented in previous chapters.*

	Instance		Method O1: Right-shift Heuristic			Method O4: I-LS Heuristic			Method S3: SAPI Heuristic			Method S4: SAPI Exact			CP-based algorithm (subproblem: SAPI [3.5 h])			
	N°	Name	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [s]	CPU to best [s]
a) Balance stability and minimizing delay	1	france_1_a_7	6 103	64.32%	3	3 714	0.00%	78	3 714	0.00%	72	3 714	0.00%	72	3 714	0.00%	44	44
	2	france_2_a_7	22 405	196.91%	4	7 546	0.00%	53	7 546	0.00%	74	7 546	0.00%	74	7 546	0.00%	44	44
	3	france_3_a_7	48 469	272.09%	3	13 026	0.00%	245	13 026	0.00%	122	13 026	0.00%	122	13 071	0.35%	190	190
	4	france_4_a_7	6 819	9.45%	3	6 230	0.00%	81	6 230	0.00%	71	6 230	0.00%	71	6 230	0.00%	45	45
	5	france_5_a_7	32 692	122.35%	3	14 703	0.00%	70	14 703	0.00%	61	14 703	0.00%	61	15 438	5.00%	211	211
	6	france_6_a_7	68 329	175.21%	3	24 903	0.30%	256	26 460	6.57%	78	24 828	0.00%	525	24 828	0.00%	91	91
b) Emphasize stability	7	france_1_b_7	6 103	52.38%	3	4 005	0.00%	55	4 005	0.00%	58	4 005	0.00%	58	4 005	0.00%	45	45
	8	france_2_b_7	22 405	196.91%	3	7 546	0.00%	52	7 546	0.00%	66	7 546	0.00%	66	7 546	0.00%	45	45
	9	france_3_b_7	48 469	270.61%	3	13 078	0.00%	81	13 078	0.00%	82	13 078	0.00%	82	13 078	0.00%	57	57
	10	france_4_b_7	8 619	32.17%	3	6 521	0.00%	58	6 521	0.00%	63	6 521	0.00%	63	6 521	0.00%	47	47
	11	france_5_b_7	32 692	122.35%	3	14 703	0.00%	57	14 703	0.00%	59	14 703	0.00%	59	14 703	0.00%	54	54
	12	france_6_b_7	68 329	164.18%	4	25 865	0.00%	186	26 461	2.30%	210	25 865	0.00%	210	28 971	12.01%	224	224
c) Emphasize minimizing delay	13	france_1_c_7	6 103 000	67.02%	3	3 654 002	0.00%	55	3 654 002	0.00%	76	3 654 002	0.00%	76	3 654 002	0.00%	45	45
	14	france_2_c_7	22 405 000	196.91%	3	7 546 000	0.00%	54	7 546 000	0.00%	95	7 546 000	0.00%	95	7 546 000	0.00%	46	46
	15	france_3_c_7	48 469 000	273.82%	3	12 966 004	0.00%	107	12 966 004	0.00%	138	12 966 004	0.00%	138	12 966 004	0.00%	70	70
	16	france_4_c_7	8 619 000	39.69%	3	6 170 002	0.00%	85	6 170 002	0.00%	73	6 170 002	0.00%	73	6 170 002	0.00%	45	45
	17	france_5_c_7	32 692 000	122.35%	3	14 703 000	0.00%	67	14 703 000	0.00%	68	14 703 000	0.00%	68	14 703 000	0.00%	46	46
	18	france_6_c_7	68 328 900	176.07%	4	24 835 010	0.34%	261	24 751 006	0.00%	89	24 751 006	0.00%	523	26 433 005	6.80%	213	213
	Average:			**141.93%**	**3**		**0.04%**	**105**		**0.49%**	**86**		**0.00%**	**135**		**1.34%**	**87**	**87**

	Instance		Method O1: Right-shift Heuristic			Method O4: I-LS Heuristic			Method S3: SAPI Heuristic			Method S4: SAPI Exact			CP-based algorithm (subproblem: SAPI [12 h]) - Heuristic			
	N°	Name	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	Cost [$]	GAP [%]	CPU [sec]	CPU to best [s]
a) Balance stability and minimizing delay	19	chile_1_a	8 340	2.77%	1	8 115	0.00%	34	8 115	0.00%	12	8 115	0.00%	12	8 130	0.18%	24	24
	20	chile_2_a	18 300	15.75%	1	15 810	0.00%	14	15 810	0.00%	12	15 810	0.00%	12	15 825	0.09%	24	24
	21	chile_3_a	31 380	24.67%	1	25 170	0.00%	36	25 170	0.00%	18	25 170	0.00%	18	25 200	0.12%	32	32
	22	chile_4_a	8 580	4.00%	1	8 250	0.00%	11	8 250	0.00%	13	8 250	0.00%	13	8 265	0.18%	23	23
	23	chile_5_a	22 680	20.19%	1	18 870	0.00%	13	18 870	0.00%	14	18 870	0.00%	14	18 885	0.08%	24	24
	24	chile_6_a	47 580	52.35%	1	31 230	0.00%	41	31 230	0.00%	16	31 230	0.00%	16	31 275	0.14%	28	28
	25	chile_7_a	10 800	3.15%	1	10 470	0.00%	10	10 470	0.00%	12	10 470	0.00%	12	10 485	0.14%	23	23
	26	chile_8_a	30 800	14.12%	1	26 990	0.00%	14	26 990	0.00%	14	26 990	0.00%	14	27 005	0.06%	24	24
	27	chile_9_a	63 500	34.68%	1	47 150	0.00%	36	47 150	0.00%	16	47 150	0.00%	16	47 195	0.10%	32	32
	28	chile_10_a	12 600	2.69%	1	12 270	0.00%	10	12 270	0.00%	13	12 270	0.00%	13	12 285	0.12%	23	23
	29	chile_11_a	34 400	12.46%	1	30 590	0.00%	15	30 590	0.00%	13	30 590	0.00%	13	30 605	0.05%	24	24
	30	chile_12_a	68 900	31.11%	1	52 550	0.00%	36	52 550	0.00%	14	52 550	0.00%	14	52 595	0.09%	31	31
b) Emphasize stability	31	chile_1_b	8 340	2.21%	1	8 160	0.00%	10	8 160	0.00%	10	8 160	0.00%	10	8 400	2.94%	23	23
	32	chile_2_b	18 300	9.32%	1	16 740	0.00%	13	16 740	0.00%	11	16 740	0.00%	11	16 980	1.43%	24	24
	33	chile_3_b	31 380	19.13%	1	26 340	0.00%	21	26 520	0.68%	13	26 340	0.00%	35	26 760	1.59%	29	29
	34	chile_4_b	8 580	2.14%	1	8 400	0.00%	12	8 400	0.00%	11	8 400	0.00%	11	8 640	2.86%	23	23
	35	chile_5_b	22 680	14.55%	1	19 800	0.00%	11	19 800	0.00%	12	19 800	0.00%	12	20 040	1.21%	24	24
	36	chile_6_b	47 580	46.85%	1	32 400	0.00%	17	32 580	0.56%	14	32 400	0.00%	30	32 820	1.30%	31	31
	37	chile_7_b	10 800	1.69%	1	10 620	0.00%	12	10 620	0.00%	12	10 620	0.00%	12	10 860	2.26%	24	24
	38	chile_8_b	30 800	10.32%	1	27 920	0.00%	11	27 920	0.00%	12	27 920	0.00%	12	28 160	0.86%	24	24
	39	chile_9_b	63 500	31.42%	1	48 320	0.00%	19	48 500	0.37%	16	48 320	0.00%	29	48 740	0.87%	32	32
	40	chile_10_b	12 600	1.45%	1	12 420	0.00%	11	12 420	0.00%	12	12 420	0.00%	12	12 660	1.93%	24	24
	41	chile_11_b	34 400	9.14%	1	31 520	0.00%	12	31 520	0.00%	13	31 520	0.00%	13	31 760	0.76%	26	26
	42	chile_12_b	68 900	28.26%	1	53 720	0.00%	16	53 900	0.34%	16	53 720	0.00%	34	54 140	0.78%	30	30
c) Emphasize minimizing delay	43	chile_1_c	8 340 000	2.96%	1	8 100 001	0.00%	10	8 100 001	0.00%	11	8 100 001	0.00%	11	8 100 002	0.00%	24	24
	44	chile_2_c	18 300 000	15.97%	1	15 780 002	0.00%	11	15 780 002	0.00%	12	15 780 002	0.00%	12	15 780 003	0.00%	25	25
	45	chile_3_c	31 380 000	24.82%	1	25 140 002	0.00%	35	25 140 002	0.00%	16	25 140 002	0.00%	16	25 140 004	0.00%	28	28
	46	chile_4_c	8 580 000	4.38%	1	8 220 002	0.00%	11	8 220 002	0.00%	12	8 220 002	0.00%	12	8 220 003	0.00%	24	24
	47	chile_5_c	22 680 000	20.38%	1	18 840 002	0.00%	13	18 840 002	0.00%	12	18 840 002	0.00%	12	18 840 003	0.00%	24	24
	48	chile_6_c	47 580 000	52.50%	1	31 200 002	0.00%	43	31 200 002	0.00%	19	31 200 002	0.00%	19	31 200 005	0.00%	30	30
	49	chile_7_c	10 800 000	3.45%	1	10 440 002	0.00%	12	10 440 002	0.00%	12	10 440 002	0.00%	12	10 440 003	0.00%	23	23
	50	chile_8_c	30 800 000	14.24%	1	26 960 002	0.00%	16	26 960 002	0.00%	12	26 960 002	0.00%	12	26 960 003	0.00%	25	25
	51	chile_9_c	63 500 000	34.76%	1	47 120 003	0.00%	37	47 120 003	0.00%	21	47 120 003	0.00%	21	47 120 005	0.00%	30	30
	52	chile_10_c	12 600 000	2.94%	1	12 240 002	0.00%	11	12 240 002	0.00%	11	12 240 002	0.00%	11	12 240 003	0.00%	23	23
	53	chile_11_c	34 400 000	12.57%	1	30 560 002	0.00%	15	30 560 002	0.00%	14	30 560 002	0.00%	14	30 560 003	0.00%	26	26
	54	chile_12_c	68 900 000	31.19%	1	52 520 003	0.00%	34	52 520 003	0.00%	20	52 520 003	0.00%	20	52 520 005	0.00%	30	30
	Average:			**17.07%**	**1**		**0.00%**	**19**		**0.05%**	**14**		**0.00%**	**15**		**0.56%**	**26**	**26**

GAP_Y = (Y -Y)/Y**

Figure 6.7: *Comparative Analysis for the CPU time required for computing the best solution (Network 1 and 2). Algorithm 4 versus several methods presented in the previous chapters.*

Part III

Generalization of SAPI

Chapter 7

Generalization of SAPI

Contents

Abstract of the Chapter

All rescheduling problems share singular characteristics that can be exploited by a generic version of SAPI. In this chapter you will find a generalization of SAPI, initially developed for the railway rescheduling problem in Chapter 5. This methodology is divided into two big phases composed in total of nine steps. A complementary application of SAPI is then presented in the next chapter.

7.1 Introduction

Scheduling concerns the arrangement of a number of related operations in time. In this manner, there are many different kinds of scheduling problems depending on the application field. For example, manufacturing scheduling problems consist in allocating different resources (machines) to task (operations) to create products by maximizing the revenues (or minimizing costs). On the other hand, transportation scheduling problems usually concerns the construction of a timetable: arrival and departure times of every vehicle or container (*e.g.*, buses, trains, or airplanes) for the corresponding element in the network (*e.g.*, bus stops, stations, airports, or airport gates). Both, manufacturing and transportation, have to deal with several constraints: following a feasible sequence,

respecting minimal and maximal processing (translating) time, assuring combination of parts (correspondences), and others depending on the characteristics of every problem.

As it was introduced in Chapter 3, disruption management and rescheduling can be defined as a reactive process of repairing a disturbed schedule after unexpected incidents. Some examples of possible incidents are: breakdown, scraps, due date's changes, order cancelations, and other kind of delays.

There are two important characteristics in general disruption management problems that are exploited by this method. First, because it is a repair problem, a reference base solution is known. It is expected that this reference plan has been optimized with a previous method, and it is necessary to return to this plan as soon as possible. This solution corresponds to the planned schedule before disturbances, called the *original plan*. And second, it is expected that disturbances are propagated depending on some factors. Therefore, in simple words, the thesis of the methodology is that it is possible to determine which part of the problem will be affected by given disturbances considering some easy to calculate factors. Thus, computational effort will be concentrated in the most affected area of the system, while the rest could be equal or very similar to the original plan and discarded in the solution search space. As a result, the CPU time could decrease drastically without losing possibility to obtain good solutions.

In particular, to study the propagation of disturbances, we propose to use a statistical analysis with logistic regression widely used in different science fields. Then, the method uses the results of this regression for reducing the complexity, *e.g.*, by fixing integer variables in a MIP model. In this way, SAPI reduces the number of variables and so on the size of the feasible region, *i.e.*, the solution search space.

The remainder of the chapter is organized as follows. The proposed methodology is developed in Section 7.2. The first phase (*learning*) is presented in Section 7.3 while the seconde one (*implementation*) is presented in Section 7.4. Finally, some conclusions are drawn in Section 7.5. A complementary application of SAPI is discussed in Chapter 8.

7.2 Description of the Methodology

The proposed methodology is divided in two phases: design & learning and implementation. Every phase has several steps (see Table 7.1). It should be noted that these steps are general guidelines that must be adapted to the problem to solve.

The first phase is *Design & Learning*. The objective is to formulate the problem and estimate the required parameters. This phase is divided in five steps. As other design tasks, there is not a detailed manual and it requires creativity and knowledge in the problem with the purpose of having good results. As a design and learning phase, this phase will be performed only once (obviously it could be reviewed) and, in general, it is invariable for every instance of the problem. The results of this phase give a guide and parameters for making an algorithm implemented in the second phase.

On the other hand, the *Implementation* phase is the set of steps used to solve each

instance. Thus, these steps have to be repeated every time one wants to solve a specific instance, *i.e.*, these steps define a solution algorithm.

Table 7.1: *Phases and steps of the methodology*

Phase	Step
I. Design & Learning	1. Defining events. 2. Formulating the rescheduling problem (modeling). 3. Defining the aspect to be reduced. 4. Defining a regression model. 5. Estimating coefficients for the regression model.
II. Implementation	6. Calculating an initial solution X^0. 7. Calculating values of the factors and the probabilities. 8. Reducing the complexity using the probabilities. 9. Solving the problem.

7.3 Phase I. Design and Learning

Step 1. Defining Events in the Scheduling Problem

The term "event" is understood as a point in time that represents the start or completion of a set of activities, *i.e.*, operations or movements of vehicles. Thus, it is necessary to associate the decision variables of the model to specific events of the problem. Obviously, this task depends on the chosen formulation. For example, for the railway rescheduling problem described in Chapter 4, events are the arrivals and departures of trains. In general, these events are strongly interconnected. As a consequence, any disturbance on a specific event may affect also other related events.

Step 2. Formulating the Rescheduling Problem (Modeling)

This step consists in creating an abstract model using a mathematical language for describing a specific rescheduling problem. The model has to describe the system using a set of variables and constraints that determine relationships between variables. The values of variables could be: real or integer number, boolean (binary) values or strings. In general, models could by classified by different perspectives: linear or nonlinear, static vs. dynamic, deterministic vs. probabilistic (or stochastic). As a result of this step, it is possible to have for example: a constraint programming (CP) formulation or a mixed integer programming (MIP) formulation.

In addition to the formulation, the "original plan" must be known. The original plan is the schedule arrangement before any disturbances. Thus, the objective of the

formulation is to obtain a new provisional plan as close as possible to the original one, *i.e.*, minimizing the impact of disturbances. Therefore, if there are no disturbances, the new provisional plan should be exactly the original plan. As a result, minimizing the impact of disturbances is to minimize the difference between both plans. Here, the word "difference" could be, for example, the delay occurred by the disturbance, or changes in assignment of resources.

Step 3. Defining the Aspect to be Reduced

To perform this step, it is necessary to answer this question: if there is an "oracle" telling us what (and how) events are going to be affected by a given disturbance, how is it possible to use this information to reduce the problem?

The answer of this question depends on the model. In general, it will be possible to eliminate several variables corresponding to events not affected, or to define efficient branching rules in Branch and Bound algorithms.

As the objective of this methodology is to simplify the problem, a good way to reduce the problem it is to reduce the source of combinatory. For example, in MIP formulations, we are more interested in reducing the number of integer variables than the continuous one.

Therefore, in this step we could define what kinds of integer variables may be fixed. For constraints models, for example, we could be interested in reducing the search space (domain reduction) or customizing an appropriate search strategy

Step 4. Defining a Regression Model: Logistic Regression Analysis

Logistic regression has been used in diverse fields with very good results. Logistic regression is part of a category of statistical models called generalized linear models. General linear models analyze one or more continuous dependent variables and one or more independent variables, whether they are categorical or quantitative. Note that an independent variable is typically the variable being changed and the dependent variable is the observed result of the independent variable being changed, *i.e.*, the dependent variable depends on the value of the independent variables. The relationship between a dependent variable and some independent variables is expressed as an equation that contains a term for the weighted sums of the values for the independent variable, plus a term for all the unknowns, *i.e.*, the error term.

In particular, logistic regression analysis allows analysts to estimate multiple regression models when the response is binary, *i.e.*, 0 or 1. Therefore, our model has two possible outcomes: 1, if the variable associated to an event is affected by the disturbance, and 0 in other case.

The relationship between the predictor (independent) and response (dependent) variables is not a linear function in logistic regression; instead, the logistic regression

function is:

$$\theta = \frac{e^{(\alpha+\sum_{i=1}^{n}\beta_i z_i+\varepsilon)}}{1+e^{(\alpha+\sum_{i=1}^{n}\beta_i z_i+\varepsilon)}} \tag{7.1}$$

Where:

α: Constant of the equation and,

β: Coefficient of the predictor variables.

ε: Error term.

