INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Railway Traffic Management

Pedro Alexandre de Carvalho Afonso

Dissertation submitted for obtaining the degree of

Master in **Electrical and Computer Engineering**

Jury

Advisor: Prof. Carlos Filipe Gomes Bispo

Thesis Comitee

President: Prof. Carlos Jorge Ferreira Silvestre

Members: Prof. Paulo Fonseca Teixeira

Dr. Matthijs Theodor Jan Spaan

Prof. Carlos Filipe Gomes Bispo

September 2008

Acknowledgements

My first words go to my family, to whom I deeply thank, for the infinite patience and infinite support.

I would also like to thank Prof. Carlos Bispo, for his guidance throughout this year and availability, especially in the final stage.

I want to dedicate a very special word to my group of friends namely, Diogo Couto, Ricardo Mah, Bruno Baleizão, Duarte Estrada, Rui Tavares, André Violante, Miguel Pacheco, Armando Marques and Paulo Andrade, who give me the honour to be my friends.

A deep thank you, to all of the above mentioned as well as to everyone close to me, which, in their own way, contributed for the development of this Thesis.

Abstract

With the growing interest in the railway sector, mainly because of energetic reasons, there is also a need to increase the efficiency of the railway lines. One way to optimize this sector is to improve quality in the train control process itself. Nowadays, train dispatching is still mostly done by human operators that use elementary tools and thereby solving conflicts sub-optimally.

Motivated by these factors, this report presents a model capable of detecting and solving conflicts, for single track railways. More specifically, this model proposes two resolution methods: a heuristic resolution and a search for the optimal solution.

To evaluate the quality of the developed model, several tests were made, obtaining encouraging results. These results showed that it is possible to solve conflicts optimally or near optimally, in a feasible amount of time.

This program comes with a graphic interface so that the interaction with the dispatcher can be more user friendly.

The major novelties of this work regard the improvement on the conflict detection process, the introduction of capacity conflicts, and the creation of several parameters to adjust the search for the optimal solution.

Keywords

Re-scheduling, Single Track, Railway, Meet and Pass, Train, Conflict, Decision Support System

Resumo

Com o interesse crescente no sector ferroviário, devido à crise energética, cresce também a necessidade de rentabilizar ao máximo a utilização das suas linhas. Uma forma de o fazer, passa pela optimização do controlo ferroviário. Este controlo, hoje em dia, continua a ser feito por controladores humanos que, com algumas ferramentas elementares, tomam decisões de forma não óptima.

Motivado por estes factores, este relatório apresenta um modelo capaz de detectar e resolver conflitos, para linhas de baixo tráfego. Mais precisamente, este modelo propõe dois processos diferentes de resolução, sendo um deles uma resolução heurística e o outro uma procura pela solução óptima.

Para avaliar a qualidade do modelo desenvolvido, foram feitos vários testes de simulação, obtendo-se resultados encorajadores, que demonstram ser possível resolver conflitos de forma óptima ou quase óptima, em tempo aceitável.

Este programa vem acompanhado de uma interface gráfica, de modo a facilitar a interacção entre o programa e o controlador.

As principais inovações deste trabalho estão relacionadas com a melhoria no processo de detecção de conflitos, a introdução de capacidades nas estações, e na criação de vários parâmetros para ajustar a procura pela solução óptima.

Palavras-chave

Comboio, Linha de baixo tráfego, Sistema de apoio à decisão, Conflito, Horário.

Table of Contents

List of Figures

List of Tables

List of Acronyms

DSS	Decision Support System
FIFO	First In First Out
FOFI	First Out First In
FCFS	First Come First Served
FLFS	First Leave First Served
GUI	Graphical User Interface

Chapter 1

Introduction

This chapter gives a brief overview of the work. Before establishing work targets and original contributions, the scope and motivations are brought up. At the end of the chapter, the work structure is provided.

1.1 Context and Motivation

The railway industry plays a vital role in many countries. All railway companies try to achieve more regular and reliable train services, in order to satisfy their customers. One way to optimize these services is to improve quality in the train control process itself. Therefore, railway operators plan train services in detail (i.e. timetables), defining the train order and timing, at junctions and platforms. A robust timetable should be able to deal with minor delays occurring in real-time. However, unexpected events such as technical failures, track incidents, etc, may cause primary delays, which affect the running times, dwelling and departing events. Due to the interaction between trains, these delays may be propagated as secondary delays to other trains, and so, disturbing the entire network. This is why train dispatching is very important. Not only do dispatching orders keep the railway safe from collisions, but also have the objective of minimizing secondary delays throughout the network. Now, as the first objective may be simpler to ensure, the second one is a lot more complicated and challenging. Even so, train dispatching is still mostly done by human operators that use elementary Decision Support Systems (DSS). These systems help them to quickly and effectively re-schedule train movements, according to simple dispatching rules. Such rules solve conflicts by means of a simple and local decision criterion, not taking a global look at the system and hence, not searching for an optimal solution. In the last decade however, with the strong competition facing rail carriers, the privatization of many national railroads, and the enormous advances in technology, like computers and telecommunications, many researchers developed new optimization models. Despite the recent advances, providing satisfactory solutions for such a complex problem is still drawing the attention of researchers.

1.2 Objectives

The main goal of this work is to develop, implement and evaluate a model, capable of solving the meet and pass problem and then provide feasible solutions to the human controller. The objectives of this Thesis are:

- Detect conflicts between trains in a single railway line
- Create a first solution for the conflicts, in a short period of time
- Search for the optimal solution
- Propose several near optimal solutions to the dispatcher

1.3 Summary of original contributions

This Thesis includes several contributions to the field of train re-scheduling, with the particular emphasis on the conflict detection and in the resolution process.

The first contribution of this Thesis is the optimization of the conflict detection process. The algorithm detects conflicts without having to check all the train pairs in the schedule. This improvement results in a conflict detection algorithm with a linear complexity, and consequently, saving time in the overall process of searching for the optimal solution.

The introduction of capacity conflicts in the problem is also an original contribution. Taking capacity into consideration makes the problem more complex and harder to solve, but on the other hand, it adds more realism to the model.

Most models in the literature focus in achieving either a fast heuristic solution or an optimal solution after a long exhaustive search. As for this model, it allows for the dispatcher to have a mixture of both solutions. This is possible because of the search parameters introduced, like the time horizon and the maximum search time, which make the optimal search more versatile.

1.4 Structure of this Thesis

This Thesis is organized in six chapters, including this first one introducing the Thesis, and the sixth one referring to the conclusions and future work.

Chapter 2 presents a detailed review on the train re-scheduling problem as well as the main technologies and systems already developed to address the problem. First, some basic notions about this problem are provided.

Chapter 3 introduces the model developed in this Thesis, by presenting its architecture and a functional description of each of its modules. Then, the mathematical formulation of the model is provided.

Chapter 4 presents an in-depth description of all architecture's modules, particularly the algorithms implemented, in order to allow the reader to get a complete understanding of the entire process. This chapter also intends to provide sufficient information for a proper and easy usage of the application by any user.