The possible outcomes for each response variables are 1, with a probability θ, and 0 with the probability 1-θ, where 0 <= θ <= 1. Let us notice that this type of variables is also called Bernoulli variables in probability theory terminology.

Finally, if we associate to each event a response variable then the value 1 indicates that the event may be affected by a disturbance with the probability θ or not with a probability 1-θ.

On the other hand, the independent variables can take any form, because logistic regression makes no assumption about the distribution of the independent variables. Therefore, they do not have to be normally distributed, linearly related or of equal variance within each group.

To find the predictor variables, it is necessary to identify a set of factors that may explain why an event is affected or not by a disturbance. Unfortunately, there are no rules to define such factors. Depending on how an event is defined for a given problem, different factors may be found. Nevertheless, some "generic" factors (reported below) can be defined, but should be fitted on the model that has to be solved:

- A measure of distance between the first disturbance and the event: one expects to have a negative correlation, *i.e.*, the longer distance from the original disturbance, the lower probability to be affected.
- A quantification of resources shared close to the event. The probability to be affected may be partially explicated by the number of resources in the neighborhood of the event.
- A quantification of density of other events close to the event. Dense system are prone to be affected by disturbance.

Step 5. Estimating Coefficients for the Regression Model

The goal of this step is twofold: estimating the coefficients and validating the significance for the regression model. Both are result of the same process of analyzing a statistical sample.

The process can be explained as follows. After solving an easy instance using, for example, state-of-the-art solvers; we have the values for the decision variables in the solution and other information that permits calculating the values of the independent variables. Both independent variables and observed values for the dependent variable compose a "statistical sample". With this sample, it is possible to estimate the coefficients of the regression model using an appropriate method, for example the method of least squares. This approach finds the best fit in the least-squares sense by minimizing the sum of squared residuals. In this context, a residual is the difference between an observed value of the dependent variable and the value given by the regression model.

The second objective of this step is to validate the regression model. Consequently, it is necessary to study the significance of the whole model, in general; and of each factor in particular.

A good method to study the fitness of the model and its variables is *Chi-Square Goodness-of-Fit Test* that determines whether the logistic function adequately fits the observed data. Also, it is necessary to estimate correlations between the coefficients in the fitted model. These correlations can be used to detect the presence of serious multicollinearity, *i.e.*, correlation amongst the predictor variables, and determining which factors could be redundant with others.

Sometimes the whole model is significant, but it could contain too many independent variables and the elimination of some of them should be studied. *Stepwise regression* is used in the exploratory phase of research where the goal is to discover relationships. Between the main approached in stepwise regression, *backward elimination* appears to be the preferred method of exploratory analysis, where the analysis begins with a full or saturated model and variables are eliminated from the model in an iterative process. The fit of the model is tested after the elimination of each variable to ensure that the model still fits the data. When no more variables can be eliminated from the model, the analysis has been completed.

7.4 Phase II. Implementation

The model provided by the previous steps has to be naturally implemented and solved. We call this stage "Implementation Phase". It is necessary in this phase to choose an adequate programming language and software, for instance IBM ILOG CPLEX for a MIP model or IBM ILOG CP Optimizer for a CP model. But, it is also necessary to fix as better as possible a series of aspects dealing with the initial solution, factor values, and preprocessing aspects. These points are discussed below.

Step 6. Calculating an Initial Solution from the Original Plan

From the original schedule an initial solution may be quickly computed by keeping unchanged the order of tasks, (*e.g.*, trains, flights, and buses) and the assignment of resources (*e.g.*, tracks, gates, and aircrafts). This procedure, called "Right-Shift Reschedul-

ing" is general, in the sense that it can be applied to any rescheduling problem (Ovacik and Uzsoy, 1992). The resulting solution is in many cases not good (far from the optimal solution) since it is equivalent to directly propagate delays to other tasks. Nevertheless, it is very easy to compute and many of the events (not concerned by the disturbance) of such solution are exactly the same as in the optimal solution.

Step 7. Calculating Values of the Factors and Probabilities

In the design & learning phase it was defined a regression model and coefficients for significant factors. The goal of this step is to evaluate the regression model for every select decision variable. Note that, in practice, the regression model (see equation 7.1) is going to give a real number $0 \leq \theta \leq 1$. This value is interpreted as the probability of variables to be affected by the disturbance.

Step 8. Reducing the complexity using the probabilities

There are several ways to reduce the problem using the calculated probabilities. In this section we expose four strategies: using random numbers for selecting variables to fix; taking the x% of variables with higher probabilities in the reduced problem, while the rest are fixed; splitting the set of variables in several subsets according to the probability, and for every subset add appropriate cuts; and finally mixing the previous three strategies.

Strategy 1. Using random numbers to select variables to fix This strategy consists in generating Bernoulli random variables for every variable considered in the regression model. The Bernoulli distribution is a discrete probability distribution that takes value 1 with probability p and 0 with probability $1 - p$. Thus, it will be used a random number generator to generate Bernoulli variables where $p = \theta$ (the probability calculated in the previous step using the regression model). If the random generation gives a value equal to 0, the variable associated will be fixed to its value in the current incumbent solution, *e.g.*, initial solution. The rest of the decision variables will be calculated by a solving procedure.

The number of eliminated variables could be calculated.

In a special case that all variables are bernoulli variables with the similar characteristic, i.e., the same value for the probability $p = \theta_0$, the distribution of the number of the fixed variables is called *Binomial* and its mean is:

$$\begin{aligned} y &= E(x) \\ &= n(1 - \theta_0) \end{aligned}$$

Where:

y: Expected number of fixed variables.

n: Number of variables possible to fix.

$x \sim B(n, 1 - \theta_0)$

On the other hand, if the values of the probability p, for each Bernoulli variable, differs then the mean is calculated by

$$y = \sum_{j=1}^{n} (1 - \theta_j) \tag{7.2}$$

Where:

y: Expected number of fixed variables.

n: Number of variables possible to fix.

Using this strategy, it is possible to adapt a method called *Mimausa* proposed by (Mautor and Michelon, 1997) (Mautor and Michelon, 2001). Mimausa is an iterative heuristic method and at each iteration, a limited set of k variables are selected while the other variables are fixed to their current value. Then a reduced size subproblem with only k variables is then solved optimally.

Adapted to this method, one performs several iterations updating the reference plan with the last best known solution. Because of the random generator of Bernoulli variables, at each iteration the procedure probably takes different variables. Note that each iteration gives, at least, the same current solution.

Strategy 2. Taking the x% of variables with the highest probabilities in the reduced problem, while the rest are fixed It is exactly the same base idea of fixing variables that Strategy 1. The difference is, instead of generate a random Bernoulli variables, this strategy will fix all variables with the highest probabilities (θ) until select the x% of the total number of variables. As a consequence, the number of the fixed variables is a constant.

A significant difference with the first strategy is that this strategy will give exactly the same result for two independent running of the same instance because there is no random numbers involved in the process.

Strategy 3. Splitting the set of variables in several subsets according to the probability, and for every subset add appropriate cuts This strategy is used only in the case when the associated variables (or some of them) are binary.

Strategies 1 and 2 are based on (hard) variable fixing or diving idea. The main problem is that it is very hard to detect if the selection of the variables was appropriated. So, a good question is how to fix variables without losing the possibility of finding

good solutions. A soft fixing idea was given by (Fischetti and Lodi, 2003) using *Local Branching* cuts.

Let B, represents index set of binary variables. Additionally, X^0 corresponds the vector of values of a feasible solution of the problem, using the original or reference plan. Let $S = \{j \in B : x_j^0 = 1\}$ and $R = \{j \in B : x_j^0 = 0\}$. For a known integer parameter k, it is possible to define a $k - OPT$ neighborhood as a set of feasible solutions of the problem, and satisfying the additional local branching constraint:

$$\sum_{j \in S}(1 - x_j) + \sum_{j \in R}(x_j) \leq k \tag{7.3}$$

The constraint 7.3 gives the maximal number of variables that will split their values i.e. from one to zero or from zero to one. Adding this constraint to the model, the space search solution will decrease.

The strategy 3 consists in adding several local branching constraints, each of them grouping variables according to the probability calculated earlier. Variables with high probability to be affected will use a high value of k. That means that a large number of changes will be allowed for them. At the opposed, a group of variables with less probability to be affected will use a modest value of k.

Strategy 4. Mixing strategy 2 and 3 To use a mix strategy, it is possible to split the set of variables in three subsets as follows:

- The first subset will contain X% of free variables with higher probabilities to be affected.
- The second subset will contain other Y% of the variables with higher probabilities for which local branching cuts will be used. Let us notice that this set could be also split in several subsets using different values of k: large values of k for high probability variables and lower values of k for low probabilities variables.
- The third subset will contain the rest of the variables fixed to their values in the theoretical plan.

Step 9. Solving the problem

Following problem reduction, the last step is to solve it. At this point it is necessary to define two aspects: iterations and the ending criteria.

This procedure could be iterative or not. A non iterative method consists in just solving the first reduced problem found and return the solution. In contrast, an iterative procedure solves several reduced size subproblems with the intention to reach the optimum.

The question here it is how to form new reduced problems in each iteration. Depending on the way the problem was reduced (see the previous step to select a strategy), it is possible to repeat the procedure changing the reference solution (incumbent) and to solve again. Another possibility is to modify the values of the probabilities, for example at each iteration multiplying the values of all probabilities by a factor $g \geq 1$ in order to increase the size of the search space.

There are many possible ending criteria. First, a maximal number of iterations without improvements in the objective function. Second, a maximal running time. And finally, the algorithm must stop when the reduced problem corresponds exactly to the original one's.

7.5 Conclusions

In this chapter a new methodology, called SAPI (Statistical Analysis of Propagation of Incidents), has been proposed for disruption management problems. One of the characteristics of these kinds of problems is that a base/reference schedule is known, and the objective it is to create a new provisional schedule as close as possible to the original plan in order to return as soon as possible to normal (optimized) operations. This main aspect is exploited by SAPI. The method focuses the computational effort in the part mostly affected by disturbances.

SAPI assumes that the effects of disruptions can be propagated to other upcoming events. Nevertheless, this propagation is not uniform to all events and could be forecasted by a statistical analysis. For example, events near to the incidents have a high probability to be disrupted and it seems to be logic to try rescheduling them. In contrast, events that are far away from the incidents will be probably unchanged (in comparison to the original plan).

The main idea in SAPI is to suppose that the probability for an event to be affected by a given disturbance can be determined by several factors. A statistical analysis, using logistic regression, is used to compute these probabilities.

Afterwards, resulting probabilities allow reducing the problem by eliminating from the problem all events for which corresponding probabilities are under a prefixed threshold.

Two different applications of SAPI can be consulted in Chapter 5 and Chapter 8.

Chapter 8

Application of SAPI for Rescheduling Flights and Passengers

Contents

Abstract of the Chapter

This chapter deals with an application of SAPI to another rescheduling problem in the airline industry. The problem to solve is closed to the train rescheduling one's and has been proposed in the ROADEF challenge 2009 on *Disruption Management for Commercial Aviation*, an international algorithmic challenge organized by the French Operational Research Society (ROADEF) . The problem consists in rescheduling flights, aircraft and passengers simultaneously after some disturbances. SAPI has been adapted successfully for this problem and tested using the instances of the challenge. The algorithm was compared with several approaches developed by 27 different research teams. The

procedure shows to be viable in practice and obtained the third place in that international competition

8.1 Introduction

Airline industry is one of the most competitive and capital-intensive industry in the world. Suitable use of resources and scheduling of operations have a direct impact on company returns, this is the reason why optimization and OR techniques have an important role. Airlines operate following an optimized schedule of flights taking into account many factors such as the availability of crew members, the demand of passengers, and several other operational constraints. Nevertheless, following normal operations is not always possible. Day after day, many little disruptions perturb the original schedule that becomes suboptimal and, in some cases, infeasible. For example in the USA, only 76.04 percent of flights were *on-time* from January 2008 to December 2008 while the rest were canceled or delayed, *i.e.*, they arrived 15 or more minutes later than the original schedule[1].

As a consequence airlines are more and more interested in systems to return as quickly as possible to normal operations after disruptions. A very good example of this interest can be observed with the software called "CrewSolver" developed by CALEB[2] for Continental Airlines. This decision support system uses operations research to generate globally optimal, or near optimal, crew recovery solutions while satisfying multiple complex constraints. Thus it allows pilots and flight attendants to return to their original schedules quickly after disruptions. The impact of this system was internationally recognized and the creators of the systems won the Franz Edelman Award[3] in 2002 (Yu et al., 2003).

Good surveys of disruption management for flight scheduling can be found in (Filar et al., 2001), (Yu and Qi, 2004), (Clausen et al., 2005). The first articles published in the literature appeared in the mid-1980s with the works of Teodorovic *et al.* (Teodorovic and Guberinic, 1984), (Teodorovic and Stojkovic, 1990), (Teodorovic and Stojkovic, 1995). In these works the problem is to find a new daily flight schedule in circumstances where some aircraft become unavailable. Later, Jarrah *et al.* (Jarrah et al., 1993) presented the first relevant computational results indicating cost reductions between 20% and 90% compared with an unoptimized schedule recovery procedure. This approach was implemented by United Airlines reporting $540,000 savings in delay costs from October 1993 to March 1994 (Rakshit et al., 1996).

A relevant work considering an integrated approach was presented by Lettovskŕ in 1993 (Lettovský, 1997). In this work the Airline Integrated Recovery (AIR) procedure has been proposed. Many systems are currently based on AIR. The AIR model integrates decision variables and constraints belonging to three aspects: crew assignment,

[1] Bureau of Transportation Statistics. http://www.transtats.bts.gov/
[2] CALEB Technologies Corp. http://www.calebtech.com/
[3] INFORMS Edelman Award is also known as the "super-bowl" of Operations Research

aircraft routing, and passenger flow. This combination was computationally intractable because of the complexity of the mixed integer linear programming formulation created to model the problem. The AIR is then decomposed into subproblems where the master problem is the *Schedule Recovery Model* (SRM). An iterative procedure solves SRM first, then the *Aircraft Recovery* problem and the *Crew Recovery* problem. This procedure is repeated until a satisfactory solution is found. The third and last subproblem, the *Passenger Flow* model, finds new itineraries for disrupted passengers.

Nevertheless, trying to return to normal operations by decomposing the problem into sequential decision levels does not guarantee a global optimum, and the last aspect (passengers) tends to be more affected. For this reason we consider an integrated point of view, addressing the problem of rescheduling aircraft, flights, and passengers simultaneously after disruptions.

The remainder of the chapter is organized as follows. The next section details the airline rescheduling problem. In Section 8.3 a Mixed Integer Programming formulation (MIP) is presented to model this problem. Our algorithm proposed for solving the problem is described in Section 8.4. Section 8.5 details the main aspect of the algorithm: SAPI (*Statistical Analysis of Propagation of Incidents*) that is based on the regression analysis developed in Section 8.6. In Section 8.7 we report the results of the computational tests performed in the context of the competition ROADEF challenge 2009. Finally, our conclusions and future directions of research are summed up in the last section.

8.2 Description of the Problem

We consider the problem of repairing a perturbed flight schedule after disruptions as presented in the ROADEF challenge 2009 "*Disruption Management for Commercial Aviation*" (ROADEF, 2008). Figure 8.1 outlines this problem with three main components: data (inputs), optimization procedure and results (outputs).

Before detailing the problem, we present the basic terminology used in this document. A *flight* refers to a scheduled airline trip, which is an origin airport which is associated with a departure time and the destination airport which is associated with an arrival time. An *aircraft* refers to a physical airplane identified by a unique code that is used to fulfill a flight. A *rotation* is an aircraft route, *i.e.*, is a sequence of connected flights that are served by an aircraft. An *itinerary* is a set of one or several *passenger(s)* sharing basic characteristics for their trip. The description of the itinerary is composed of one or several flight legs with one cabin class specified for each leg.

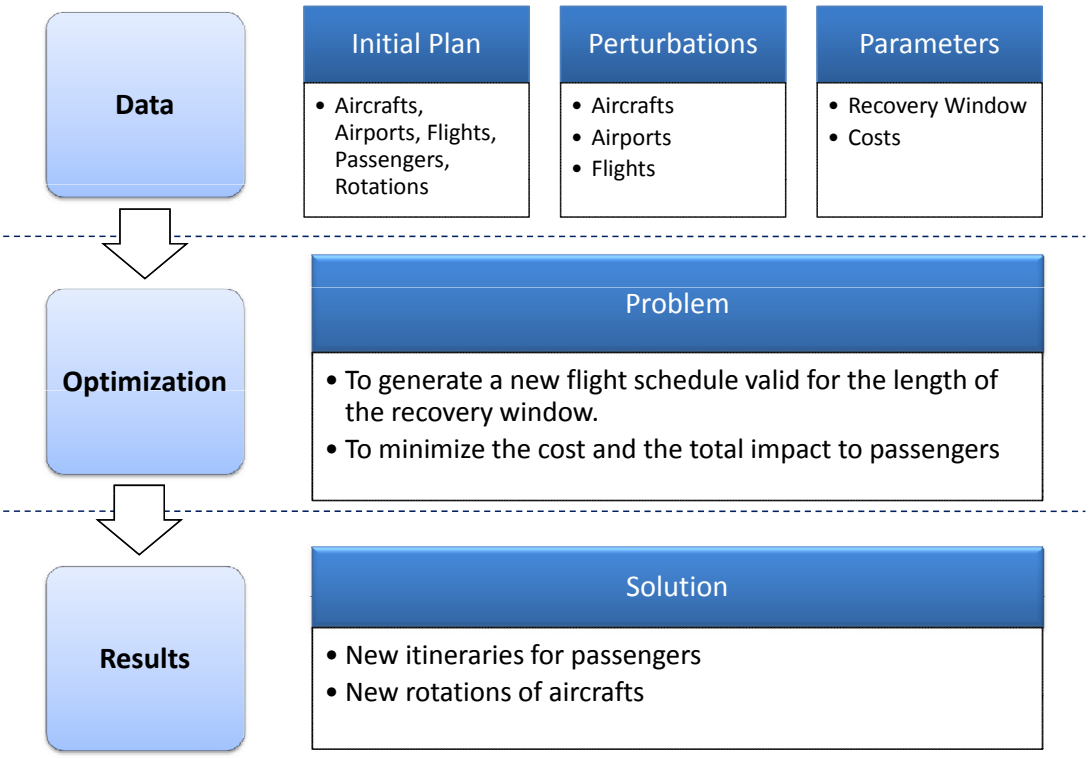


Figure 8.1: *The problem: ROADEF challenge 2009.*

Data (Inputs)

Foremost, an original (unperturbed) schedule is known, that is departure/arrival times, aircraft rotation plan, and the itineraries (origin - destination - flights) for a set of passengers. Other initial information is also given: aircraft and airport capacities, the length of the recovery window, and unitary costs for calculating the objective function.

Whatever the disruptions are, they are modeled as followings:

- Flight delays: A given lateness in departure and arrival of some flights.
- Cancelation of flights: A flight that was canceled and cannot be served by any aircraft.
- Unavailability of aircraft: An aircraft that cannot fly in a given period of time.
- Temporary reductions in airport capacities: The capacity of departures and arrivals are limited for a given period of time.

Optimization

The goal is to recover a perturbed flight schedule through different decisions such as delaying/canceling flights, changing aircraft assignments, and rescheduling/canceling passengers. In particular, we are interested in a new provisional schedule that minimizes both: impact in operational costs and impact for passengers.

Additionally, several constraints have to be considered. They can be classified in two types: hard and soft. On the one hand, hard constraints must be respected in any feasible solution. In other words, if one of the constraints is violated the solution is automatically considered unfeasible and cannot be implemented. On the other hand, soft constraints could be interpreted as "desirable" requests. Violations of such constraints are penalized in the objective function. Hard and soft constraints are described below.

Hard Constraints

H_1 : Aircrafts have a limited seating capacity distributed in different cabins.
H_2 : All aircraft have to perform all planned maintenances.
H_3 : Airports have a limited capacity.
H_4 : There is a minimum connection time for every passenger.
H_5 : There is a minimum turn-round and transit time for aircraft.
H_6 : Surface public transportation cannot be modified.
H_7 : A modified itinerary must have the same final destination as the original itinerary
H_8 : The maximum total delay for passengers at the destination (as compared to the original itinerary) must not exceed 18 hours for domestic and continental flights, and 36 hours for an intercontinental flight.
H_9 : All flights have to respect the range of the aircraft.
H_{10} : The duration of original flights is fixed.

Soft Constraints

S_1 : Each aircraft has to be at its final position by the end of the recovery time window.
S_2 : Each passenger has to arrive to their original destination on-time.
S_3 : Flights cannot be canceled.
S_4 : Passengers cannot be canceled.
S_5 : Passengers cannot be downgraded, *e.g.*, passing from first to economic class.

Results (Outputs)

Finally, the output is composed of two elements: new itineraries and new rotations. The first one is a file with new routes for passengers considered in the problem. These

itineraries include a list of all the flights already taken by passengers before the beginning of the recovery window, and the flights that passengers will have to take during the recovery window. The second element of the solution is the new flight/aircraft schedule, known as rotations. Every flight is associated to one aircraft (or canceled), specifying the new arrival and departure times.

8.3 Mathematical Formulation

We propose a Mixed Integer Programming (MIP) formulation to model this problem. This formulation involves all constraints presented in the previous section. Roughly speaking, "hard constraints" are modeled as restrictions that the decision variables must satisfy, and the objective function represents the weighted average cost or penalization of violating "soft constraints". Note that these penalty coefficients were given for every instance of the ROADEF challenge 2009.

The MIP model could be interpreted as two integrated multi-commodity flow problems: the first one related to aircraft flowing through airports and the second one to passengers flowing through flights. As other multi-commodity flow problems, the mathematical formulation includes capacity, flow conservation and demand satisfaction constraints. This particular problem is NP-complete, because the original multi-commodity flow problem is shown to be NP-complete for integer flows (Even et al., 1976).

In this section we describe all elements of the model: indices, sets, data, costs, decision variables, objective function, and constraints.

Indices and Sets

The indices and the general sets used in this formulation are:

p: Index for airports.

i: Index for aircraft.

j: Index for flights.

k: Index for itineraries.

m: Index for cabin types.

t: Index for time.

P: Set of airports.

F: Set of flights in the recovery time window.

A: Set of aircraft.

I: Set of itineraries.

C: Set of cabin types.

T: Set of slots of time in the recovery time window.

$FlightAirport_p^A$: Set of flights arriving to airport $p \in P$.

$FlightAirport_p^D$: Set of flights departing from airport $p \in P$.

Additional sets, defined to reduce the size of the problem, are:

$DirIt$: Set of passengers who require a direct flight.

$ConIt$: Set of passengers who are allowed to take two or more flights (with connections).

$CompF_i^A$: Set of flights compatible with aircraft $i \in A$.

$CompF_k^I$: Set of flights compatible with itinerary $k \in I$.

$CompNF_j^F$: Set of (next) flights compatible with an aircraft after flight $j \in F$.

$CompA_j^F$: Set of aircraft compatible with flight $j \in F$.

$CompI_j^F$: Set of itineraries compatible with flight $j \in F$.

$CompFF_i^A$: Set of the first flights compatible with aircraft $i \in A$, *i.e.*, only one of these flights can be the first flight of aircraft i.

$CompFF_k^I$: Set of the first flights compatible with itinerary $k \in I$, *i.e.*, only one of these flights can be the first flight of itinerary k.

$CompLF_k^I$: Set of the last flights compatible with itinerary $k \in I$. *i.e.*, only one of these flights can be the last flight of itinerary k.

$CompAP_i^A$: Set of airports compatible with aircraft $i \in A$.

$CompAP_k^I$: Set of airports compatible with itinerary $k \in I$.

$Conn_k^I$: Set of pairs of flights where a connection for itinerary $k \in I$ is possible.

Data

Other important data are:

n_k: Number of passengers of itinerary $k \in I$.

Ori_k: Origin airport of itinerary $k \in I$.

$Dest_k$: Destination airport of itinerary $k \in I$.

$Start_j^{F0}$: Original departure time of flight $j \in F$.

End_j^{F0}: Original arrival time of flight $j \in F$.

End_k^{I0}: Original arrival time of itinerary $k \in I$.

AF_{ij}^0: 1, if aircraft $i \in A$ is assigned to flight $j \in F$ in the original schedule; 0 otherwise.

IF_{kj}^0: Original number of passengers for itinerary $k \in I$ in flight $j \in CompF_k$.

$Airport_i^0$: Initial airport for aircraft $i \in A$.

d_j: Duration of flight $j \in F$.

TR_i: Turn-round time: minimum time to prepare the aircraft $i \in A$ for the subsequent flight.

CT: Connection time for passengers.

R_i: Range for aircraft $i \in A$ (maximal duration of a flight).

$CapP_{im}^A$: Capacity of passengers for aircraft $i \in A$ in cabin type $m \in C$.

$CapA_{pt}^P$: Capacity of arrival flights for airport $p \in P$ in time $t \in T$.

$CapD_{pt}^P$: Capacity of departure flights airport $p \in P$ in time $t \in T$.

Lim_t^{LB}: Lower limit of time $t \in T$.

Lim_t^{UB}: Upper limit of time $t \in T$.

M_1, M_2, M_3, M_4: Large constants.

Unitary Costs

The unitary costs for penalization in the objective function are :

c_{ij}^{BP}: Unitary cost for bad position, if the last flight of aircraft i is j.

c_{kjm}^{DW}: Unitary cost for downgrading, if one passenger of itinerary k take flight j in cabin m.

c_k^{CP}: Unitary cost for canceling one passenger of itinerary k.

c_k^{CL}: Unitary cost (legal) for canceling one passenger of itinerary k.

c_k^{DP}: Unitary cost for delaying one minute one passenger of itinerary k.

c_k^{DL}: Unitary cost (legal) for delaying one minute one passenger of itinerary k.

c_j^{CanF}: Unitary operational cost associated with flight j.

Decision Variables

Decision variables of this model are:

$Start_j$: Departure time of flight $j \in F$.

End_j: Arrival time of flight $j \in F$.

$Delay^I_{kj}$: Delay of itinerary $k \in I$ if the last flight is $j \in CompLF^I_k$

AF_{ij}: 1, if aircraft $i \in A$ is assigned to flight $j \in CompF^A_i$; 0 otherwise.

FF_{ij}: 1, if flight $j \in CompF^A_i$ is the first flight of aircraft $i \in A$; 0 otherwise.

LF_{ij}: 1, if flight $j \in CompF^A_i$ is the last flight of aircraft $i \in A$; 0 otherwise.

IFC_{kjm}: 1, if itinerary k takes flight $j \in CompF^I_k$ in cabin type $m \in C$; 0 otherwise.

CF_j: 1, if flight $j \in F$ is canceled; 0 otherwise.

CI_k: 1, if itinerary $k \in I$ is canceled; 0 otherwise.

$\omega_{ij\hat{j}}$: 1, if flight j is before flight $\hat{j}$ for aircraft i; 0 otherwise.

ρ_{kj}: 1, if itinerary k uses flight j; 0 otherwise.

$Hour^D_{jt}$: 1, if flight $j \in F$ departs in time $t \in T$; 0 otherwise.

$Hour^A_{jt}$: 1, if flight $j \in F$ arrives in time $t \in T$; 0 otherwise.

$CostBadPosition$: Total cost for bad final position of aircraft.

$CostDown$: Total cost for downgrading passengers.

$ConstCanceled_{Pax}$: Total cost associated with cancelation of trips.

$CostCanceled_{Legal}$: Total legal cost associated with cancelation of trips.

$CostDelay_{Pax}$: Total cost associated with delay of passengers.

$CostDelay_{Legal}$: Total legal cost associated with delay of passengers (food and hotel).

$CostCancelFlights$: Total operational cost of the canceled flights.

Objective Function

The problem studied in this chapter can be formulated as follows:

$$\begin{aligned}\text{Minimize }: & CostBadPosition + CostDown + CostCanceled_{Pax} + CostCanceled_{Legal} + \\ & CostDelay_{Pax} + CostDelay_{Legal} - CostCancelFlights \end{aligned} \tag{8.1}$$

The objective function (8.1) is to minimize the rescheduling cost, defined as the weighted sum of the absolute deviations between the planned and the provisional schedule. The weighs for each individual cost are defined by unitary cost parameters, *e.g.*, c_k^{DP} is the unitary cost for delaying one minute one passenger of itinerary k.

Note also that each component in the objective (8.1) represents a penalization of the deviations from the original plan. For example, if all aircraft finish at the planned airport at the end of the recovery window, then $CostBadPosition = 0$. In the same way, it should be noted that there are not canceled or delayed passengers in the original plan (without disruptions).

Constraints

The constraints of this model are classified in several sets: (a) aircraft and flights, (b) capacity of airports, (c) passengers and aircraft seating capacity, (d) cost, and (e) domain of variables.

Aircraft and Flights

The constraints associated to aircraft and flights are:

$$CF_j + \sum_{i \in CompA_j^F} AF_{ij} = 1 \qquad \forall j \in F \qquad (8.2)$$

$$End_j = Start_j + d_j \qquad \forall j \in F \qquad (8.3)$$

$$Start_{\hat{j}} \geq End_j + TR_i\omega_{ij\hat{j}} - M_1(1 - \omega_{ij\hat{j}}) \qquad \forall i \in A; j \in CompF_i^A; \hat{j} \in CompNF_j^F \qquad (8.4)$$

$$\sum_{\substack{j \in CompF_i^A \\ \hat{j} \in CompNF_j^F \cup CompF_i^A}} \omega_{ij\hat{j}} = AF_{i\hat{j}} - FF_{i\hat{j}} \qquad \forall i \in A; \hat{j} \in CompF_i^A \qquad (8.5)$$

$$\sum_{\substack{j \in CompF_i^A \\ \hat{j} \in CompNF_j^F}} \omega_{ij\hat{j}} = AF_{ij} - LF_{ij} \qquad \forall i \in A; j \in CompF_i^A \qquad (8.6)$$

$$\sum_{j \in CompF_i^A} FF_{ij} \leq 1 \qquad \forall i \in A \qquad (8.7)$$

$$\sum_{j \in CompF_i^A} LF_{ij} \leq 1 \qquad \forall i \in A \qquad (8.8)$$

$$AF_{ij} \geq FF_{ij} \qquad \forall i \in A; j \in CompF_i^A \qquad (8.9)$$

$$AF_{ij} \geq LF_{ij} \qquad \forall i \in A; j \in CompF_i^A \qquad (8.10)$$

Constraints (8.2) state that only one aircraft is assigned to each flight j, otherwise the flight is canceled. Flight duration is fixed to $d_j, \forall j \in F$ in constraints (8.3).

Constraints (8.4) assure the minimum time, TR_i, to prepare aircraft $i \in A$ between flight j and flight $\hat{j}$ (the subsequent flight) where $\omega_{ij\hat{j}} = 1$ if flight j is before flight $\hat{j}$

for aircraft i and 0 otherwise. Let M_1 be a "sufficiently" large constant, *e.g.*, the ending time of the recovery window. Constraints (8.5) and (8.6) represent flow conservation constraints where an aircraft has always a subsequent flight with the exception of its last flight valid in the recovery window.

The fact that only one is the first (last) flight for a given aircraft is assured by constraints (8.7) and (8.8). These constraints are used to identify the earliest flight and the latest flight in the recovery window for every aircraft. These constraints are complemented by constraints (8.9) and (8.10): one flight can be the first (last) flight of an aircraft only if this flight is assigned to this aircraft.

Capacity of Airports

The capacity constraints for airports are formulated as follows:

$$End_j \geq Lim_t^{LB} Hour_{jt}^A \qquad \forall j \in F; t \in T \qquad (8.11)$$

$$End_j \leq Lim_t^{UB} Hour_{jt}^A + M_2(1 - Hour_{jt}^A) \qquad \forall j \in F; t \in T \qquad (8.12)$$

$$\sum_{j \in FlightAirport_p^A} Hour_{jt}^A \leq CapA_{pt}^P \qquad \forall p \in P; t \in T \qquad (8.13)$$

$$Start_j \geq Lim_t^{LB} Hour_{jt}^D \qquad \forall j \in F; t \in T \qquad (8.14)$$

$$Start_j \leq Lim_t^{UB} Hour_{jt}^D + M_2(1 - Hour_{jt}^D) \qquad \forall j \in F; t \in T \qquad (8.15)$$

$$\sum_{j \in FlightAirport_p^D} Hour_{jt}^D \leq CapD_{pt}^P \qquad \forall p \in P; t \in T \qquad (8.16)$$

$$\sum_{t \in T} Hour_{jt}^A + CF_j = 1 \qquad \forall j \in F \qquad (8.17)$$

$$\sum_{t \in T} Hour_{jt}^D + CF_j = 1 \qquad \forall j \in F \qquad (8.18)$$

Constraints (8.11) and (8.12) define the value of the decision variables $Hour_{jt}^A, \forall j \in F; t \in T$. This variable is equal to 1 if the arrival time is between Lim_t^{LB} and Lim_t^{UB} and 0 otherwise. These parameters split the recovery window in several time slots t which delimit the intervals of limited capacity at airports. Let M_2 be a large constant chosen as small as possible. Constraints (8.13) state the maximal number of arrival allowed during time slot t for airport p, for all $p \in P$ and $t \in T$. Similarly, constraints (8.14), (8.15), (8.16) restrict the number of departures for the time slot t at the airport p, for all $p \in P$ and $t \in T$. Finally, constraints (8.17) and (8.18) indicate that variables $Hour_{jt}^A$ and $Hour_{jt}^D$ can take value 1 only if flight $j \in F$ is not canceled.

Passengers and Aircraft Seating Capacity

The constraints associated to passengers and aircraft seating capacity are formulated as follows:

$$\sum_{k \in CompI_j^F} n_k IFC_{kjm} \leq CapP_{im}^A AF_{ij} + M_3(1 - AF_{ij}) \quad \forall i \in A; j \in CompF_i^A; m \in C \quad (8.19)$$

$$\sum_{\substack{j \in CompF_k^I \\ m \in C}} IFC_{kjm} + CI_k = 1 \quad \forall k \in DirIt \quad (8.20)$$

$$\sum_{\substack{j \in CompFF_k^I \\ m \in C}} IFC_{kjm} + CI_k = 1 \quad \forall k \in ConIt \quad (8.21)$$

$$\sum_{\substack{j \in CompLF_k^I \\ m \in C}} IFC_{kjm} + CI_k = 1 \quad \forall k \in ConIt \quad (8.22)$$

$$\sum_{m \in C} IFC_{kjm} - \sum_{m \in C} IFC_{k\hat{j}m} = 0 \quad \forall k \in ConIt; (j, \hat{j}) \in Conn_k^I \quad (8.23)$$

$$Start_{\hat{j}} \geq End_j + CT(\rho_{kj} + \rho_{k\hat{j}} - 1) - M_4(2 - \rho_{kj} - \rho_{k\hat{j}}) \quad \forall k \in ConIt; (j, \hat{j}) \in Conn_k^I \quad (8.24)$$

$$Delay_{kj}^I \geq End_j - End_k^{I0} - M_4(1 - \rho_{kj}) \quad \forall k \in I; j \in CompLF_k^I \quad (8.25)$$

$$IFC_{kjm} \leq (1 - CF_j) \quad \forall k \in I; j \in CompLF_k^I; m \in C \quad (8.26)$$

$$IFC_{kjm} \leq \rho_{kj} \quad \forall k \in I; j \in CompLF_k^I \quad (8.27)$$

The first set of constraints (8.19) models aircraft seating capacity, $CapP_{im}^A$, for aircraft $i \in A$ and cabin type $m \in C$. Note that these constraints are "activated" only when flight j is served by aircraft i. Let M_3 be a large constant. For this constraint a good value is $M_3 = n_k$.

For passengers taking a direct flight (set $DirIt$), constraints (8.20) are added to ensure that only one flight will be taken, otherwise the trip of the passenger must be canceled. On the other hand, constraints (8.21), (8.22) are needed for passengers that can take several flights with connections (set $ConIt$). Additionally, constraints (8.23) guarantee the flow conservation of passengers in airports when they have a connection.