Chapter 5 presents the results of the tests carried out to evaluate the program performance.

Finally, Chapter 6 is dedicated to the conclusions and eventual future work.

Chapter 2

Models for Train Dispatching: Literature Review

This chapter has the objective to provide an overview on train re-scheduling models. With that purpose in mind, first some basic notions about train dispatching are explained. Then in, Section 2.2, a summarization of the problem is made followed by a brief review of the most relevant models.

2.1 Basic Concepts

Before describing the most relevant dispatching models, it is important to explain some aspects of train dispatching.

2.1.1 Definition of terms used

Along this Thesis, there will be used some technical terms which will now be formally defined:

Single railway track - One where traffic in both directions shares the same track

Track Segment - A track segment is a part of a railway line that is bounded by two distinct end points, in this case the meetpoints

Block – Section of the railway line delimited by signals

Siding - A section of track which can be used for the crossing or passing of trains under single track operations. The terms ‘crossing loop’ or ‘passing loop’ are also used in some countries to describe such track sections. A train station on a single line track will usually contain a siding.

Meetpoint - Location where two trains may cross simultaneously. In this context, meetpoints include not just stations, but also sidings.

Train Conflict - There are two situations, namely: when two trains approach each other on a single line track travelling in opposite directions (Meet); and when a faster train catches a slower train travelling in the same direction (Pass).

Minimum Headway - The minimum time length separating two trains on a single line track. This is usually determined by signals in the case when the trains are travelling in the same direction. When travelling in opposite directions, the minimum headway is determined by the time required for one train to clear the track segment sufficiently before the opposing train can enter it.

2.1.2 Timetables

Train dispatchers have two main tools to supervise the network which are the timetable and the train diagram. The timetable is a schedule of trains on a given railway infrastructure. It contains the arrival and departure times of the trains, not only from stations, but also from intermediate stations and sidings. The train diagram, or graphical timetable, is a representation of the timetable in a more perceptive way. It is basically a time distance graph where all the trains’ routes are represented. The advantage of this diagram is that it makes it much more simple and intuitive to read the timetable and to detect conflicts. In Figure 2.1 is an example of a train diagram with 26 passenger trains and 17 stations.

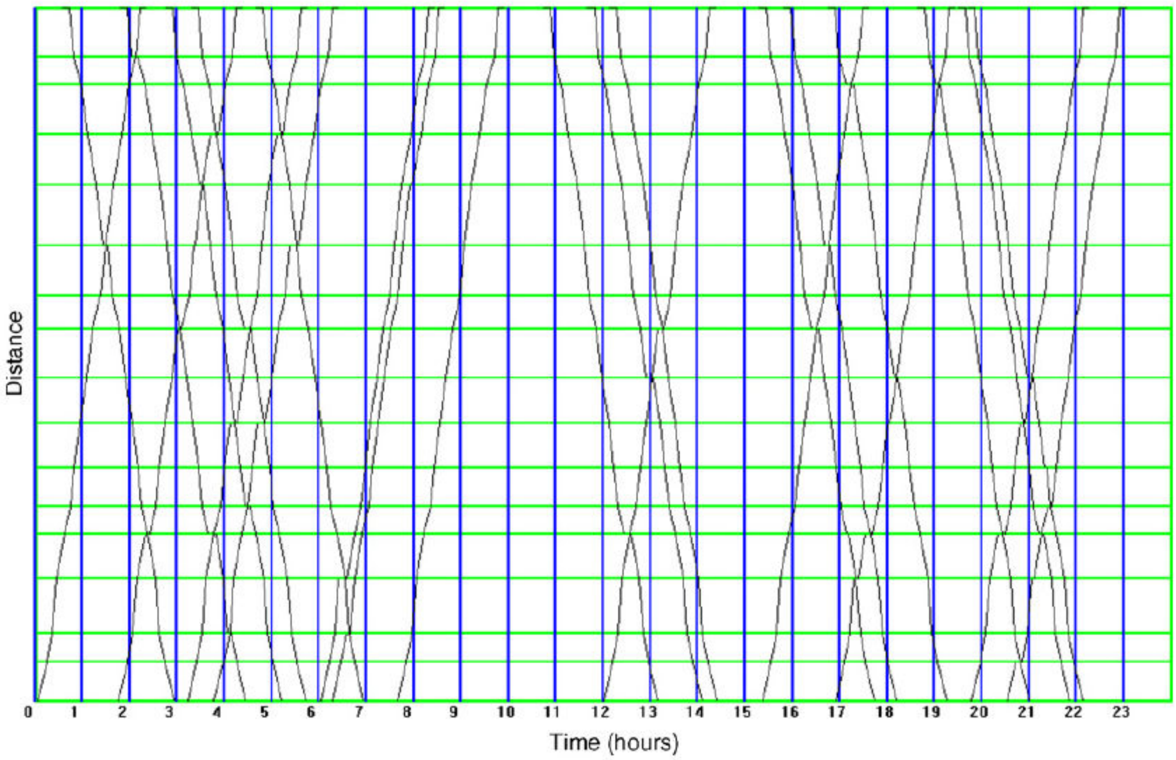


Figure 2.1 Train diagram example (extracted from [Zhou2006]).

The X axis represents time of day, the Y axis the sequence of stations in distance scale. Lines indicate the movement of trains, with the slope indicating direction and speed, horizontal meaning stand-still. Usually, the outbound direction is defined upwards. In this example, the green horizontal lines represent the meetpoints along the track. This means that all the intersections between black lines (trains) must also intersect with one green line (meetpoint) to be free of conflicts.

2.1.3 Safety technologies

There are two different systems to ensure the safety of the railway networks: the fixed block technology and the moving block technology. A block section is a track segment between two signals. The signals control the train traffic and impose safe distance headways. There are signals along the lines and also before every station, passing loops, junctions, etc. The most common type of signalling is the three-aspect signalling. A signal aspect may be red, yellow or green. If the subsequent block is occupied by another train, the signal is red. A yellow signal aspect means that the subsequent block section is empty, but the following block is occupied by another train. Finally, a green signal aspect indicates that the next two blocks are empty. Trains have to stop when facing a red signal and wait for that signal to change to green or yellow. When facing a yellow sign the train may proceed its course but has to decelerate so that it can stop in case the next signal is red. Each block may host at most one train at a time.

As for the moving block technology, it does not need signals since the exact position and speed of each train is known. Safety is ensured by the regulation of their respective speeds. The safety standards define a maximum speed for each train, depending on the distance from the preceding train, necessary to grant space to stop completely in case of emergency. With this technology, a track segment can host more than one train at a time.

2.1.4 Dispatching rules

Dispatching rules solve conflicts by means of a local decision criterion. Two of the most common rules are the first-in-first-out (FIFO) rule and the first-out-first-in (FOFI) rule. The FIFO rule solves conflict situations by assigning the block section in discussion to the first train that required it. This means that this rule actually does not require any special dispatching order and lets the traffic proceed with its actual order. This rule is also known as first-come-first-served (FCFS).