For passengers taking several flights with connections, a minimal connection time is assured by constraints (8.24) defined by parameter CT. In other words, these constraints state that passengers k are allowed to take connection $(j, \hat{j})$ only if the difference between the departure time of flight $\hat{j}$ and the arrival time of flight j is greater than or equal to CT. Let M_4 be a large constant.

Constraints (8.25) added to the model for computing the delay of passengers. This value is very important to calculate some terms in the objective function. It should be noted that these constraints are defined only for the last flight of passengers k.

The last set of constraints in this subsection are (8.26) and (8.27). They define a valid relationship between variables IFC_{kjm}, CF_j, and ρ_{kj}. That is, a passenger can be assigned to a flight-cabin (IFC_{kjm}) only if this flight is not canceled (CF_j). Similarly, a passenger can be assigned to a flight-cabin only if he (she) is also assigned to this flight.

Costs

Costs of the objective function are computed as follows:

$$CostBadPosition = \sum_{\substack{i \in A \\ j \in CompF_i^A}} c_{ij}^{BP} LF_{ij} \tag{8.28}$$

$$CostDown = \sum_{\substack{k \in I \\ j \in CompF_k^I; m \in C}} c_{ijm}^{DW} n_k IFC_{kjm} \tag{8.29}$$

$$CostCanceled_{Pax} = \sum_{k \in I} c_k^{CP} n_k CI_k \tag{8.30}$$

$$CostCanceled_{Legal} = \sum_{k \in I} c_k^{CL} n_k CI_k \tag{8.31}$$

$$CostDelay_{Pax} = \sum_{k \in I} c_k^{DP} n_k Delay_k^I \tag{8.32}$$

$$CostDelay_{Legal} = \sum_{k \in I} c_k^{DL} n_k Delay_k^I \tag{8.33}$$

$$CostCancelFlights = \sum_{j \in F} c_j^{CanF} CF_j \tag{8.34}$$

Constraint (8.28) calculates the penalization cost in the case of non-compliant location of aircraft at the end of the recovery window. To model this constraint we consider the last flight of the aircraft and compare the destination of this flight to the expected location of the aircraft, where c_{ij}^{BP} is the unitary cost for bad position, if the last flight of aircraft i is flight j considering the family, the model, and the configuration of the aircraft.

Let c_{kjm}^{DW} be the unitary cost for downgrading, if one passenger of itinerary k take flight j in cabin m. Thus, constraint (8.29) is the penalization in the case of reaccommodation of a passenger. For example, if a passenger use his original cabin class then $c_{kjm}^{DW} = 0$.

For cancelation and delay of trips, it is necessary to differentiate $CostCanceled_{Pax}$ vs. $CostCanceled_{Legal}$ and $CostDelay_{Pax}$ vs. $CostDelay_{Legal}$. On the one hand, $CostCanceled_{Pax}$ and $CostDelay_{Pax}$ are the costs associated with the passenger viewpoint for cancelation

and delays respectively. On the other hand $CostCanceled_{Legal}$ and $CostCanceled_{Pax}$ are the airline cost associated with the cancelation and delays of the trip. These values are calculated using information such as the ticket price and legal financial compensations.

Domains of the Variables

The last part of this MIP formulation is the definition of the domains of the variables:

$$Start_j, End_j \geq 0 \qquad \forall j \in F \tag{8.35}$$
$$Delay^I_{kj} \geq 0 \qquad \forall k \in I, j \in CompLF^I_k \tag{8.36}$$
$$AF_{ij}, FF_{ij}, LF_{ij} \in \{0,1\} \qquad \forall i \in A; j \in CompF^A_i \tag{8.37}$$
$$CF_j \in \{0,1\} \qquad \forall j \in F \tag{8.38}$$
$$IFC_{kjm} \in \{0,1\} \qquad \forall k \in I; j \in CompF^I_k; m \in C \tag{8.39}$$
$$CI_k \in \{0,1\} \qquad \forall k \in I \tag{8.40}$$
$$\omega_{ij\hat{j}} \in \{0,1\} \qquad \forall i \in A; j \in CompF^A_i; \hat{j} \in CompNF^F_j \tag{8.41}$$
$$\rho_{kj} \in \{0,1\} \qquad \forall k \in I; j \in CompF^I_k \tag{8.42}$$
$$Hour^D_{jt}, Hour^A_{jt} \in \{0,1\} \qquad \forall j \in F; t \in T \tag{8.43}$$

And the auxiliary cost variables:

$$CostBadPosition, CostDown \geq 0 \tag{8.44}$$
$$ConstCanceled_{Pax}, CostCanceled_{Legal} \geq 0 \tag{8.45}$$
$$CostDelay_{Pax}, CostDelay_{Legal}, CostCancelFlights \geq 0 \tag{8.46}$$

The complete MIP formulation is then composed of objective function (8.1) and constraints (8.2) - (8.34) and the domain of the variables (8.35) - (8.46). It should be noted that only time variables ($Start_j, End_j, Delay^I_{kj}$) and cost variables are real non-negative decision variables while the rest are defined to be binary variables. This is the main reason why using a standard MIP solver is possible only for very small instances of the problem. In the next section we present an algorithm enabling to deal with this problem for real-size instances.

8.4 Algorithm

Our algorithm builds a restricted MIP model limited to the feasible region of expected high quality solutions. This is possible by adding constraints limiting the changes in comparison with the original schedule. In addition, an original solution method called SAPI (Statistical Analysis of Propagation of Incidents) is carried out to solve the remaining model.

Figure 8.2 shows a flow diagram explaining the steps of the process. The principal function is called *Main*. The inputs are the instance files and the optimization parameters such as the maximum running time.

After initializing and reading files, the algorithm creates a collection of virtual flights to model special events: maintenances and disruptions of aircraft (function "Create Virtual Flights: Maintenances and Disruptions"). The goal is to forbid, during a limited period of time, any assignment of aircraft to real flights, in other words, airplanes are blocked to be unavailable during the event.

The next step is "Split Passengers". Data files contain group of passengers called "itinerary". The function tries to split these passengers in order to obtain better solutions. The advantage of splitting itineraries is that allowing passengers following different routes helps to use better the aircraft capacity. Increasing the number of itineraries implies to add integer variables to the model and so increases the complexity. For this reason, this function studies the size of the instances before trying any splitting of passengers. Consequently, there is a trade-off between quality of the solution and performance of the method.

Another important function is "Create New Flights". We investigate the possibility of creating new flights to minimize the impact of disruptions. We first tried to accept any possible new flight considering all the combination of airports given in the data sets. However, this was not efficient because many pairs of airports do not have enough passengers to construct a profitable flight leg between them. After analyzing some other possibilities, the last version of our system adds flight using only original routes with few, perturbed, or canceled flights. In other words, this function duplicates flights when it seems to be necessary. Additionally, when a flight is generated, another "twin" flight is also added to the model in the opposite direction (round-trip).

Once all data have been preprocessed and parameters have been defined, it is necessary to calculate several sets that will define the degrees of freedom to find a solution. Note that the more degree of freedom, the more complexity. Thus, the purpose of function "Reduction of Sets" is to build and reduce the size of the following sets:

$$CompF_i^A, CompFF_i^A, CompAP_i^A; \forall i \in A$$

$$CompNF_j^F, CompA_j^F, CompI_j^F; \forall j \in F$$

$$CompF_k^I, CompAP_k^I, Conn_k^I; \forall k \in I$$

$$CompFF_k^I, CompLF_k^I; \forall k \in ConIt$$

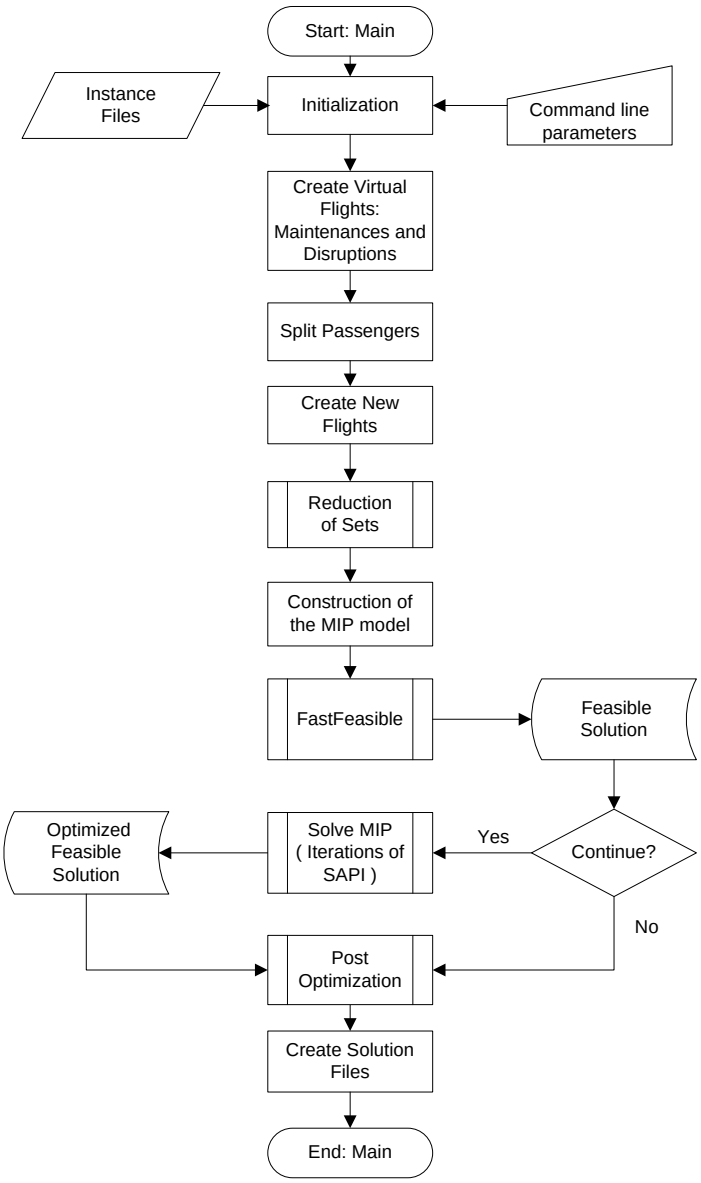


Figure 8.2: *Flow diagram of the solution method.*

To build these sets, our algorithm considers real constraints and other extra conditions. For example two real constraints are: an original planned aircraft can be replaced by another one if they are of the same family and the range is respected, *i.e.*, the range of the new aircraft has to be greater than or equal to the distance from the origin to the destination of the flight. The first sets to be constructed are $CompF_i^A$ and $CompF_k^I$ because the other ones are derived from these two.

The function "Reduction of Sets" is composed of two main *for* loops. The first one searches for compatible aircraft to flights by using distance matrices computed with the

Floyd-Warshall Algorithm (Cormen et al., 1990). For example, if an aircraft i can arrive without delay (or a maximum tolerated delay) from its original initial position to the origin airport of the flight j, then i is added to the set of compatible aircraft $CompA_j^F$. Other conditions are also utilized, but they are auto-activated when the size of the instance is considerably large, *e.g.*, the algorithm considers not only limiting flights to the aircraft of the same family, but also the same configuration and model.

It is reasonable to think that only a subset of flights is appropriated to be considered for a given group of passengers. For this reason a similar procedure is performed to define good candidates of flights for each passenger. This function evaluates different conditions that flights have to respect for being compatible with a particular group of passengers. For example, flights that exceed a distance limit from the original trip are discarded. As other parameters, this distance limit is automatically adjusted depending on the size of the instance.

At this point, we have all the elements to build the mathematical model. The step called "Construction of the MIP model" translates all data and calculated special sets into variables and constraints of a Mixed Integer Programming (MIP) formulation. Once the model is built, a procedure called "FastFeasible" tries to find a feasible solution quickly. This function solves a subproblem where all changes in aircraft and passenger assignments are not allowed, fixing many integer variables by changing their bounds. This procedure only decides if it is better to cancel flights and passengers or to keep the original schedule. Because many integer variables are fixed, the remaining subproblem is much easier to solve. Nevertheless, the quality of this solution is not good enough and delays are propagated since changing the assignment of aircraft and passengers are not allowed. For that reason, the aim of the next steps is to improve this first feasible solution as much as possible within a reasonable predefined processing time.

Depending on the available remaining processing time, the algorithm iterates further or is stopped. If it is possible to continue, the algorithm performs the step called "Solve MIP (Iterations of SAPI)" based on the function SAPI (see Figure 8.3). This function is fully described in the next section. On the other hand, if there is not enough processing time for SAPI, then "Post Optimization" is carried out.

The method concludes with a postprocessing procedure called "Post Optimization". The purpose of this function is to improve the final result given by SAPI. To achieve this goal, it analyzes all the current canceled passengers and tries to re-assign them in any combination of flights that will take them to their destinations. This strategy is justified by the fact that the cost of cancelation of passengers is much more expensive than the cost of delays and filling the remaining empty seats has a marginal cost of zero.

The very last step, "Create Solution Files", generates the two final files: a new file for itineraries and a new file for rotations. The objective is to transform the final value of decision variables, most of them binary, to practical information like dates, aircraft, and passengers.

8.5 Statistical Analysis of Propagation of Incidents (SAPI)

This section describes the major step of the algorithm: Statistical Analysis of Propagation of Incidents (SAPI). This methodology was originally developed to reschedule trains under disrupted operations (see Chapter 5). Nevertheless its basic principle can be also exploited for the solution of this problem:

The probability that an event is affected by a perturbation depends on some factors.

In this special case, the relevant *event* is a *flight*. Thus, if we are able to calculate these probabilities, it will help solving the problem. In particular, we propose to use a logistic regression to estimate these probabilities that will be used to fix some integer variables. In this way, the preprocessing function implemented in many MIP solvers will eliminate them of the model, as a consequence the remaining MIP will be much easier to solve.

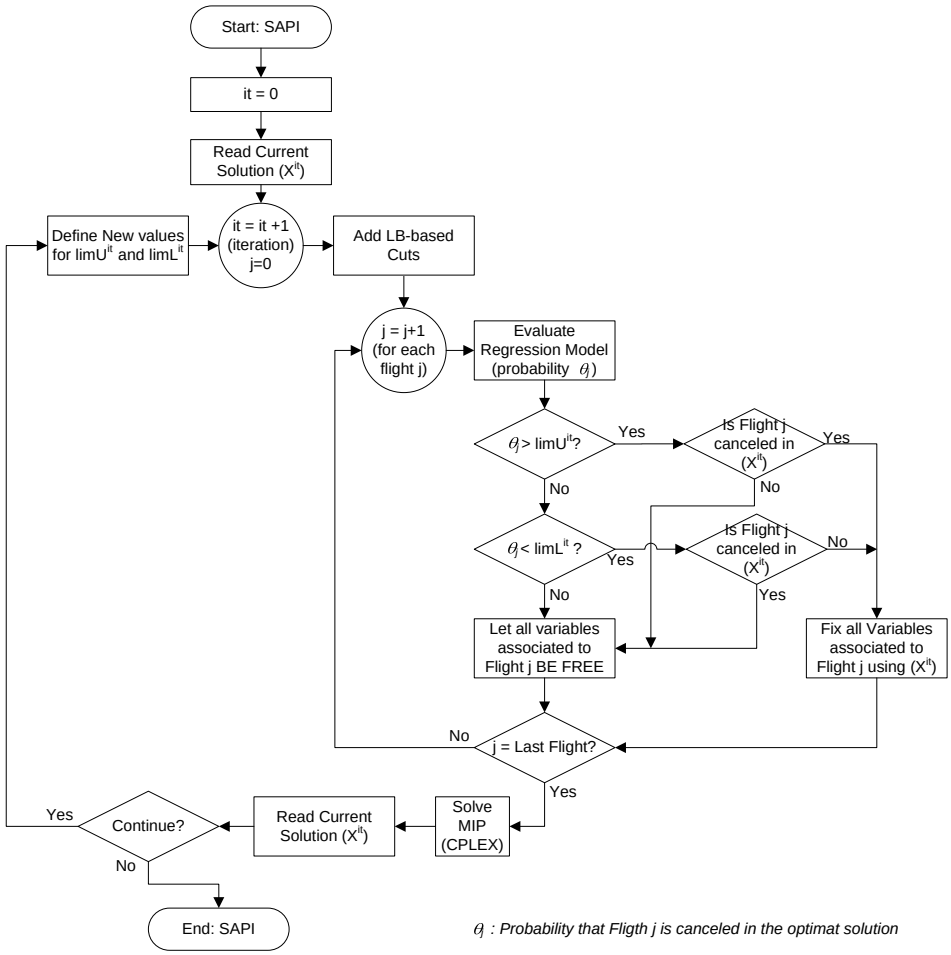


Figure 8.3: *Flow diagram of function SAPI (Statistical Analysis of Propagation of Incidents).*

Figure 8.3 presents the flowchart of this function. The first steps are to initialize the iteration counter $it = 0$ and to read the current feasible solution calculated by function "FastFeasible". The next steps belong to a *while loop* that allows iterations to be repeatedly performed until the ending criteria is evaluated as `true`, for example using a maximum allowed running time. Each iteration can be divided in three components: adding LB-based cuts, fixing variables, and solving the remaining MIP.

8.5.1 Adding LB-based cuts

The objective is to explore solution neighborhoods defined by local branching cuts affecting decision variables CF_j[4]. Thereby, the number of changes at each iteration is limited by the local branching cut defined as follows:

$$\sum_{j \in \lambda^{it}} CF_j \leq LB \tag{8.47}$$

where:

λ^{it} : Set of flights that are **not** canceled in the current incumbent solution, says at iteration (it).

LB : local branching parameter.

There is a clear tradeoff between the computational effort and the quality of the solution depending on the value of LB. If the value of LB is very small, it is easier to calculate the new solution because only few changes are allowed in the current iteration, however good solutions could be discarded. For this reason, the parameter LB is automatically defined depending on the characteristic of each instance. Thereby, for small instances this parameter will have a relative large value because the remaining MIP model will be easier to solve. On the other hand, for large (hard) instances the algorithm will try small values of LB.