The FOFI rule assigns the block to the first train that is able to leave it. For this evaluation, the time that each train will take to enter the block and leave it available again has got to be calculated first. The precedence is then given to the train that is able to leave the block first. The FOFI is also referred to as first-leave-first-served (FLFS).

2.1.5 Objective functions

For evaluating the quality of the obtained solutions, dispatching models need to have an objective function. There are several objective functions each one with its qualities. The choice of what function to use depends on the dispatcher criteria. The following four optimization criteria, i.e., objective functions have been selected as the most common:

- Minimization of the total tardiness $D = \sum_{i=1}^{n} T_i$;
- Minimization of the total weighted tardiness $WD = \sum_{i=1}^{n} w_i T_i$;
- Minimization of the maximum tardiness $T_{max} = max\{T_1, T_2, \dots, T_n\}$. This criterion minimizes the maximum delay but many trains may suffer small delays. This is a criterion that is more suited when passenger trains prevail for scheduling;
- Minimization of the maximum weighted tardiness $WT_{max} = max\{w_1 T_1, w_1 T_2, \dots, w_n T_n\}$. This function is more suited for mixed traffic situations where the weights w_i are usually the same for trains with the same priority.

2.2 The Most Relevant Re-scheduling Models

2.2.1 Introduction

All of the models found in the literature present a common structure which is basically the structure of a typical decision support system. A Decision Support System (DSS) can be defined as a computer-based, interactive system that aids the process of decision making. For this particular case, it can also be called as an automated dispatching support system, which is a real-time traffic management system with a short time horizon that aids dispatchers to control the train traffic. These systems have three main functions:

- Predict train movements
- Detect expected conflicts
- Propose a solution for the conflicts

However, these dispatching systems are not designed to replace dispatchers but to support them in taking the best possible decisions.

Considering the main features of the typical models, they can be also described as follows:

- It serves a line between two major stations or terminals. There are normally several meetpoints on that line.
- Every train has a schedule for that line and that schedule represents the initial optimum.
- It is continuously updating train positions received from the signal systems.

Several approaches for re-scheduling railway traffic have been suggested since the early seventies. Still, it was only in the last decade, with the advances in technology, that the most relevant research was conducted. An extensive survey of approaches for railway traffic scheduling and re-scheduling can be found in [Cordeau98]. In this chapter, only some of the most relevant and recent models are described. Because of the large diversity that characterises most models, a textual description was adopted. Train dispatching systems can be broadly divided into two categories: fixed-speed, and variable-speed models. Fixed-speed models assume that trains travel at their maximum speed whenever possible, and later an acceptable speed profile is determined. Variable-speed models consider velocity as a variable giving more realism to the model.

2.2.2 Inter-train conflict management

This first model described here was developed by Ismail Sahin [Sahin99], and can be classified as a fixed-speed model. In this model a heuristic algorithm for rescheduling trains in a single-track railway was developed. The objective was to obtain better conflict solutions than train dispatchers, and optimal or near optimal solutions in a reasonable amount of time. The time horizon considered is a single day. Assuming initially a conflict-free schedule, the algorithm compares the scheduled arrival times with the actual arrival times in order to detect deviations from the initial schedule. Then, for this set of disturbed trains, a discrete-event simulator verifies if there are any conflicts and which one will be the first to occur. This will be the only conflict that the program will solve, not taking into account the other following potential conflicts. Since there can be only two resolutions for the conflict, which is stopping one train or the other, the program will simulate both alternatives. For each case, the algorithm will calculate the expected time of arrival at scheduled points of every train in the problem, taking into consideration their future conflict delays. Finally the decision is taken choosing the alternative that causes the minimum total delay of the system. The algorithm proceeds by detecting and resolving the immediate conflicts, consecutively, one at a time. In Figure 1 is a flowchart that summarizes this algorithm.

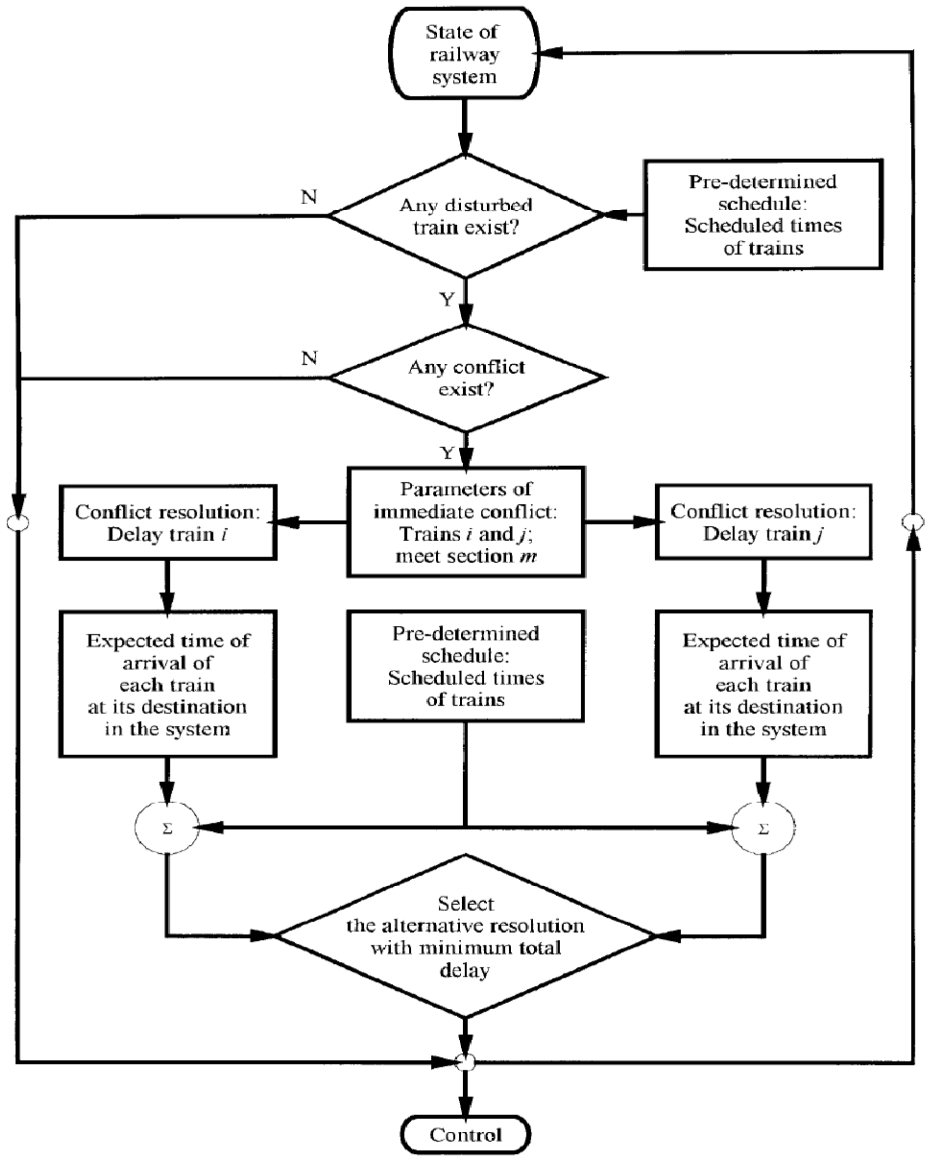


Figure 2.2 Process of railway traffic control based on inter-train conflict management (extracted from [Sahin99]).