8.5.2 Fixing variables

The main aspect of SAPI is the use of a statistical analysis to fix integer variables. This analysis is performed evaluating a regression model described in detail in Section 8.6.

The use of these probabilities is quite different than for our previous application of SAPI in railways (see Chapter 5). Here, at each iteration, the value of the probability of canceling flight j, given by θ_j, is compared with two constants $limL^{it}$ and $limU^{it}$ where

[4]Local branching cuts were originally proposed by Fischetti and Lodi as an exact solution technique based on linear inequalities (Fischetti and Lodi, 2003)

$limL^{it} < limU^{it}$. These values are automatically adjusted depending on the processing time taken by the last iteration. There are two cases for fixing variables.

Case 1: if flight j is canceled in the current solution (X^{it}) and $\theta_j > limU^{it}$, all the decision variables associated to this flight are automatically fixed using the value of the current solution. Let Λ^{it} be the set of all flights to be automatically canceled at iteration it:

$$\Lambda^{it} \triangleq \{j; \theta_j > limU^{it} \wedge CF_j^{it-1} = 1\} \tag{8.48}$$

where CF_j^{it-1} is the value of variable CF_j at the end of iteration $it-1$, and $\triangleq$ means "is defined as".

Then, the variables to fix at iteration it are:

$$AF_{ij} = 0 \qquad \forall i \in A \wedge j \in CompF_i^A \cup \Lambda^{it} \tag{8.49}$$
$$FF_{ij} = 0 \qquad \forall i \in A \wedge j \in CompF_i^A \cup \Lambda^{it} \tag{8.50}$$
$$LF_{ij} = 0 \qquad \forall i \in A \wedge j \in CompF_i^A \cup \Lambda^{it} \tag{8.51}$$
$$CF_j = 1 \qquad \forall j \in \Lambda^{it} \tag{8.52}$$
$$IFC_{kjm} = 0 \qquad \forall k \in I \wedge j \in CompF_k^I \cup \Lambda^{it} \wedge m \in C \tag{8.53}$$
$$\omega_{ij\hat{j}} = 0 \qquad \forall i \in A \wedge j \in CompF_i^A \wedge \hat{j} \in CompNF_j^F \wedge (j \in \Lambda^{it} \vee \hat{j} \in \Lambda^{it}) \tag{8.54}$$
$$\rho_{kj} = 0 \qquad \forall k \in I \wedge j \in CompF_k^I \cup \Lambda^{it} \tag{8.55}$$
$$Hour_{jt}^D = 0 \qquad \forall j \in \Lambda^{it} \wedge t \in T \tag{8.56}$$
$$Hour_{jt}^A = 0 \qquad \forall j \in \Lambda^{it} \wedge t \in T \tag{8.57}$$

Case 2: if flight j is not canceled in the current solution (X^{it}) and $\theta_j \leq limL^{it}$, all the decision variables associated to this flight are automatically fixed using the value of the current solution. Let Π^{it} be the set of all flights that will not be canceled at iteration it:

$$\Pi^{it} \triangleq \{j; \theta_j < limL^{it} \wedge CF_j^{it-1} = 0\} \tag{8.58}$$

Then, the variables to fix at iteration it are:

$$AF_{ij} = AF^0_{ij} \qquad \forall i \in A \wedge j \in CompF^A_i \cup \Pi^{it} \quad (8.59)$$

$$FF_{ij} = FF^0_{ij} \qquad \forall i \in A \wedge j \in CompF^A_i \cup \Pi^{it} \quad (8.60)$$

$$LF_{ij} = LF^0_{ij} \qquad \forall i \in A \wedge j \in CompF^A_i \cup \Pi^{it} \quad (8.61)$$

$$CF_j = 0 \qquad \forall j \in \Pi^{it} \quad (8.62)$$

$$IFC_{kjm} = IFC^0_{kjm} \qquad \forall k \in I \wedge j \in CompF^I_k \cup \Pi^{it} \wedge m \in C \quad (8.63)$$

$$\omega_{ij\hat{j}} = \omega^0_{ij\hat{j}} \qquad \forall i \in A \wedge j \in CompF^A_i \wedge \hat{j} \in CompNF^F_j \wedge (j \in \Pi^{it} \vee \hat{j} \in \Pi^{it}) \quad (8.64)$$

$$\rho_{kj} = \rho^0_{kj} \qquad \forall k \in I \wedge j \in CompF^I_k \cup \Pi^{it} \quad (8.65)$$

$$Hour^D_{jt} = Hour^{D0}_{jt} \qquad \forall j \in \Pi^{it} \wedge t \in T \quad (8.66)$$

$$Hour^A_{jt} = Hour^{A0}_{jt} \qquad \forall j \in \Pi^{it} \wedge t \in T \quad (8.67)$$

where:

x^0 is the value of any decision variable x in the current incumbent solution, *i.e.*, at the end of iteration $it - 1$.

8.5.3 Solving the remaining MIP

The last step is to add the cut (8.47) and all constraints (8.49) to (8.67) to the original MIP model. After that, a standard MIP solver is called to solve the remaining MIP. Because of the new constraints, the solution of this reduced MIP is much easier than the original one.

8.6 Logistic Regression

Logistic regression analysis allows estimating multiple regression models when the response being modeled is dichotomous and can be scored 0 or 1. Particularly, our model has two possible outcomes: 1, if the variable associated to an event is affected by the incidents and 0 otherwise. Thus, the dependent variable is dichotomous, that is, it can take the value 1 with a probability of success θ, or the value 0 with probability of failure $1 - \theta$. On the other hand, independent (predictor) variables in logistic regression can take any form and makes no assumption about the distribution of the independent variables.

We reduce the combinatory for the variables that model cancelation of flights (variables $CF_j, \forall j \in F$). These variables are strongly linked to many other integer variables and reducing the number of them helps fixing other variables. For example, if a flight is canceled, neither passengers nor aircraft can be assigned to that flight. As a result, the constraints associated with airport capacity will also benefit from this reduction. For

the rest of the integer variables, it was not possible to find predictor variables statistically significant.

The regression model is defined by the following equation:

$$\theta_j = \frac{exp(\eta)}{1 + exp(\eta)} \quad \forall j \in F \tag{8.68}$$

Where:

$$\eta = \alpha + \beta^1 \rho_j^1 + \beta^2 \rho_j^2 + \beta^3 \rho_j^3 + \varepsilon_j$$

θ_j : Probability that variable $CF_j = 1$ in the optimal solution, where $j \in F$.
α : Constant of the regression equation.
β^i : Regression coefficient of the predictor variable i.
ε_j : Error term.

The predictor variables are:

ρ_j^1 This predictor variable has only two values $\in \{0, 1\}$. $\rho_j^1 = 1$ if the original aircraft assigned to flight j has canceled flights before the beginning of the recovery time window, and $\rho_j^1 = 0$ otherwise.

ρ_j^2 This predictor variable is also binary, that is $\rho_j^2 \in \{0, 1\}$. It has a value equals to 1 if the original aircraft assigned to flight j has delayed flights before the beginning of the recovery time window, and 0 otherwise.

ρ_j^3 This predictor variable is a real number $\in [0, 1]$ that represents the relative time of flight j. Flights closer to the beginning of the recovery time window are expected to be more affected by disruptions. In contrast, latter flights have smaller probabilities to be affected because of the robustness of the original schedule, *i.e.*, robust schedules isolate disruptions and reduces the downstream impact. It should be noted that a relative value is preferred rather than absolute ones. The reason is quite simple: the goal is that the values of the regression parameters remain valid for any kind of instance and not only for the one used as statistical sample.

The coefficients of the regression model: α, β^1, β^2, and β^3, need to be estimated. As other regression models, they could be calculated by maximum likelihood[5]. A statistical sample is required to make inference of these coefficients. To achieve this objective we use a small instance of the problem and solve the MIP optimally with different

[5]The interested readers can find in (Aldrich, 1997) a good introduction to this method

kinds of incidents. Because it is a training process, the processing time has not the same importance as solving instances with real incidents. It is important to remark that these estimations (of parameters) can be improved gradually using continuously this method, says daily. This is possible because we increase the size of the sample, gaining more and more information each time we use SAPI.

Using this set of optimal solutions, we prepared a statistical sample with the value of the decision variables ($CF_j, \forall j \in F$) and the value of the predictor variables (ρ). This sample is then processed by a statistical software, in particular, we use Statgraphics 4[6]. In order to demonstrate that the quality of the coefficients does not depend on the instance, we decide to use the same coefficients for all instances.

Figure 8.4 shows the Analysis of Deviance for the regression model. For the reason that the P-value for the model in the analysis of deviance is less than 0.01, there is a statistically significant relationship between the variables at the 99% confidence level. In addition, the P-value for the residuals is greater than 0.10, indicating that the model is not significantly worse than the best possible model for this data at the 90% or higher confidence level.

Source	Deviance	Df	P-Value
Model	244.50	3	0
Residual	1386.34	4995	1
Total (corr.)	1630.84	4998	

Figure 8.4: *Analysis of Deviance: Percentage of deviance explained by model = 14.99*

The percentage of deviance explained by the model is equal to 14.99 (see Figure 8.4). This statistic is similar to the usual R^2 statistic. That is, 14.99% of the variability of the value of variables $CF_j, \forall j \in F$ in the optimal solution can be explained by this regression model without solving the problem. Although this is not a great value, using SAPI the results are assured better than a random selection of variables $CF_j, \forall j \in F$.

Figure 8.5 shows estimated values for the regression parameters. Notice that the highest P-value for the likelihood ratio tests is 0.0070, belonging to β^2. Because the P-value is less than 0.01, that term is statistically significant at the 99% confidence level. Consequently, we do not consider removing any variables from the model.

Parameter	Estimate	P-Value
α	-100.60	-
β^1	23.49	0.0000
β^2	18.03	0.0070
β^3	17.41	0.0000

Figure 8.5: *Estimated Regression Model (Maximum Likelihood)*

To assure that implementing SAPI has a positive impact in final results, we inves-

[6]http://www.statgraphics.com/

tigated the performance of the algorithm for several instances. We observed that all of them follow the same behavior. Figure 8.6 shows an example to illustrate the impact of SAPI in the convergence to the optimal value. This chart exhibits two different versions of the algorithm. The first version applies SAPI with a logistic regression for selecting the variables to fix. The second version selects the variables randomly. It is possible to appreciate that the value of the objective function using SAPI and logistic regression improves faster than a random selection of variables. As a consequence, given a limited processing time, it will return a better solution.

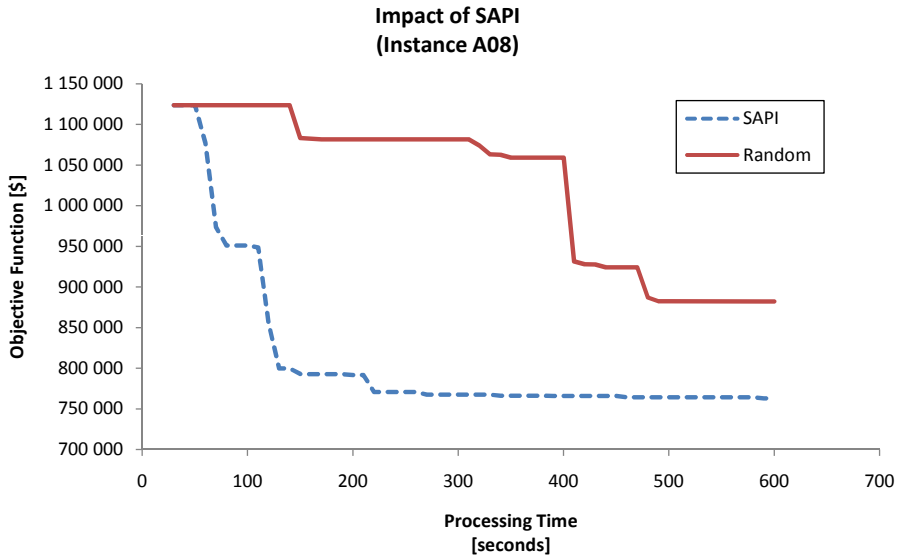


Figure 8.6: *Impact of SAPI. This graph shows two different version of the algorithm: a) SAPI with a logistic regression b) random selection of flights*

8.7 Results on the ROADEF challenge 2009 instances

In this section we present the results obtained in the ROADEF challenge 2009 in order to compare this method with others under the same conditions, *i.e.*, the same machine, the same processing time, and the same cost/solution checkers. All programs provided by the competition participants were evaluated on the target computer at AMADEUS[7]. The organization ran the algorithms on a machine with AMD Turion 64 x2 processor and 2 GB RAM with Windows XP (32 bit) or Linux (64 bit). Several commercial solvers were allowed: IBM ILOG Cplex 11.x, IBM ILOG CP Optimizer 1.x, Dash Optimization Xpress-MP release 2007B, and Artelys Kalis 2007B. In particular, this algorithm (SAPI) was implemented using Microsoft Visual Studio 2005 (C♯) and ILOG CPLEX 11.1 as a standard MIP solver.

[7]http://www.amadeus.com

- **"A" Instances (Qualification):** Theses instances were used in the qualification phase of the competition. All of them are composed of 608 flights, 35 airports, and 85 aircraft, however they differ in the number of passengers (from 36010 to 93424 people), the type of the disruption (aircraft, airport, flight, or a combination of them) and the length of the recovery time window (from 12 to 52 hours). It is important to remark that we compared our solutions on "A" Instances with the solution provided for the qualification phase because the results of the last version of other systems are not known for those instances. The solution and cost checkers were provided by the organization and the machine was a PC Intel Core Duo T5500 1.66 GHz 2 GB RAM over Windows Vista Operation System, very similar to that used by AMADEUS.

- **"B" Instances:** They are composed of 1422 flights, 44 airports, and 255 aircraft. The number of passengers varies from 207193 to 263574. Several types of disruption are also considered: aircraft, airports, flights, and a combination of them. The length of the recovery time window fluctuates from 40 to 52 hours. The results were evaluated on the target computer at AMADEUS.

- **"X" Instances:** These instances were published after the competition and were partially employed for calculating the final normalized score. They combine variations of "A", "B", and other new much larger instances. They are composed from 608 to 2178 flights; the number of aircraft varies from 85 to 618; and the number of passengers from 36010 to 700683. The same types of disruptions are considered, that is: aircraft, airports, flights, and a combination of them. The length of the recovery time window fluctuates from 14 to 78 hours. The results were evaluated on the target computer at AMADEUS.

It is important to remark that the length of the recovery window defines the number of aircraft rotations because the same flight code may be repeated in different days. Hence, this is one of the main factors influencing the complexity of the instance.

The normalized score of Method M on Instance I is given by the equation:

$$Score(M, I) = \frac{W(I) - z(M, I)}{W(I) - B(I)} \tag{8.69}$$

where:

$z(M, I)$: objective function value obtained by Method M on Instance I

$W(I)$: worst objective function value found on Instance I, considering all competitors and this method.

$B(I)$: best objective function value found on Instance I, considering all competitor and this method.

The organization of the competition decided that if a method does not provide a solution or if the returned solution is infeasible, its score is set to two times $W(I)$.

Figures 8.7, 8.8, and 8.9 present the final results for our algorithm (SAPI) compared to other systems. Finally, Figure 8.10 shows the final ranking of the competition. The final average score is computed using instances B01,...,B10 and instances XA1,...,XB4[8].

	Set A (Qualification Results)										
	A01	A02	A03	A04	A05	A06	A07	A08	A09	A10	Avg Score
SAPI	**1.00**	**0.96**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.996**
(Q) Bisaillon- Cordeau- Laporte- Pasin	0.99	0.96	0.96	0.99	0.98	1.00	0.96	0.96	1.00	0.99	0.979
(Q) Hanafi- Wilbaut- Mansi	0.99	0.95	0.99	0.96	0.93	0.99	0.93	0.99	0.95	0.91	0.960
(Q) Acuna-Agost - Michelon- Feillet- Gueye	0.98	0.68	0.76	1.00	0.83	0.99	0.78	0.92	1.00	0.88	0.882
(Q) Darlay- Kronek- Schrenk- Zaourar	0.91	0.24	0.86	0.94	0.93	0.95	0.71	0.94	0.96	0.93	0.839
(Q) Jozefowiez- Mancel- Mora-Camino	0.96	0.67	0.84	0.80	0.86	0.99	0.66	0.92	0.71	0.86	0.828
(Q) Eggermont- Firat- Hurkens- Modelski	0.74	0.24	0.35	0.88	0.74	0.80	0.51	0.87	0.94	0.82	0.688
(Q) Demassey- Jussien- Lorca- Menana-Quillet- Richaud	0.96	0.49	0.80	0.61	0.54	0.98	0.65	0.89	0.59	0.60	0.712
(Q) Dickson- Smith- Li	0.65	0.00	0.26	0.81	0.60	0.73	0.47	0.83	0.88	0.70	0.594
(Q) Estellon- Gardi- Nouioua	0.92	0.06	0.09	0.86	0.53	0.96	0.37	0.76	0.90	0.60	0.606
(Q) Eggenberg- Salani	0.00	0.29	0.00	0.50	0.51	0.00	0.00	0.51	0.50	0.52	0.284
(Q) Peekstok- Kuipers	0.99	1.00	0.99	0.00	0.00	0.26	0.54	0.00	0.00	0.00	0.378

Figure 8.7: *Results on "A" instances. This method (SAPI) is compared with the results obtained during the qualification phase of the ROADEF challenge 2009. Note that "(Q) Acuna-Agost - Michelon - Feillet - Gueye" is an earlier version of the method described in this work.*

It should be noted that this method is particularly good for mid-size instances ("A" and "B" instances) compared with the other participants. This is explained because our method is based on an integrated model and can find better "hidden" solutions. Other methods do not consider these solutions because they all solve the problem by decomposing it into several parts. This advantage is less effective for larger instances because of the automatic selection of reduction parameters, *i.e.*, larger instances implies a strong reduction and some of the "hidden" solutions will be discarded.

Figure 8.11 presents the number of solved instances for every method on set B and X. It also includes a comparative score analysis between SAPI and all the other methods individually. In order to use comparable data, we limit this analysis on common feasible instances, *i.e.*, the both methods were able to find feasible solutions. The results show that SAPI obtain very good results compared with all other systems individually.

[8] http://challenge.roadef.org/2009/resultats.en.htm

	Set B										
	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	Avg Score
SAPI	**0.99**	**0.95**	**0.99**	**0.99**	**0.95**	**0.97**	**0.94**	**0.97**	**0.98**	**0.94**	**0.966**
Bisaillon- Cordeau- Laporte- Pasin	1.00	0.99	1.00	1.00	0.93	0.99	0.99	0.99	0.97	0.80	0.966
Hanafi- Wilbaut- Mansi- Clautiaux	0.89	0.71	0.90	0.90	0.96	0.91	0.73	0.89	0.90	1.00	0.878
Eggermont- Firat- Hurkens- Modelski	0.95	0.89	0.95	0.95	0.74	0.92	0.83	0.92	0.91	0.50	0.855
Darlay- Kronek- Schrenk- Zaourar	0.96	0.82	0.97	0.97	0.83	0.90	0.80	0.90	0.89	0.80	0.885
Peekstok- Kuipers	0.99	0.96	0.99	0.99	1.00	0.96	0.93	0.95	0.98	0.96	0.969
Jozefowiez- Mancel- Mora- Camino	1.00	1.00	1.00	1.00	0.82	1.00	1.00	1.00	1.00	0.73	0.954
Dickson- Smith- Li	0.79	0.52	0.81	0.81	0.52	0.74	0.55	0.77	0.74	0.53	0.676
Eggenberg- Salani	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.000

Figure 8.8: *Results on "B" instances.*

	Set X												
	XA01	XA02	XA03	XA04	XB01	XB02	XB03	XB04	X01	X02	X03	X04	Avg Score
SAPI	**0.99**	**0.99**	**0.98**	**0.75**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.309**
Bisaillon- Cordeau- Laporte- Pasin	0.95	0.98	0.93	0.86	1.00	0.97	1.00	0.91	1.00	1.00	1.00	1.00	0.967
Hanafi- Wilbaut- Mansi- Clautiaux	1.00	1.00	1.00	1.00	0.96	0.99	0.96	1.00	0.00	0.00	0.00	0.00	0.659
Eggermont- Firat- Hurkens- Modelski	0.93	0.90	0.90	0.71	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.368
Darlay- Kronek- Schrenk- Zaourar	0.98	0.00	0.97	0.00	0.98	0.93	0.00	0.00	0.00	0.00	0.00	0.00	0.322
Peekstok- Kuipers	1.00	0.97	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.247
Jozefowiez- Mancel- Mora- Camino	1.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.165
Dickson- Smith- Li	0.00	0.00	0.00	0.00	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.067
Eggenberg- Salani	0.51	0.52	0.51	0.00	0.51	0.53	0.52	0.58	0.00	0.00	0.00	0.00	0.307

Figure 8.9: *Results on "X" instances.*

In fact, only two of the other methods obtained better solutions with a difference in the score less than 1%.

Analyzing the results, it is also possible to appreciate that extremely large instances of set "X" are intractable for our method because of the large number of variables and memory limitation. Although the method is efficient, it is also necessary to load the complete MIP model and this may utilize too much memory.

Team	Category	Global rank (category rank)	Average score (%)
Bisaillon, Cordeau, Laporte, Pasin	Senior	1(1)	95.90
Hanafi, Wilbaut, Mansi, Clautiaux	Senior	2(2)	92.73
SAPI	**Senior**	**3(3)**	**74.26**
Eggermont, Firat, Hurkens, Modelski	Junior	4(1)	72.01
Darlay, Kronek, Schrenk, Zaourar	Junior	5(2)	70.62
Peekstok, Kuipers	Senior	6(4)	70.31
Jozefowiez, Mancel, Mora-Camino	Senior	7(5)	64.02
Dickson, Smith, Li	Junior	8(3)	42.02
Eggenberg, Salani	Junior	9(4)	20.48

Figure 8.10: *Final ranking of the ROADEF challenge 2009*

	# Solved Instances			Comparative Score		
	Set B	Set X	Total	# Common Instances	Score SAPI	Score Method
SAPI	**10**	**4**	**14**	-	-	-
Bisaillon- Cordeau- Laporte- Pasin	10	12	22	14	0.9548	**0.9562**
Hanafi- Wilbaut- Mansi- Clautiaux	10	8	18	14	**0.9548**	0.9128
Eggermont- Firat- Hurkens- Modelski	10	5	15	14	**0.9548**	0.8559
Darlay- Kronek- Schrenk- Zaourar	10	4	14	12	**0.9694**	0.8998
Peekstok- Kuipers	10	3	13	12	0.9694	**0.9713**
Jozefowiez- Mancel- Mora-Camino	10	2	12	12	**0.9694**	0.9603
Dickson- Smith- Li	10	1	11	10	**0.9661**	0.6762
Eggenberg- Salani	8	7	15	11	**0.9755**	0.1402

Figure 8.11: *Number of instances solved by the methods and a comparative analysis with the other methods individually. SAPI is compared to other methods individually. An instance is considered in this analysis only if the both methods find a feasible solution.*

8.8 Conclusions and Perspectives

We have proposed and analyzed computationally a method for finding solutions to the problem of rescheduling flights, aircraft and passenger simultaneously under disrupted operations. The approach uses a Mixed Integer Programming (MIP) formulation that includes all the aspects required in the ROADEF challenge 2009 (ROADEF, 2008). The model can be interpreted as two multi-commodity flow problems, one related to aircraft and the other one to passengers.

Nevertheless, state-of-the-art MIP solvers are not efficient enough to solve this problem because of the complexity of constraints, the number of integer variables, and the size of the instances. For this reason, we applied Statistical Analysis of Propagation of Incidents (SAPI), to emphasize the search in probable good part of the solution space. This method has been proven to be effective for rescheduling railway operations and this additional application shows that is also viable to solve other disruption management problems.

The contribution of SAPI is quite simple but effective. The idea is to study the probability that disruptions affect future tasks (flights) and use these probabilities to focus the search in affected areas while the rest is kept fixed to the original (unperturbed) schedule. This approach is then extended to an iterative procedure where the original (unperturbed) schedule is replaced by the solution of the last iteration.

We reported computational results on the instances proposed by the ROADEF challenge 2009. Our results were compared with the results given by the finalist teams of the competence. As it is possible to appreciate, our algorithm seems to be effective to solve this problem efficiently, obtaining results over the average of the finalist teams and having obtained the third position of the competition. The results are much better if only feasible instances are considered to compare the methods. In fact, the results showed that extremely large instances of set "X" are intractable for our method because of the large number of variables and memory limitation.

Future directions of research should actually address improvements for larger instances complementing this method with decomposition techniques. The other direction of future research is to include explicitly crew in the reparation procedure. Two different approaches are possible: (a) fully integrated or (b) separated formulation. It is expected that a full integrated formulation will be hard to solve because of the additional complexity. On the other hand, an iterative procedure that connects the two separated models seems to be more adequate.

Part IV

Conclusions and Appendices

Chapter 9

Conclusions and Perspectives

In this final chapter we draw the general conclusions recalling the main objectives of this Thesis: a) to study the Railway Rescheduling Problem (RRP); b) to develop complementary formulations for the problem; c) to find, describe, implement, and test new solution methods to solve the RRP; and d) to develop a new general method that may be reused for other rescheduling problems.

The first part of the Thesis responds to the first objective. We have deeply described the Railway Rescheduling Problem as the problem to build a new provisional timetable after disrupted operations including a temporary plan of tracks and platforms. After integrating Chapter 2 and 3, it is possible to conclude that the version of the RRP studied in this Thesis is a "horizontal integration" of two generic decision problems: "time tabling daily" and "detailed platform assignment daily". Using the proposed classification, we have also observed that this problem is strongly related to the short-term versions of "crew scheduling", "rolling stock", and "shunting". That is, the solution of the RRP has a direct impact to these other problems and *vice versa*. This is a very important aspect to be considered when developing real-life control systems, because looking for solutions of an isolate problem without considering the interactions could be inefficient and even inviable in practice.

The second main objective is the development of complementary formulations for the RRP. The second part of this Thesis is dedicated to fulfill this goal. We have presented two formulations: MIP and CP. Both models include many practical rules and constraints, which explains its relative complexity compared to other models presented in the literature. Indeed, it supports allocation of tracks (and platforms) connection between trains, bidirectional/multi-track lines and extra time for accelerating and braking. Result analysis have shown that the proposed MIP has permitted to develop very efficient solution methods, however it employs more memory than the CP because of the large number of binary variables needed to model the order of trains.

The objective of developing new solution methods for the RRP was also treated in the second part of the Thesis. We have developed four base solution methods: i) Right-Shift Rescheduling, ii) MIP-based local search, iii) Statistical Analysis of Propagation

of Incidents (SAPI), and iv) a CP-based approach. Most of the methods have several versions by extending them to iterative approaches. Note that right-shift was not originally presented in this document. In the same way, the MIP-based local search method was inspired on the well known *local branching cuts*. The originality in this Thesis concerning these both methods is the way how they were adapted to this problem taking into account precious information about the original (unperturbed) schedule. To the best of our knowledge, all the other solution methods are proposed for the first time in this Thesis[1].

The proposed solution methods were presented in an incremental order. Consequently, right-shift rescheduling was the first and constructs a feasible solution by keeping the same order of trains and avoiding changes of tracks and unplanned stops. This procedure is carried out by fixing integer decision variables. The resulting model becomes easier to solve by standard solvers but the solutions are generally far away from the optimum. Afterward, we have developed the MIP-based local search method that uses the solution of right-shift rescheduling for calculating an initial solution. The big difference with traditional local search mechanisms is that neighborhoods are obtained through the introduction, in the MIP model, of linear inequalities called *local branching* cuts. These cuts are constructed using important information obtained from the original (unperturbed) schedule. As a result, this method obtains better solutions than right-shift rescheduling.

SAPI is the third method proposed in this thesis and, to our point of view, one of the major contributions of this Thesis. The central idea of SAPI is to try to catch, by a statistical analysis, the effects of disruption propagation on upcoming events. The effects of disruptions propagation are quantitatively evaluated by computing the probability that an event will be affected or not. With this information, some cuts may be added or some integer variables fixed. SAPI finds higher quality solutions and faster than the previous methods.

The last solution method proposed in this Thesis is a cooperative CP/MIP approach taking advantages of both methods: good quality solution for MIP and lower memory requirements for CP. In this method CP is used to model large problems without memory problems while a MIP-based method is performed in a subproblem. More specifically, the strategy of this method is twofold: domain filtering and an appropriate variable/values ordering.

From the literature it can be seen that several papers dealing with scheduling and rescheduling in rail transportation are available but comparing efficiency of these algorithms is very difficult because of the differences in the infrastructure representations and the lack of benchmark instances. That is the reason why, the last objective of this Thesis was to generalize one of our methods in such a way to be able to apply this generalization to another (similar) problem for which public instances were available.

A generalization of SAPI has been studied. One of the characteristics of any rescheduling problem is that a base/reference schedule is known, and the objective it is to create

[1]Some solution methods are also described in research papers derived from this Thesis.

a new provisional schedule as close as possible to the original plan with the aim to return quickly to normal operations. This main aspect is exploited by SAPI. The method focuses the computational effort evaluating changes in the "most" affected part of the schedule. This methodology is then applied for rescheduling flights and passenger simultaneously under disrupted operations. We reported computational results on the instances proposed by the ROADEF challenge 2009, an international competition carried out every two years. The goal of this challenge are to allow industrial companies witnessing recent developments in the field of OR and Decision Analysis; and to allow researchers showing their knowledge and demonstrating their know-how on practical problems. Our results were compared to the results given by the finalist teams of this competition. SAPI have been classified on the third position (over 11 qualified teams) of the competition, showing that the method is effective.

Another important result of this Thesis is the software called Railway Rescheduling Tool (RRT) that gives a framework for RRP algorithmic development. It has a database design for the data of this problem, an interface to call different solution methods, and offers visualization tools and reports to analyze different solutions (see Chapter 10 (appendix)).

Concerning other results of this Thesis, we would like to remark that the MAGES project was nominee finalist to PREDIT award in the category "*Technologies pour les transport de merchandises* (Technologies for cargo transportation)" in Paris, France (2008)[2]. Additionally, several parts of this Thesis were presented in OR conferences while some research papers were also developed and submitted for publication.

In conclusion, this Thesis has answered the objectives by proposing original models and solution methods for rescheduling problems in general and to the railway rescheduling problem in particular. Nevertheless, some aspects are still open for future research. The next subsection presents some of the perspectives of this work:

Perspectives

In spite of the last progress in OR methods and hardware, it seems that real-world optimization applications in the railway industry are not as extensive as in the airline industry. Our work is only one step in this direction and can be complemented in several directions.

It should be noted that the proposed approaches are based on the assumption that all incidents are known, *i.e.*, deterministic. It is true that the incidents that trigger the reactive procedure are known, but only the first ones. It is also evident that upcoming events, during the recovery horizon, can be also affected by independent future incidents. A way of dealing with this inconvenient and mitigating the effects of these possible unknown disruptions is developing a robust optimization procedure where some coefficients in the models are known within certain bounds. Definitely, the study of the robust railway rescheduling problem constitutes a line for future research.

[2]MAGES project was supervised by PREDIT

Numerical tests show also that MIP formulations have the disadvantage of requiring a large amount of memory because of the number of binary variables. Future directions of research should actually address improvements for larger instances using decomposition techniques.

The other direction of future research is to explicitly include other related resources for example the crew. Two different approaches are then possible: (a) full integrated or (b) separated formulation or decomposition. It is expected that a fully integrated formulation will be hard to solve because of the additional complexity added to the problem. On the other hand, an iterative procedure that connects the two separated models seems to be more adequate.

Chapter 10

Appendix: Software RRT (Railway Rescheduling Tool)

Railway Rescheduling Tool (RRT) is an application developed for testing the methods proposed in this document. The main objective of this tool is to give a framework to code algorithms for the railway rescheduling problem. The goal is to have a database design for the needed data of this problem, an interface to call different solution methods, and finally offers visualization tools and reports to analyze different solutions.

10.1 Principal Characteristics

- Train, network and incident's data are stored in a relational database. It uses (OLE-DB) that permits to use several database engines. The compatible systems are: Oracle, MS-SQL Server, My SQL, MS-Access, plain text files, and others. It is also possible to use it in a computer network, i.e., Internet.

- It was developed in .NET Framework using C Sharp, a modern, general-purpose, object-oriented programming language.

- This software has a graphical interface, completely designed for final users.

- Visualization tools: network viewer and traffic diagrams.

- Algorithms developed in independent classes (even they could be programmed in other languages).

- It uses the last optimization tools: IBM ILOG Cplex 11 and ILOG CP Optimizer 2.

- It has a graphical simulation tool for analyzing the original and the provisional schedule.

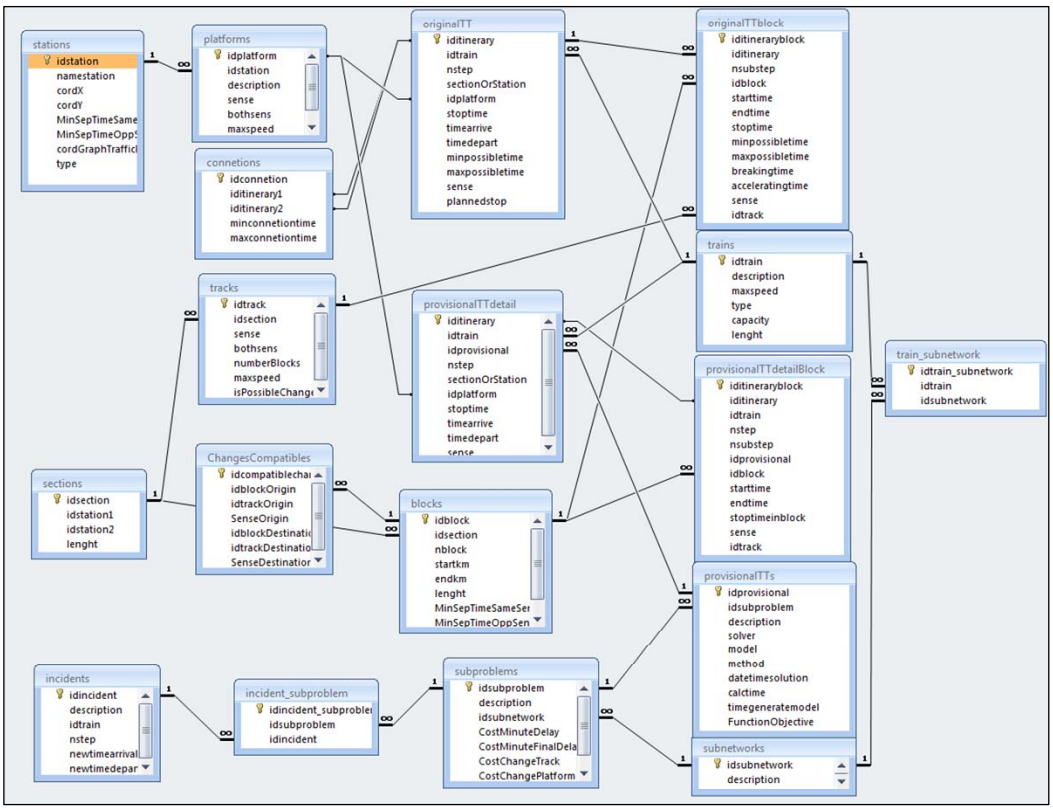


Figure 10.1: *Database design.*

10.2 Database Design

RRT uses a database design with several tables and relationship between them. This design permits to save separately data and problem instances. For example, it is possible to solve different kinds of incidents in the same network and also you can save different solutions to analyze and compare them before to take the last decision.

The database engine performs all the effort to give an answer for a given query.