Good results are reported for computational experiments on small instances with 20 trains and 19 meetpoints. The optimal solution was found for 13 times over 26 problem instances. Comparing the heuristic algorithm with the dispatchers' solution, the savings in the total waiting times is 2.5 min per train.

2.2.3 On-line timetable re-scheduling

This second fixed-speed model was proposed in [Adenso-Díaz99] for solving real-time timetable perturbations on a regional network. They did an interesting research on historical data about the possible incidents and their average duration. This is very important so as to be able to foresee the influence that an incident will have on timetables. The model used is a MIP (mixed integer program) where the objective function, to be maximized, is the number of passengers transported. As expected, this problem is very complex, with several constraints, hence a backtracking method is used to explore the solution space. The quality of each solution is calculated considering the number of passengers transported, the delays and the priority of each service. In Figure 2.3. is an example of the search tree used for exploring the solution space.

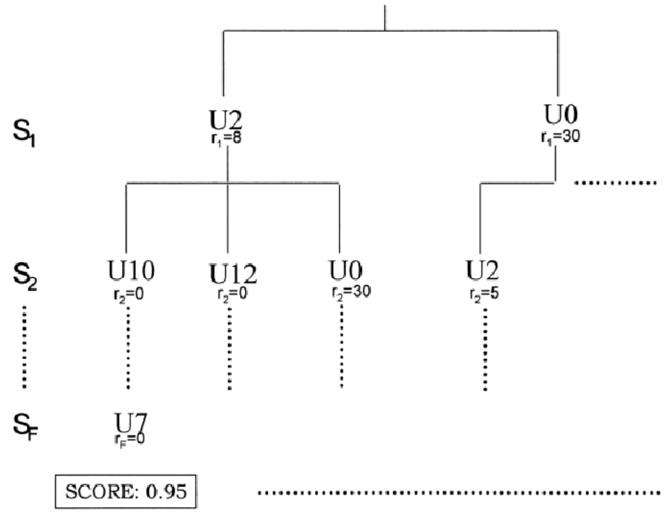


Figure 2.3 An example of the backtracking tree in the process of exploring the solution space (extracted from [Adenzo-Díaz99]).

A horizon of the next F services is considered in order to reduce the search tree. Also, a depth-first search with a branch-and-bound procedure is chosen. Note that all possible solutions are at the level F of the tree. Finally a specified number of the best possible solutions are presented to the train dispatcher. The system was implemented in 1998 in Asturias, Spain, and was able to offer a set of useful solutions in less than 5 min.

2.2.4 Greedy Travel-Advance Strategy

In [Medanic02] a discrete-event model was used to obtain time-efficient and energy-efficient suboptimal schedules. This discrete-event model approach takes much less effort to develop than formulating an integer programming problem as has been done in the past.

The discrete events considered are the times when a train reaches a meetpoint. Assuming that the velocities of the trains for each section of the route are fixed, and that the times of origin of the trains are given, the time for the next event can be easily calculated. This algorithm is called greedy because it is locally optimal and depends on local information. This means, when a train reaches a meetpoint the algorithm will only consider the trains in its vicinity. If this train can reach the next meetpoint safely, then the model proceeds to the next event, if not, the algorithm decides which nearby train has to be stopped at a meetpoint.

Since this model assumes that trains travel at their maximum velocities allowed in the sections of the line, this greedy schedule is then converted into an efficient pacing schedule, where the optimal pacing of trains is established in order to save energy. To obtain the optimal velocities, an average is calculated over the velocities of the sections that the train will travel, and also ensuring that the times of arrival previously calculated will remain the same.

The computational experiments showed that this algorithm is very moderate in computational effort comparing with other programming formulations. It also showed that it can modify into a strategy of optimal pacing velocities without affecting the time efficiency ratio.

2.2.5 Re-scheduling with train speed coordination

Andrea D'Ariano *et al.* [D'Ariano07a] introduced a variable-speed dispatching system that can control railway traffic more realistically. Acceleration and deceleration times were modelled considering the constraints of the signalling system and the rolling stock characteristics.

This system is composed of three parts: data loading, which collects data from the field; a conflict detection and resolution procedure with fixed-speed profiles, which has to solve the train scheduling problem; and a variable speed model that iteratively checks if the train speed profiles are acceptable.

More specifically, in the conflict detection and resolution procedure, the train scheduling problem was formulated as a job-shop problem with blocking and no-wait constraints. This type of approach uses the alternative graph [Mascis02] as the model structure. Each job (train) must pass through a prescribed sequence of machines (block sections). The passing of a train trough a particular block section (an operation) is a node of the alternative graph. A fixed arc represents the running time of a train trough the block section (solid arrows in Figure 2.3). Whenever two jobs require the same resource, there is a potential conflict. In this case, one of the pair of the alternative arcs (dashed arrows in Figure2.3), representing the minimum time headway between the associated trains has to be selected. By the inspection of the alternative graph, one can detect conflicts very rapidly. If the selection of an alternative arc increases the starting time of the following operation, a conflict between the two considered trains has been detected.

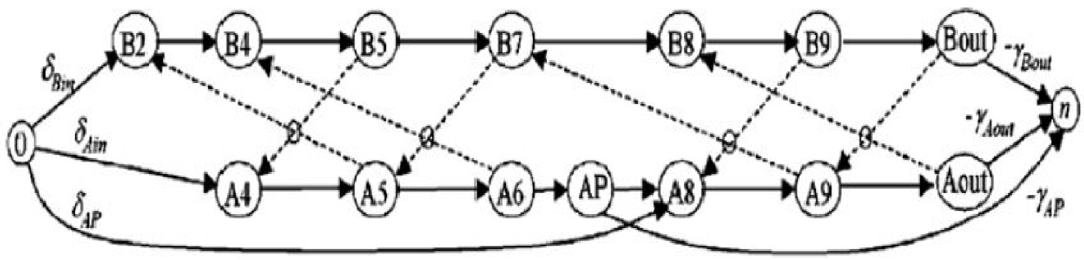


Figure 2.4 The alternative graph for the example with two trains (extracted from [Mascis02b])

For the conflict resolution three different classes of algorithms were used: simple dispatching rules (FIFO and FOFI), a greedy heuristic called AMCC (Avoid Most Critical Completion Time) based on the alternative graph, and a branch-and-bound algorithm that finds an optimal solution for fixed-speed profiles.

As for the variable speed model, it completes the alternative graph model in terms of signalling, which only considers red or green aspects. When a train faces a yellow signal, speed adjustments are made in the alternative graph and a new feasible solution is searched. This model is then able to set up a schedule without conflicts and ensure minimum distance headway between trains while keeping acceptable speed profiles.

Computational tests based on a Dutch rail network showed the feasibility of this system, even for large problems. It took the computer approximately 200 seconds to solve a problem with 54 trains and a time horizon of one hour.