10.3 Screenshots

Several screenshots of the RRT can be seen in Figures 10.2, 10.3, 10.4, 10.5, 10.6, 10.7, 10.8.

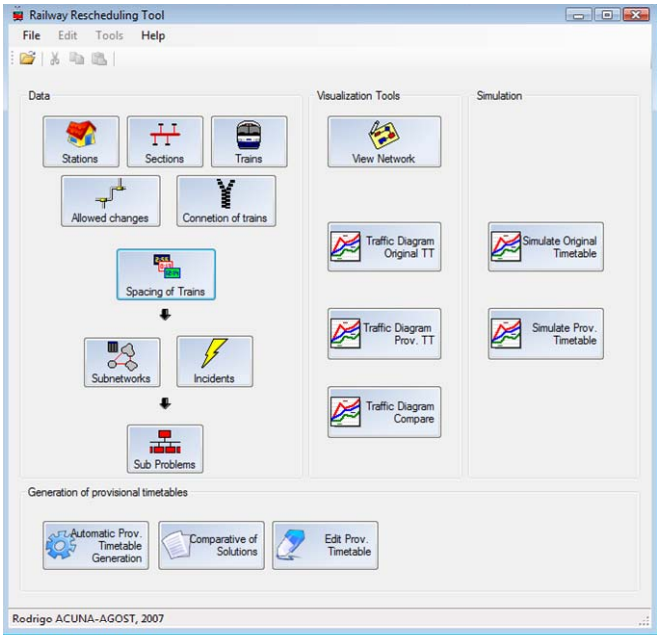


Figure 10.2: *Main Screen: The main screen permits to open a database file and to access directly to different functions of the software.*

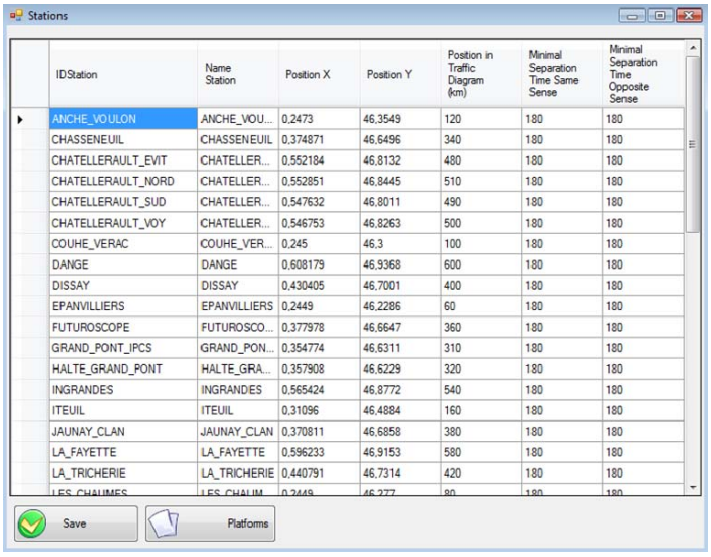


Figure 10.3: *Masters: This is one of the masters. Using the software, the user could change directly the database, without understand the complex relation between tables*

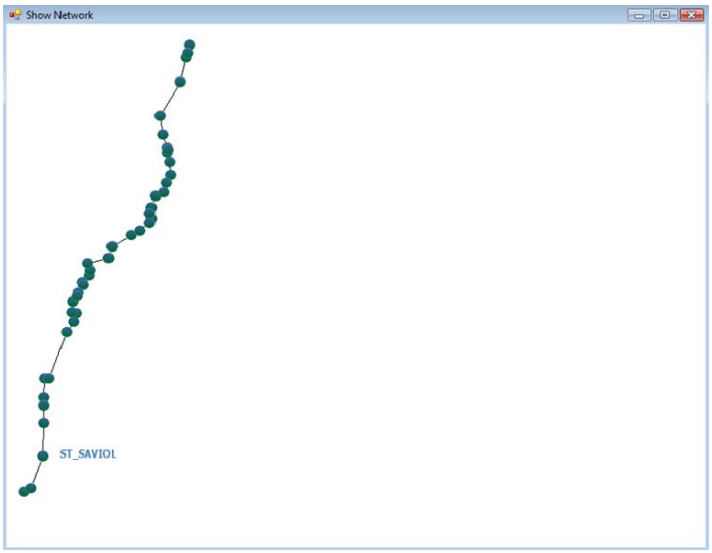


Figure 10.4: *Network Viewer: Using the coordinates of the stations, and the information about the sections, this software draws the network. The system is able to auto-scale the draw for using the whole screen and also, it uses the international coordinates system of latitude and longitude. For this example, it was used the exact coordinates of these stations given by Google Earth (GoogleInc, 2008)*

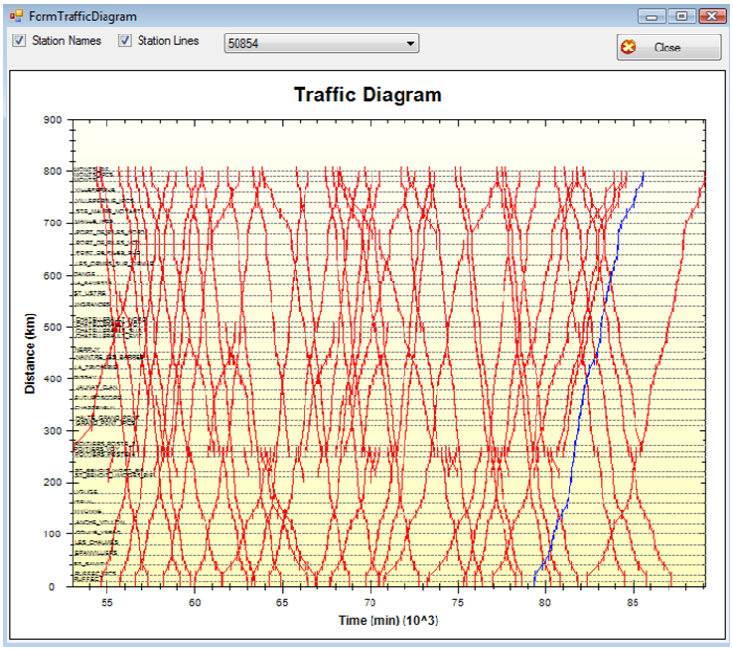


Figure 10.5: *Traffic Diagram: This is one of the most used graphics in train transportation. It is a two dimensional chart with the time in the x-axes and the distance in y-axes. The red lines represent trains, and also there are horizontal lines representing the stations*

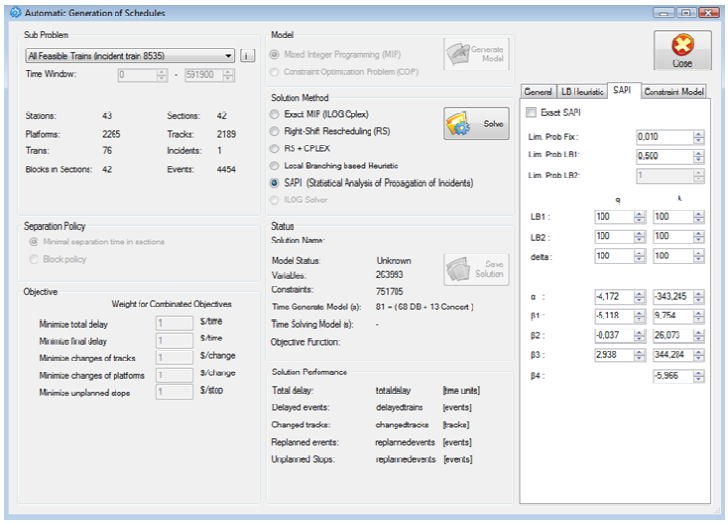


Figure 10.6: *Automatic generation of timetable: This window permits the user selects a sub-problem and to choose a solution method. The system reads the information from the database and generates an appropriate mathematical model depending on the solution method selected*

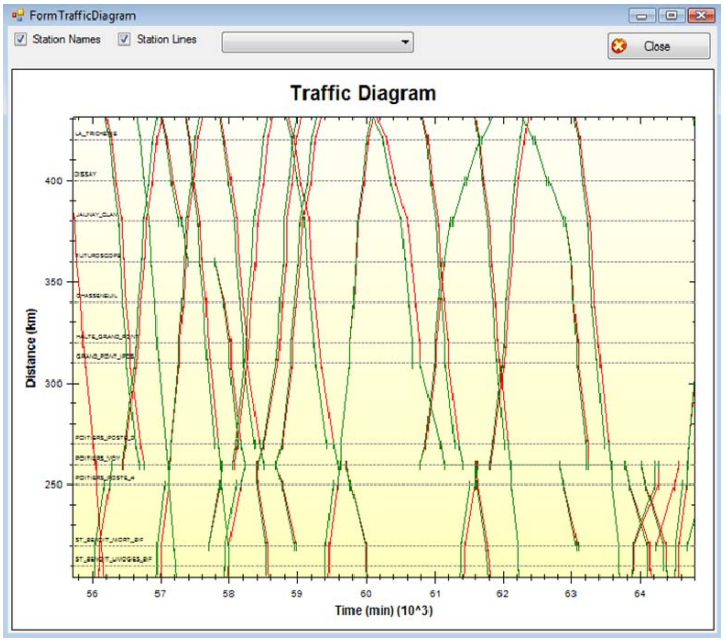


Figure 10.7: *Visualization Tool: This screen shows a typical solution of the problem. Green lines are the original schedule of trains and red lines are the provisional timetable. The horizontal distance between the red and green line for one train represents its delay.*

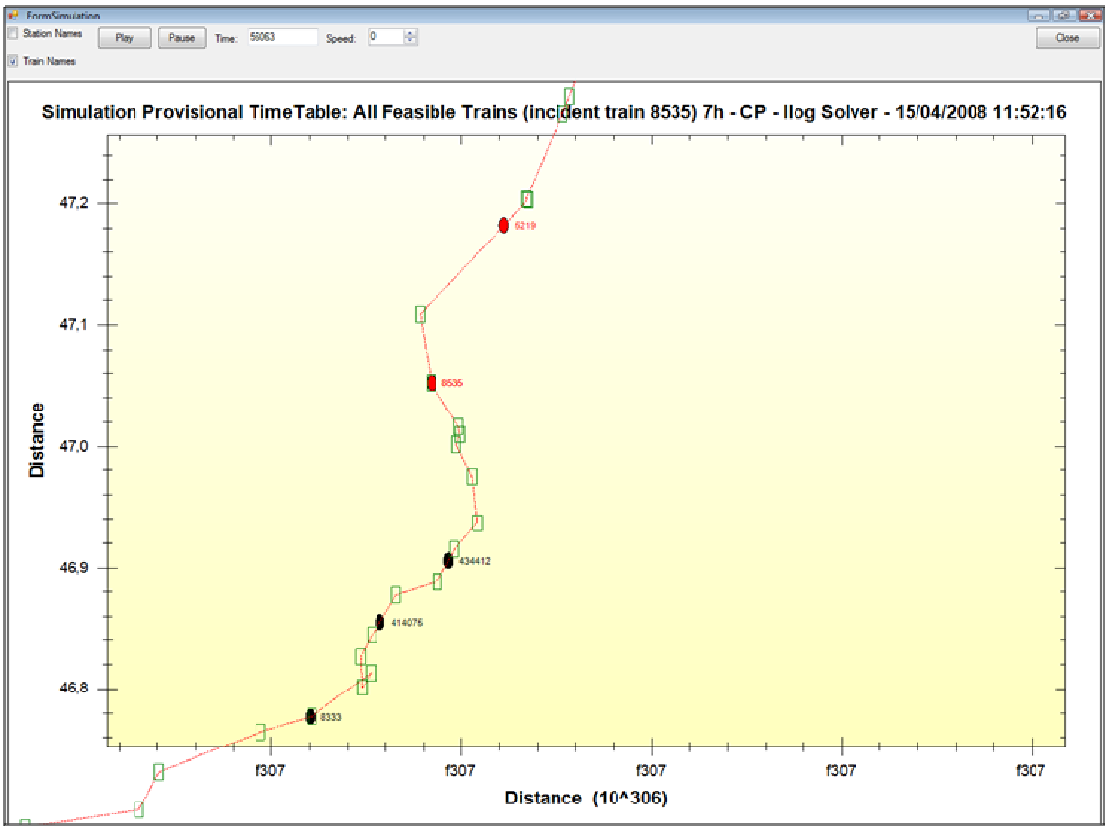


Figure 10.8: *Simulation: One of the visualization tools is Simulation. In this screen, users can simulate both: original and provisional timetables. The user is able to modify the speed, the scale, and other parameters of the simulation. One train can have three colors: black, red and green. The first one is used to represent a train in movement that is on time. Red trains are delayed trains moving. Finally, green trains correspond to trains stopped in a station.*

List of Figures

List of Tables

Bibliography

(Abrahamsson, 1998) T. Abrahamsson, 1998. Estimation of origin-destination matrices using traffic count - A literature survey. *Interim Report IR98 - 021, International Institute for Applied System Analysis, Laxenburg, Austria*.

(Abril et al., 2006) M. Abril, M. A. Salido, F. Barber, L. Ingolotti, A. Lova, and P. Tormos, 2006. Distributed models in railway industry. *Proc. 1st Workshop on Industrial Applications of Distributed Intelligent Systems (INADIS 2006). CD-ROM. ISBN 85-87837-11-7*.

(Acuna-Agost and Gueye, 2006) R. Acuna-Agost and S. Gueye, 2006. MAGES web page http://awal.univ-lehavre.fr/∼lmah/mages/. Internet.

(Adenso-Diaz et al., 1999) B. Adenso-Diaz, M. Oliva-Gonzalez, and P. Gonzalez-Torre, 1999. On-line timetable re-scheduling in regional train services. *Transportation Research Part B: Methodological 33*, 387–398.

(Aldrich, 1997) J. Aldrich, 1997. R. a. fisher and the making of maximum likelihood 1912 to 1922. *Statistical Science 12*(3), 162–176.

(Assad, 1980) A. Assad, 1980. Models for rail transportation. *Transportation Research Part A 14(A)*, 205–220.

(Bauer, 2008) A. Bauer, 2008. SNCF-RFF, le couple infernal du rail (in French). *Les Echos. 2008-11-12 (http://archives.lesechos.fr/archives/2008/LesEchos/20298-43-ECH.htm)*.

(Billionnet, 2003) A. Billionnet, 2003. Using integer programming to solve the train platforming problem. *Transportation Science* (37), 213–222.

(Brucker et al., 2002) P. Brucker, S. Heitmann, and S. Knust, 2002. Scheduling railway traffic at a construction site. *OR Spectrum 24*(1), 19–30.

(Bussieck et al., 1997) M. Bussieck, T. Winter, and U. T. Zimmermann, 1997. Discrete optimization in public rail transport. *Mathematical Programming 79*(1), 415–444.

(Cai and Wang, 1993) B.-G. Cai and J.-Z. Wang, 1993. A study of the expert system for train dispatching. *TENCON '93. Proceedings. Computer, Communication, Control and Power Engineering.1993 IEEE Region 10 Conference on*, 730–733.

(Cai and Goh, 1994) X. Cai and C. Goh, 1994. A fast heuristic for the train scheduling problem. *Computers & Operations Research 21*(5), 499–510.

(Caprara et al., 2001) A. Caprara, M. Fischetti, P. Guida, M. Monaci, G. Sacco, and P. Toth, 2001. Solution of real-world train timetabling problems. *Proceedings of the 34th Annual Hawaii International Conference System Sciences (HICSS-34) 3*, 3030.

(Caprara et al., 1999) A. Caprara, M. Fischetti, P. Guida, P. Toth, and D. Vigo, 1999. Solution of large-scale railway crew planning problems: the italian experience. *Computer-Aided Transit Scheduling. N.H.M. Wilson (Ed.). Springer. Lect. Notes Econ. Math. Syst. 471*, 1–18.

(Caprara et al., 2002) A. Caprara, M. Fischetti, and P. Toth, 2002. Modeling and solving the train timetabling problem. *Operations Research 50*, 851–861.

(Caprara et al., 1998) A. Caprara, M. Fischetti, P. Toth, and D. Vigo, 1998. Modeling and solving the crew rostering problem. *Operations Research 46*(6), 820–830.

(Caprara et al., 1997) A. Caprara, M. Fischetti, P. Toth, D. Vigo, and P. Guida, 1997. Algorithms for railway crew management. *Mathematical Programming* (79), 125–141.

(Caprara et al., 2007) A. Caprara, L. Kroon, M. Monaci, M. Peeters, and P. Toth, 2007. *Handbook in OR & MS, Vol. 14*, Chapter Passenger Railway Optimization (Chapter 3). Elsevier.

(Carey and Lockwood, 1995) M. Carey and D. Lockwood, 1995. A model, algorithms and strategy for train pathing. *Journal of the Operational Research Society* (46), 988–1005.

(Chang and Chung, 2005) S.-C. Chang and Y.-C. Chung, 2005. From timetabling to train regulation - a new train operation model. *Information and Software Technology 47*(9), 575–585.

(Cheng, 1998) Y. Cheng, 1998. Hybrid simulation for resolving resource conflicts in train traffic rescheduling. *Computers in Industry 35*(3), 233–246.

(Chiu et al., 2002) C. Chiu, C. Chou, J. Lee, H. Leung, and Y. Leung, 2002. A constraint-based interactive train rescheduling tool. *Constraints 7*, 167–198.

(Clausen et al., 2005) J. Clausen, A. Larsen, and J. Larsen, 2005. Disruption management in the airline industry - Review of models and methods. *Technical Report, Department of Informatics and Mathematical Modelling, Technical University of Denmark*.

(Clausen et al., 2001) J. Clausen, J. Larsen, A. Larsen, and J. Hansen, 2001. Disruption management - operations research between planning and execution. *unpublished: Technical report, IMM-2001-15 Informatics and Mathematical Modelling, Technical University of Denmark, Kgs. Lyngby, Denmark October 2001*.

(Cordeau et al., 2001) J.-F. Cordeau, F. Soumis, and J. Desrosiers, 2001. Simultaneous assignment of locomotives and cars to passenger trains. *Operations Research 49*(4), 531–548.