Chapter 3

Model Description

Chapter 3 intends to present the reader with a first perspective of the developed model. To achieve this goal, this chapter will present first the system architecture as well as its relation with the dispatching system. Then, in section 3.2 all the model's assumptions and necessary inputs are defined. In section 3.3, the problem is formulated mathematically and all the constraints are presented. Following is a detailed explanation of all the conflicts solutions, so that in the final section, the two proposed solutions are explained.

3.1 System Architecture

The developed model can be classified, according to Chapter 2, as a variable-speed decision support system in real time. It is not supposed to replace the dispatcher but to help him in taking the best possible solutions. In other words, this model is a useful tool that allows train dispatchers to foresee the consequences of their decisions and also provides them with other feasible and probably better solutions.

The system architecture is presented in Figure 3.1. It shows how the DSS will be incorporated in the dispatching process and also illustrates the type of information that will be interchanged with the dispatcher. Inside the DSS block are represented its two main functions: the conflict detection and the conflict resolution blocks.

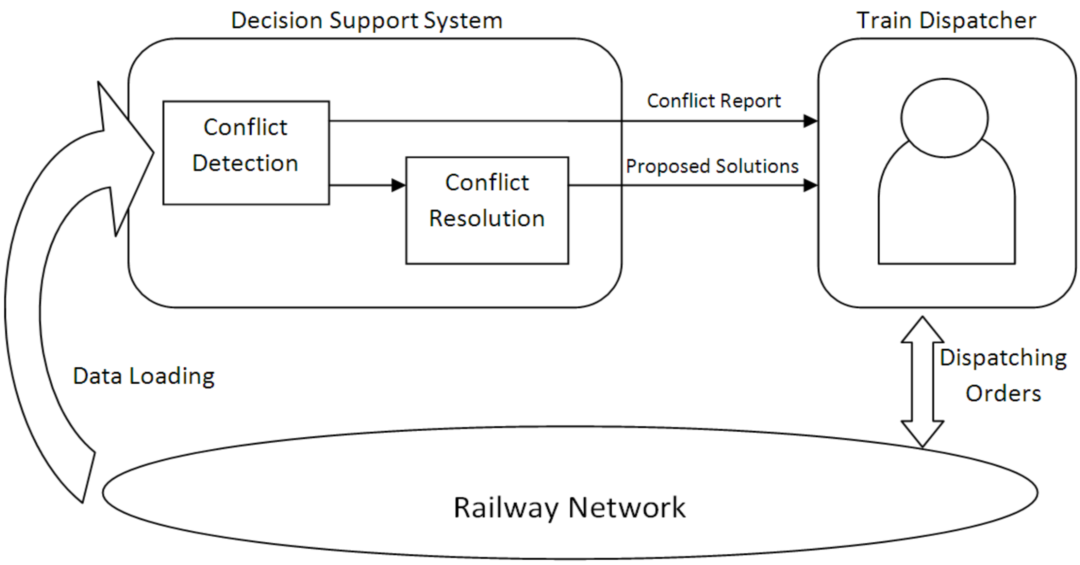


Figure 3.1 Train dispatching system architecture.

3.2 Problem Definition

Being railway scheduling such a rich and complex problem, it is necessary to define all the model's limitations, assumptions and inputs.

This model considers a single railway line that serves trains travelling in both directions. The railway is formed by track segments, which make the connection between all the meetpoints, as shown in Figure 3.2. In this context, meetpoints include not just stations, but also sidings or any location where two trains may cross simultaneously. So, for this model, trains are only able to meet or pass at meetpoints. As it is represented in Figure 3.2, train directions will be defined as inbound for trains going from right to left and outbound otherwise. Meetpoints and track segments are numbered in the outbound direction.

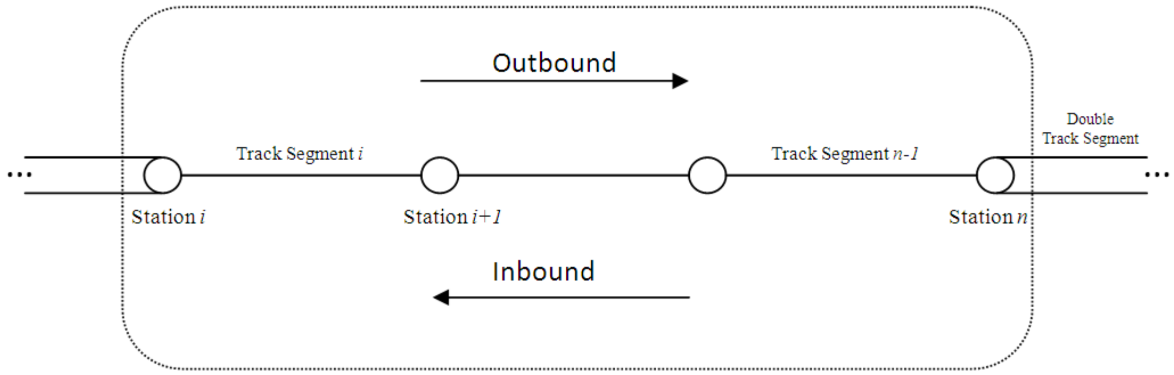


Figure 3.2 Model Railway Line Topology.

The safety technology considered is the fixed block signalling system. Therefore, trains can follow each other on a track segment with minimum safety headway. Considering this, the model also assumes that trains depart from stations as soon as possible, or in other words, as soon as the following block clears. This policy concerning trains following each other will be very important in the resolution of the pass conflicts that will be explained further ahead in Section 3.4.

Only three different types of trains are considered in the rest of this thesis but the model is valid independently of how many train types there is. The considered train types are:

- Fast Passenger Train
- Slow Passenger Train
- Freight Train

This order of presentation is also the order of priorities between them. Fast Passenger Trains have priority 1, which is the highest, and Slow Passenger Trains and Freight Trains have priorities 2 and 3, respectively.

The first and last meet points, of the considered railway network, do not have to be terminal stations. They can be the interface from single line segments to double track segments or even a denser rail network, as it happens in most cases. Anyway, the safety time intervals between arrivals and departures in these stations will not be checked.

In order to make this model more realistic, all meet points have limited capacity. This is a very important aspect that makes a big difference in the quality of the final solution. Very few models take capacity into consideration.

For simplicity, the rest of the thesis does not explicitly consider acceleration and deceleration time losses.

In addition to these model definitions, the following model inputs are enumerated:

- It is assumed that the location and speed of all the trains in the network is known at all times. This feature is essential in order to perform a proper supervision of the network and detect conflicts as early as possible.
- Being this a re-scheduling model, it is necessary to have an initial conflict free schedule. This schedule will also be considered as the initial optimum.
- Minimum dwell times of the trains at stations and their running times for each track segment are needed in order to verify the restrictions described in the following section.
- Train priorities have to be given for each train. These priorities influence the decision criteria of the produced solutions.
- Maximum capacities for all meet points are obviously also necessary.

In conclusion, a conflict is said to occur when two trains meet or pass at a segment track, when they arrive and depart from stations without the minimum safety intervals and also when a train arrives at a station that is full.

3.3 Mathematical formulations

The meet and pass problem, is essentially an optimization problem, subject to a several number of constraints. Therefore, it can be described mathematically. According to the definitions made in the previous section, the respective restrictions are now presented in a formal manner. The selected optimization criterion was the minimization of the total weighted tardiness.

Objective function:

$$\min Z = \min \sum_{i=1}^{n} w_i \ \max\{ 0 , (a_i^{m_i} - \alpha_i^{m_i})\} \tag{1}$$

subject to

Free running time constraints:

$$r_i^k \geq \tau_i^k, \qquad \forall i \in I, \qquad k = 1, 2, \dots, m-1. \tag{2}$$

Consecutive departure and arrival constraints:

$$f_i^k \geq s_i^k + \tau_i^k \ , \qquad \forall i \in I \ , \qquad k = 1, 2, \dots, m-1. \tag{3}$$

Minimum dwell time constraints:

$$\boldsymbol{s_i^u \geq \omega_i^u}\,, \qquad \forall \boldsymbol{i} \in \boldsymbol{I}, \qquad \boldsymbol{u = 1, 2, \dots, m.} \tag{4}$$

Headway constraints on arrival times at stations:

$$\boldsymbol{a_i^u \geq a_{i'}^u + g_u} \;\oplus\; \boldsymbol{a_{i'}^u \geq a_i^u + g_u}\,, \quad \forall \boldsymbol{i, i' \in I}, \qquad \boldsymbol{i \neq i'}, \qquad \boldsymbol{u \in U}. \tag{5}$$

Meet condition:

$$\boldsymbol{d_i^{u+1} \geq a_{i'}^{u+1} + g_u} \;\oplus\; \boldsymbol{d_{i'}^u \geq a_i^u + g_u} \quad, \forall \boldsymbol{u \in U}\,, \qquad \boldsymbol{i \in I_i, i' \in I_0} \tag{6}$$

Pass Condition:

$$(\boldsymbol{d_i^u \leq d_{i'}^u + h^k} \;\wedge\; \boldsymbol{a_i^{u+1} \leq a_{i'}^{u+1} + h^k}) \;\oplus\; (\boldsymbol{d_{i'}^u \leq d_i^u + h^k} \;\wedge\; \boldsymbol{a_{i'}^{u+1} \leq a_i^{u+1} + h^k}), \quad , \forall \boldsymbol{u \in U}\,, \qquad \boldsymbol{i, i' \in I_o} \tag{7}$$

Constraint (7) has an analogous formulation for the inbound case that will be omitted for the sake of simplicity.

Meetpoint capacity limits:

$$\boldsymbol{S^u \leq C^u}\,,\;\; \forall \boldsymbol{u \in U} \tag{8}$$

The notation of parameters and variables is shown below in Table 3.1.

Table 3.1 Parameters and variables used in the mathematical formulations.

Symbol	Definition
i	Train index
u	Meetpoint index
m_i	Terminal meetpoint of train i
k	Segment index
I	Set of trains, $\lvert I \rvert = n$, $I = I_i \cup I_o$, $I_i \cap I_o = \{\emptyset\}$
I_o	Set of outbound trains, I_o
I_i	Set of inbound trains, I_i
S^u	Number of trains at meetpoint u, $S \subset I$
C^u	Maximum capacity of meetpoint u
U	Set of meetpoints, $\lvert U \rvert = m$
K	Set of segments, $\lvert K \rvert = m - 1$
r_i^k	Running time for train i at segment k
τ_i^k	Minimum allowed running time for train i at segment k
s_i^u	Dwell time
ω_i^u	Minimum dwell time
d_i^u	Departure time of train i at meetpoint u
a_i^u	Arrival time of train i at meetpoint u
α_i^u	Scheduled arrival time of train i meetpoint u
s_i^k	Start time of train i at segment k
f_i^k	Finish time of train i at segment k
w_i	Weighted priority of train i
h^k	Minimum headway between arrival and departure times of two consecutive trains at segment k
g_u	Minimum headway between arrival and departure times of two consecutive trains at meetpoint u

Constraint (2) ensures that no train will travel along a segment track faster than its maximum allowed running profile. Constraint (3) guarantees that an arrival at one station is always later than the departure from the previous one. Constraint (4) imposes that trains cannot leave the station until their minimum dwell time is completed. These dwell times are necessary to load or unload passengers and freight at stations. Inequality (5) verifies if the safety time intervals between departures and arrivals at a station are respected. For security reasons, such as slip prevention, there can be no simultaneous departures or arrivals at a station, even if they are towards different segments. As for constraints (6) and (7), they are the most important ones since they check if no meet or pass occurs on a segment track. More specifically, in inequality (6), it states that two opposing trains can cross at a station after the first train arrival and the due safety time interval elapses. Restriction (7) defines that one train can follow another on a segment track, if the safety headway is respected throughout the entire segment, until they arrive at the following station. Finally, constraint (8) assures that no station capacity is exceeded.

In conclusion, if a schedule does not respect all of the above restrictions, it means that a conflict was detected and this schedule is considered unfeasible.

However, verifying the meet and pass constraints between all of the trains is unnecessary. In order to explain this statement, it is helpful to rewrite restrictions (6) and (7) in relation to the segment instead of the stations.

Meet condition:

$$\boldsymbol{f_i^k < s_{i'}^k \ \oplus \ s_i^k > f_{i'}^k \ , \ \forall k \in K \, , \qquad i, i' \in I} \tag{9}$$

Pass Condition:

$$\boldsymbol{s_i^k < s_{i'}^k \ \wedge \ f_i^k < f_{i'}^k \ \oplus \ s_i^k > s_{i'}^k \ \wedge \ f_i^k > f_{i'}^k \ , \ \forall k \in K \, , \qquad i, i' \in I} \tag{10}$$

The safety variables g_u and h^k are not considered for simplification matters.

Before the introduction of these main results, it is necessary to make the following assumptions:

- The trains are considered to be ordered by their entrance times (s_i^k) at a given segment k . This can be represented by the following equation:

$$\boldsymbol{s_i^k < s_{i+1}^k < s_{i+2}^k \ , \quad \forall \ i = 1, \dots, n-2} \tag{11}$$

  Condition (11) must always be verified for the theorems to be true.

- There is also one condition which is always valid since train cannot move instantaneously:

$$\boldsymbol{s_i^k < f_i^k} \tag{12}$$

- In both proofs, train i is always considered an outbound train. The case when i is inbound, is analogous with the outbound situation, and so it will not be demonstrated, without loss of generality.

Theorem 3.1 – If train i does not collide with train $i+1$, and train $i+1$ does not collide with train $i+2$, then train i cannot collide with train $i+2$.

Proof:

Case 1) $i+1$ is an outbound train

So, if train i does not collide with train $i+1$, then:

$$f_i^k < f_{i+1}^k \tag{13}$$

And, if train $i+1$ does not collide with train $i+2$, then there can be two cases:

Case 1.1) $i+2$ is an outbound train

If train $i+1$ does not collide with train $i+2$, then:

$$f_{i+1}^k < f_{i+2}^k \tag{14}$$

And so, from (13) and (14):

$$f_i^k < f_{i+2}^k \tag{15}$$

From conditions (11) and (15), it can be concluded that, in this case, train i does not collide with train $i+2$.