(Cordeau et al., 1998) J.-F. Cordeau, P. Toth, and D. Vigo, 1998. A survey of optimization models for train routing and scheduling. *Transportation Science 32*(4), 380–404.

(Cormen et al., 1990) T. Cormen, C. Leiserson, and R. Rivest, 1990. *Introduction to Algorithms*, Chapter 26.2 The Floyd Warshall Algorithm, 558–565. MIT Press and McGraw-Hill.

(Şahin, 1999) I. Şahin, 1999. Railway traffic control and train scheduling based on inter-train conflict management. *Transportation Research Part B: Methodological 33*(7), 511–534.

(D'Ariano et al., 2008) A. D'Ariano, D. Pacciarelli, and M. Pranzo, 2008. Assessment of flexible timetables in real-time traffic management of a railway bottleneck. *Transportation Research Part C: Emerging Technologies 16*(2), 232–245.

(D'Ariano et al., 2007) A. D'Ariano, M. Pranzo, and I. Hansen, 2007. Conflict resolution and train speed coordination for solving real-time timetable perturbations. *Intelligent Transportation Systems, IEEE Transactions on 8*(2), 208–222.

(DeLuca-Cardillo and Mione, 1998) D. DeLuca-Cardillo and N. Mione, 1998. k l-list τ colouring of graphs. *European Journal of Operational Research 106*(1), 160–164.

(Even et al., 1976) S. Even, A. Itai, and A. Shamir, 1976. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing 5*, 691–703.

(Filar et al., 2001) J. A. Filar, P. Manyem, and K. White, 2001. How airlines and airports recover from schedule perturbations: A survey. *Annals of Operations Research 108*(1-4), 315–333.

(Fioole et al., 2006) P.-J. Fioole, L. Kroon, G. Maróti, and A. Schrijver, 2006. A rolling stock circulation model for combining and splitting of passenger trains. *European Journal of Operational Research 174*(2), 1281–1297.

(Fischetti and Lodi, 2003) M. Fischetti and A. Lodi, 2003. Local branching. *Mathematical Programming 98*(1-3), 23–47.

(Flickr.com, 2006) Flickr.com, 2006. This image was originally posted to flickr by dustpuppy at www.flickr.com/photos/dustpuppy/78871005. It was reviewed on 21:50, 03 December 2006.

(Freling et al., 2002) R. Freling, R. M. Lentink, L. Kroon, and D. Huisman, 2002. Shunting of passenger train units in a railway station. *ERIM Report Series Reference No. ERS-2002-74-LIS*.

(Fukumori, 1980) K. Fukumori, 1980. Fundamental scheme for train scheduling : Application of rangeconstriction search. *Tech. Report A.I. Memo No. 596, Massachusetts Institute of Technology Artificial Intelligence Laboratory, USA*.

(Fukumori and Sano, 1987) K. Fukumori and H. Sano, 1987. Fundamental algorithm for train scheduling based on artificial intelligence. *Systems and Computers in Japan 18*(3), 52–64.

(GoogleInc, 2008) GoogleInc, 2008. Google earth version 4.3.7284.3916. *Web http://earth.google.com/*.

(Graham et al., 1979) R. Graham, E. Lawler, J. Lenstra, and A. Rinnooy Kan, 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics 5*, 287–326.

(Grandoni and Italiano, 2006) F. Grandoni and G. Italiano, 2006. Algorithms and constraint programming. *Lecture Notes in Computer Science, Book Principles and Practice of Constraint Programming - CP 2006 4204*, 2–14.

(Gueye, 2008) S. Gueye, 2008. Modules d'Aide à la GEstion des sillons (MAGES) - Rapport final d'activités. *Website MAGES - http ://awal.univ-lehavre.fr/∼lmah/mages*.

(Herrmann, 2006) T. M. Herrmann, 2006. *Stability of Timetables and Train Routings through Station Regions*. PhD Thesis, IFOR, Departement of Mathematics, Swiss Federal Institute of Technology Zurich.

(Higgings et al., 1997) A. Higgings, E. Kozan, and L. Ferreira, 1997. Heuristic techniques for single line train scheduling. *Journal of Heuristics* (3), 43–62.

(Hillier and Lieberman, 2001a) F. S. Hillier and G. J. Lieberman, 2001a. *Introduction to Operations Research*, Chapter 4. Solving linear programming problems: the Simplex Method, 109–190. McGraw-Hill.

(Hillier and Lieberman, 2001b) F. S. Hillier and G. J. Lieberman, 2001b. *Introduction to Operations Research*, Chapter 12. Integer Programming, 576–654. McGraw-Hill.

(Hirai et al., 2006) C. Hirai, N. Tomii, Y. Tashiro, S. Kondou, and A. Fujimori, 2006. An algorithm for train rescheduling using rescheduling pattern description language R. *Computers in Railways X: Computer System Design and Operation in the Railway and Other Transit Systems (COMPRAIL 2006) 88*.

(Ho and Yeung, 2001) T. Ho and T. Yeung, 2001. Railway junction traffic control by heuristic methods. *Proceedings IEEE - Electric Power Applications 148*(1), 77–84.

(Honore, 2008) R. Honore, 2008. SNCF : un rapport préconise le transfert de 14000 cheminots à une filiale indépendante (in French). *Les Echos. 2008-10-10*.

(Huisman et al., 2005) D. Huisman, L. Kroon, R. Lentink, and M. Vromans, 2005. Operations research in passenger railway transportation. *Erasmus Research Institute of Management (ERIM). Research Paper with number ERS-2005-023-LIS*. available at http://ideas.repec.org/p/dgr/eureri/30002129.html.

(Iyer and Ghosh, 1995) R. Iyer and S. Ghosh, 1995. Daryn, a distributed decision-making algorithm for railway networks: Modeling and simulation. *IEEE transactions on vehicular technology 1*, 269–274.

(Jarrah et al., 1993) A. Jarrah, G. Yu, N. Krishnamurthy, and A. Rakshit, 1993. A decision support framework for airline flight cancellations and delays. *Transportation Science 27*(3), 266–280.

(Jespersen-Groth et al., 2007) J. Jespersen-Groth, D. Potthoff, J. Clausen, D. Huisman, L. Kroon, G. Maróti, and M. Nyhave Nielsen, 2007. Disruption management in passenger railway transportation. *Working paper Erasmus Universiteit Rotterdam (http://hdl.handle.net/1765/8527)*.

(Jia and Zhang, 1994) L. Jia and X. Zhang, 1994. Distributed intelligent railway traffic control based on fuzzy decisionmaking. *Fuzzy Sets and Systems 62*(3), 255–265.

(Jovanovic and Harker, 1991) D. Jovanovic and P. Harker, 1991. Tactical scheduling of rail operations: The SCAN I system. *Transportation Science 25*(1), 46–64.

(Kitahara et al., 2000) F. Kitahara, K. Kera, and K. Bekki, 2000. Autonomous decentralized traffic management system. *Proceedings International Workshop on Autonomous Decentralized Systems*, 87–91.

(Kroon et al., 2007) L. Kroon, R. M. Lentink, , and L. Schrijver, 2007. Shunting of passenger train units: an integrated approach. *ERIM Report Series Reference No. ERS-2006-068-LIS*.

(Lamma et al., 1997) E. Lamma, P. Mello, and M. Milano, 1997. A distributed constraint-based scheduler. *Artificial Intelligence in Engineering 11*(2), 91–105.

(Larroche et al., 1996) Y. Larroche, R. Moulin, and D. Gauyacq, 1996. Sepia: a real-time expert system that automates train route management. *Control Engineering Practice 4*(1), 27–34.

(Lettovský, 1997) L. Lettovský, 1997. *Airline operations recovery: An optimization approach*. PhD Thesis, Georgia Institute of Technology.

(Lomazzi, 2008) M. Lomazzi, 2008. Air france lance ses TGV (in French). *Le Parisien. 2008-09-08*.

(Lustig and Puget, 2001) I. J. Lustig and J.-F. Puget, 2001. Program does not equal program: Constraint programming and its relationship to mathematical programming. *Interfaces 31*(6), 29–53.

(M. Peeters, 2008) L. K. M. Peeters, 2008. Circulation of railway rolling stock: A branch-and-price approach. *Computers & Operations Research 35*(2), 538Ů556.

(Maróti, 2006) G. Maróti, 2006. *Operations research models for railway rolling stock planning*. PhD Thesis, Eindhoven, Technische Universiteit Eindhoven.

(Maróti and Kroon, 2007) G. Maróti and L. Kroon, 2007. Maintenance routing for train units: The scenario model. *Computers & Operations Research* (34), 1121–1140.

(Mautor and Michelon, 1997) T. Mautor and P. Michelon, 1997. MIMAUSA: A New Local Search Method. *Tenth Meeting of the European Chapter on Combinatorial Optimization (ECCO X), Spain*.

(Mautor and Michelon, 2001) T. Mautor and P. Michelon, 2001. MIMAUSA: an application of referent domain optimization. *Laboratoire Informatique (LIA) - Universite d'Avignon et des Pays de Vaucluse, submitted to Annals of Operations Research*.

(Mesa and Boffey, 1996) J. Mesa and T. Boffey, 1996. A review of extensive facility location in networks. *European Journal of Operational Research 95*(3), 592–603.

(Missikoff, 1997) M. Missikoff, 1997. An object-oriented approach to an information and decision support system for railway traffic control. *First International Conference on Knowledge-Based Intelligent Electronic Systems KES'97*.

(Mladenović and Čangalović, 2007) S. Mladenović and M. Čangalović, 2007. Heuristic approach to train rescheduling. *Yugoslav Journal of Operations Research 17*(1), 9–29.

(Nobuyuki et al., 1996) O. Nobuyuki, M. Akatsu, and T. Murata, 1996. Rescheduling method based on distributed cooperative problem solving model. *Proceedings Emerging Technologies and Factory Automation. EFTA '96 1*, 35–41.

(Norio et al., 2005) T. Norio, T. Yoshiaki, T. Noriyuki, H. Chikara, and M. Kunimitsu, 2005. Train rescheduling algorithm which minimizes passengers dissatisfaction. *Lecture Notes in Computer Science 3533/2005*, 829–838.

(Oliveira and Smith, 2001) E. Oliveira and B. Smith, 2001. A hybrid constraint-based method for single-track railway scheduling problem. *Report: Leeds Univ. (United Kingdom). School of Computer Studies*.

(Ovacik and Uzsoy, 1992) I. Ovacik and R. Uzsoy, 1992. Analysis of periodic and event-driven rescheduling policies in dynamic shops. *International Journal of Computer Integrated Manufacturing 5*(3), 153–163.

(Pachl, 2004) J. Pachl, 2004. *Railway Operation and Control*. VTD Rail Publishing.

(Ping et al., 2001) L. Ping, N. Axin, J. Limin, and W. Fuzhang, 2001. Study on intelligent train dispatching. *Proceedings IEEE Intelligent Transportation Systems Conference*, 949–953.

(Rakshit et al., 1996) A. Rakshit, N. Krishnamurthy, and G. Yu, 1996. Systems operations advisor: A real-time decision support system for managing airline operations at united airlines. *Interfaces 26*(2), 50–58.

(Rebreyend, 2005) P. Rebreyend, 2005. DisTrain: A simulation tool for train dispatching. *8th Internation conference on Intelligent Transportation Systems*, 528–533.

(Rebreyend, 2006) P. Rebreyend, 2006. A dispatching tool for railway transportation. *The 6th International Conference on the Practice and Theory of Automated Timetabling. PATAT 2006*, 478–480.

(REDF, 2001) REDF, 2001. The roberts enterprise development fund. SRIO methodology, social return on investment. http://www.redf.org.

(RFF, 2009) RFF, 2009. 2007 financial report. *RFF website at http://www.rff.fr.*.

(ROADEF, 2008) ROADEF, 2008. Societe francaise de recherche operationnelle et aide a la decision, roadef 2009 challenge: Disruption management for commercial aviation. *Internet, http://challenge.roadef.org/2009*.

(Rodriguez, 2007) J. Rodriguez, 2007. A constraint programming model for real-time train scheduling at junctions. *Transportation Research Part B: Methodological 41*(2), 231–245.

(Sahin et al., 2005) G. Sahin, C. Cunha, and R. Ahuja, 2005. New approaches for the train dispatching problem. *Submitted to Transportation Research B*.

(Schaefer and Pferdmenges, 1994) H. Schaefer and S. Pferdmenges, 1994. An expert system for real-time train dispatching. *Computers in Railways IV, Vol 2 Railway Operations*.

(Schoenauer and Semet, 2005) M. Schoenauer and Y. Semet, 2005. An efficient memetic, permutation-based evolutionary algorithm for real-world train timetabling. *http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0510091*.

(Schrijver, 1993) A. Schrijver, 1993. Minimum circulation of railway stock. *CWI Quarterly 6*, 205–217.

(Semet and Schoenauer, 2006) Y. Semet and M. Schoenauer, 2006. On the benefits of inoculation, an example in train scheduling. *Proceedings of the 8th annual conference on Genetic and evolutionary computation GECCO '06*, 1761–1768.

(Serafini and Ukovich, 1989) P. Serafini and W. Ukovich, 1989. A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics 2*(4), 550–581.

(Silva de Oliveira, 2001) E. Silva de Oliveira, 2001. *Solving Single-Track Railway Scheduling Problem Using Constraint Programming*. PhD Thesis, Univ. of Leeds, School of Computing.

(SNCF-Group, 2005) SNCF-Group, 2005. Annual report and sustainable development report 2005 - Perfomance Indicators.

(Szpigel, 1973) B. Szpigel, 1973. Optimal train scheduling on a single track railway. *Proceedings of IFORS Conference on Operational Research 1972*, 343–351.

(Tazoniero et al., 2005) A. Tazoniero, R. Goncalves, and F. Gomide, 2005. Fuzzy algorithm for real-time train dispatch and control. *Fuzzy Information Processing Society, 2005. NAFIPS 2005. Annual Meeting of the North American*, 332–336.

(Teodorovic and Guberinic, 1984) D. Teodorovic and S. Guberinic, 1984. Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operational Research 15*(2), 178–182.

(Teodorovic and Stojkovic, 1990) D. Teodorovic and G. Stojkovic, 1990. Model for operational daily airline scheduling. *Transportation Planning and Technology 14*(4), 273–285.

(Teodorovic and Stojkovic, 1995) D. Teodorovic and G. Stojkovic, 1995. Model to reduce airline schedule disturbances. *Journal of Transportation Engineering 121*(4), 324–331.

(Tormos et al., 2006) P. Tormos, M. Abril, M. A. Salido, F. Barber, L. Ingolotti, and A. Lova, 2006. Distributed constraint satisfaction problems to model railway scheduling problems. *Computers in Railways X: Computer System Design and Operation in the Railway and Other Transit Systems*.

(Tornquist, 2006a) J. Tornquist, 2006a. Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms. *In: L. G. Kroon and R. H. Mohring (Eds.) 5th Workshop on Algorithmic Methods and Models for Optimization of Railways.*.

(Tornquist, 2006b) J. Tornquist, 2006b. *Railway traffic disturbance management*. PhD Thesis, Blekinge Institute of Technology, Sweden, Department of Systems and Software Engineering, School of Engineering, Blekinge Institute of Technology, Box 520, SE-372 25 Ronneby, Sweden.

(Törnquist, 2007) J. Törnquist, 2007. Railway traffic disturbance management - an experimental analysis of disturbance complexity, management objectives and limitations in planning horizon. *Transportation Research Part A 41*(3), 249–266.

(Törnquist and Davidsson, 2002) J. Törnquist and P. Davidsson, 2002. A multi-agent system approach to train delay handling. *Proceedings of the ECAI-02 Workshop on Agent Technologies in Logistics 2002*, 50–53.

(Tornquist and Persson, 2005) J. Tornquist and J. A. Persson, 2005. Train traffic deviation handling using tabu search and simulated annealing. *Proceedings of the 38th Hawaii International Conference on System Sciences 3*, 73a.

(Törnquist and Persson, 2007) J. Törnquist and J. A. Persson, 2007. N-tracked railway traffic re-scheduling during disturbances. *Transportation Research Part B 41*(3), 342–362.

(Vernazza and Zunino, 1990) G. Vernazza and R. Zunino, 1990. A distributed intelligence methodology for railway traffic control. *IEEE Transactions on Vehicular Technology 39*(3), 263–270.

(Vieira et al., 2003) G. E. Vieira, J. W. Herrmann, and E. Lin, 2003. Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *Journal of Scheduling 6*(1), 39–62.

(Vieira et al., 1999) P. Vieira, L. Neto, E. Bessa, and F. Gomide, 1999. Railway dispatch planning and control. *18th International Conference of the North American Fuzzy Information Processing Society*, 134–138.

(Walker et al., 2005) C. Walker, J. Snowdon, and D. Ryan, 2005. Simultaneous disruption recovery of a train timetable and crew roster in real time. *Computers & Operations Research* (32), 2077–2094.

(Wegele and Schnieder, 2004a) S. Wegele and E. Schnieder, 2004a. Automated dispatching of train operations using genetic algorithms. *Computers in Railways IX. WIT Press, Southampton, 2004*, 775–784.

(Wegele and Schnieder, 2004b) S. Wegele and E. Schnieder, 2004b. Dispatching of train operations using genetic algorithms. *Proceedings of the 9th International Conference on Computer-Aided Scheduling of Public Transport - CD-ROM, San Diego 2004*.

(Yu et al., 2003) G. Yu, M. Argüello, S. Gao, S. M. McCowan, and A. White, 2003. A new era for crew recovery at continental airlines. *Interfaces 33*(1), 5–22.

(Yu and Qi, 2004) G. Yu and X. Qi, 2004. *Disruption Management. Framework, Models and Applications*. World Scientific Publishing Co Pte Ltd.

(Zwaneveld et al., 1996) P. Zwaneveld, L. Kroon, H. Romeijn, M. Salomon, S. Dauzere-Peres, C. van Hoesel, and H. Ambergen, 1996. Routing trains through railway stations: Model formulation and algorithms. *Transportation Science* (30), 181–194.