Case 1.2) $i+2$ is an inbound train

If train $i+1$ does not collide with train $i+2$, then:

$$f_{i+1}^k < s_{i+2}^k \tag{16}$$

And so, from (13) and (16):

$$f_i^k < s_{i+2}^k \tag{17}$$

From conditions (11) and (17), it can be concluded that, in this case, train i does not collide with train $i+2$.

Case 2) $i+1$ is an inbound train

So, if train i does not collide with train $i+1$, then:

$$f_i^k < s_{i+1}^k \tag{18}$$

And, if train $i+1$ does not collide with train $i+2$, then there can be two cases:

Case 2.1) $i+2$ is an outbound train

If train $i+1$ does not collide with train $i+2$, then:

$$f_{i+1}^k < s_{i+2}^k \tag{19}$$

And so, from conditions (18) and (19):

$$f_i^k < s_{i+1}^k < f_{i+1}^k < s_{i+2}^k \iff f_i^k < s_{i+2}^k \tag{20}$$

With conditions (11) and (20), it can be concluded that, in this case, train i does not collide with train $i+2$.

Case 2.2) $i+2$ is an inbound train

If train $i+1$ does not collide with train $i+2$, then:

$$f_{i+1}^k < f_{i+2}^k \tag{21}$$

And so, from equations (18) and (21):

$$f_i^k < f_{i+2}^k \tag{22}$$

From conditions (11) and (22), it can be concluded that, in this case, train i does not collide with train $i+2$.

■

Now that all cases are verified, one must conclude that the non-conflict condition has the transitivity property. Hence, it can be generalized through all trains in the segment. If there are no conflicts between consecutive train pairs then, one must conclude that there are no conflicts in the segment at all.

Theorem 3.2 – If there is a conflict between train i and train p , $p \geq i+2$, then there is also a conflict between trains i and $p-1$, or between trains $p-1$ and p.

Proof:

Case 1) p is an outbound train

If there is a conflict, then:

$$f_i^k > f_p^k \tag{23}$$

Case 1.1) $p-1$ is an outbound train

For train $p-1$ not to collide with train i, then:

$$f_i^k < f_{p-1}^k \tag{24}$$

For train $p-1$ not to collide with train p, then:

$$f_{p-1}^k < f_p^k \tag{25}$$

Which means that,

$$\begin{cases} f_i^k < f_{p-1}^k < f_p^k \\ f_i^k > f_p^k \end{cases} <=> imp. \tag{26}$$

Thus in this case, train $p-1$ will either collide with train i or train p.

Case 1.2) $p-1$ is an inbound train

For train $p-1$ not to collide with train i, then:

$$s_{p-1}^k > f_i^k \ <=> \ f_{p-1}^k > s_{p-1}^k > f_i^k \ <=> \ f_{p-1}^k > f_i^k \tag{27}$$

For train $p-1$ not to collide with train p, then:

$$f_{p-1}^k < s_p^k \ <=> \ f_{p-1}^k < s_p^k < f_p^k <=> \ f_{p-1}^k < f_p^k \tag{28}$$

Which means that with conditions (23), (27) and (28):

$$\begin{cases} f_i^k > f_p^k \\ f_{p-1}^k > f_i^k \\ f_{p-1}^k < f_p^k \end{cases} <=> imp. \tag{29}$$

Thus in this case, train $p-1$ will either collide with train i or train p.

Case 2) p is an inbound train

If there is a conflict, then:

$$f_i^k > s_p^k \tag{30}$$

Case 2.1) $p-1$ is an outbound train

For train $p-1$ not to collide with train i, then:

$$f_{p-1}^k > f_i^k \tag{31}$$

For train $p-1$ not to collide with train p, then:

$$f_{p-1}^{k} < s_{p}^{k} \tag{32}$$

But these conditions together with condition (30):

$$\begin{cases} f_i^k > s_p^k \\ f_i^k < f_{p-1}^k \\ f_{p-1}^k < s_p^k \end{cases} <=> imp. \tag{33}$$

Thus in this case, train $p-1$ will either collide with train i or train p.

Case 2.2) $p-1$ is an inbound train

For train $p-1$ not to collide with train i, then:

$$s_{p-1}^{k} > f_{i}^{k} \tag{34}$$

But in this case, with condition (30):

$$s_{p-1}^{k} > f_{i}^{k} > s_{p}^{k} \;<=>\; s_{p-1}^{k} > s_{p}^{k} \tag{35}$$

This equation is impossible because it violates the initial condition referred in (10). Hence, in this case, train $p-1$ will either collide with train i or train p.and so, validating the theorem.

■

What Theorem 3.2 states is that any conflict between two trains, which are apart in terms of their entering order, implies a conflict between two consecutive trains, or a conflict between closer trains. Propagating the result, the consequence is that the conflict will compulsory end up between two consecutive trains.

Corollary 3.3 - In order to conclude about the existence, or non-existence of conflicts, in a given track segment, it is only necessary to check for conflicts between consecutive trains, in terms of their entering order.

Proof:

From the conclusion of Theorem 3.1, one may say that, if there are no conflicts between consecutive trains on a given track segment, then necessarily there are no conflicts between any of those trains. Therefore this condition assures the non-existence of conflicts.

From Theorem 3.2, it is concluded that, any conflict between non-consecutive trains, means that there is also a conflict between consecutive trains. Hence, this conclusion guaranties that all the existing conflicts will be detected.

□

The importance of these results is the complexity reduction in the problem of conflict detection, which was a combinatorial problem and now becomes a linear one. More specifically, for a schedule with n trains, instead of making all the (C_2^n) comparisons, it only needs $(n-1)$ comparisons. Furthermore, this algorithm is used recursively in the search for the optimal solution, which makes the reduction impact even bigger.

3.4 Solving Conflicts

Now that all the model's restrictions are properly defined, the next step is to explain how to solve the detected conflicts. Conflicts are grouped into four different types:

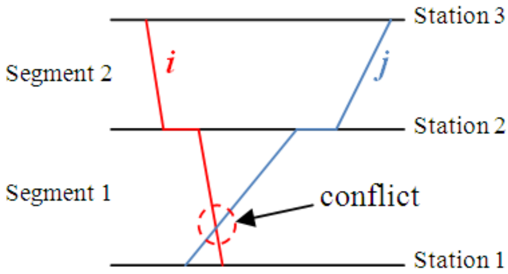
- Meet conflict
- Pass conflict
- Safety intervals at the station
- Capacity conflict

For each type of conflict, the range of possible solutions is presented next. Evidently, there are infinite possible solutions, but the optimal ones are the only ones that are worth considering. Also, the decision criteria of which train to stop and which train to pass will be explained only in the next section.

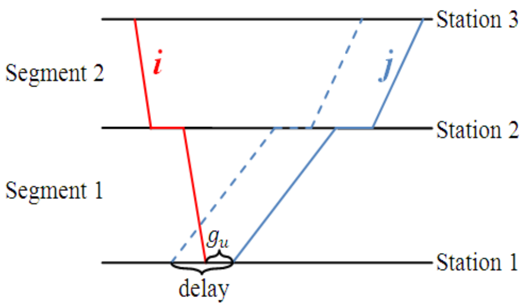
- **Meet Conflict**

This first type of conflict involves two trains and so, there are only two valid solutions to consider. These solutions are either to stop the inbound train or the outbound train. In Figure 3.3(a) is an example of a meet conflict, between trains *i* and *j* at segment track 1, followed by its two solutions in Figure 3.3(b) and (c).

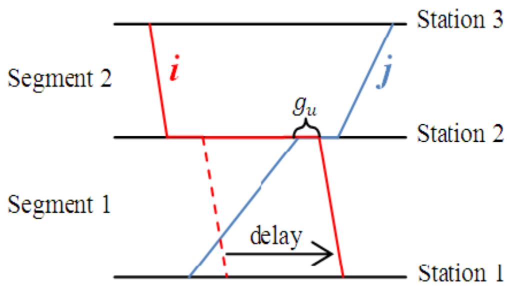
In Figures 3.3(b) and (c) is also represented the introduced delay and the safety time interval g_u . The time interval introduced can be g_u or h^k, the chosen interval is the one which has the bigger value. This is also valid for the pass conflict.



(a) Meet conflict



(b) Train *i* first



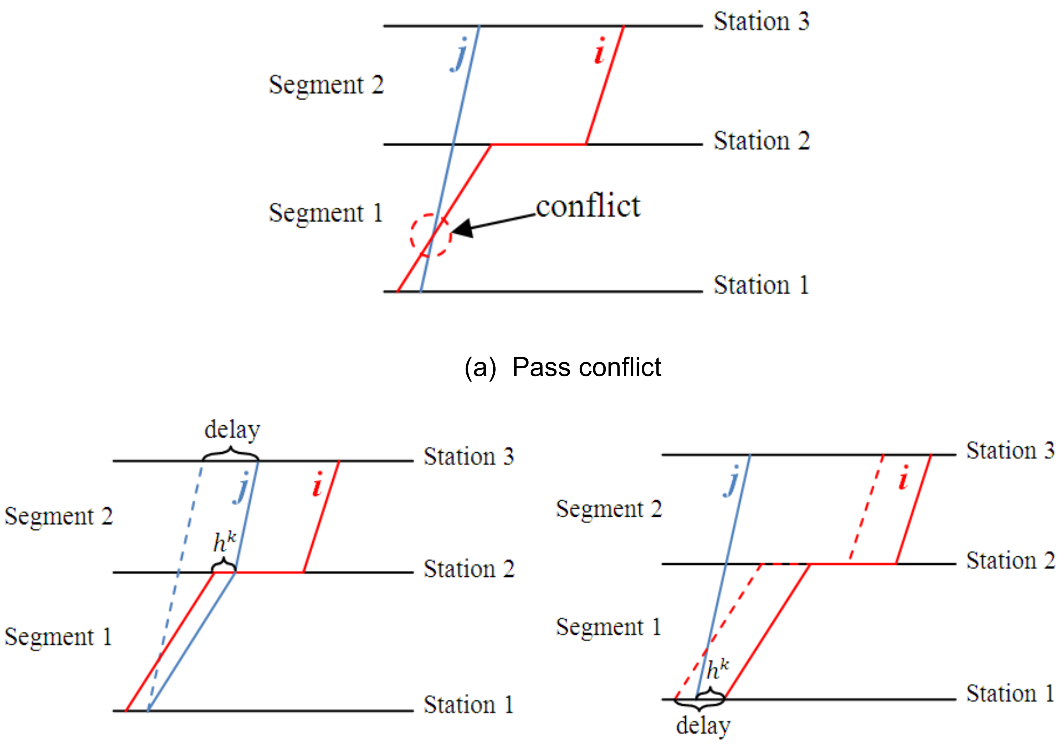
(c) Train *j* first

Figure 3.3 Resolution of a Meet conflict.

Apparently, the solution shown in 3.3(b) seems better than the one in 3.3(c) because the resulting delay is smaller. However, train priorities would also have to be considered so as to make a proper weighted decision. In meet conflicts, delays are always introduced in the dwell times of trains at the stations but the same does not happen in the type of conflict explained next.

- **Pass conflict**

The pass conflict is not as simple as the first one because, in this case, running times also have to be taken into account. As it was referred in Section 3.2, it is assumed that trains leave their stations as soon as possible, in order to achieve the minimum possible delay. Therefore, if a faster train follows a slower one, its running time will have to be decelerated in order to avoid red signals. The following Figure 3.4 illustrates the Pass conflict between fast train *j* and slow train *i.*



(a) Pass conflict

(b) Train *i* first

(c) Train *j* first

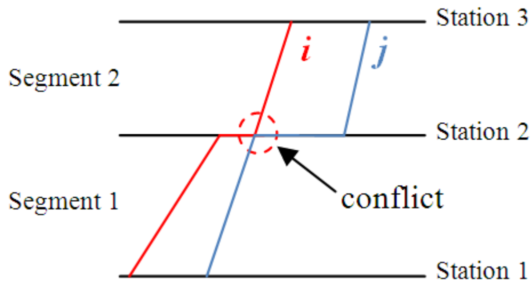
Figure 3.4 Resolution of a Pass conflict.

In Figure 3.4.(b) is a good example of an affected running time. Train *j* had to be slowed down so as to arrive at station 2 without having to stop in the middle of the segment track, and wait for train *i* to clear the following block.

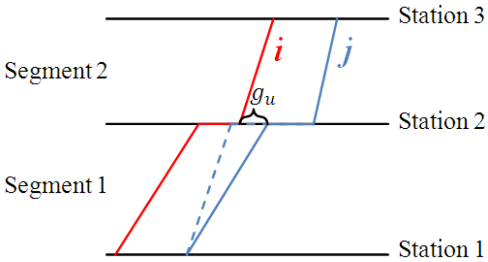
- **Safety intervals at stations**

As for the arrivals and departures at stations there are three different situations of conflict: two arrivals, two departures, one arrival and one departure. Because they are very similar situations, only the case with one arrival and one departure will be shown in Figure 3.5.

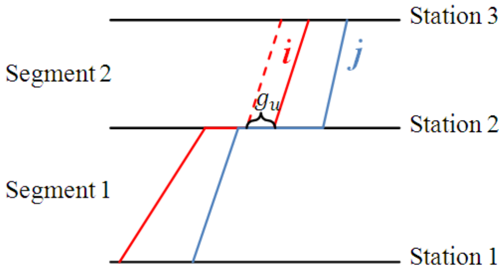
In this case, there is not much to add considering that this schedule changes are the same as the changes in the meet and pass conflicts, i.e. dwell time extensions and slower running times.



(a) Station conflict



(b) Train *i* first



(c) Train *j* first

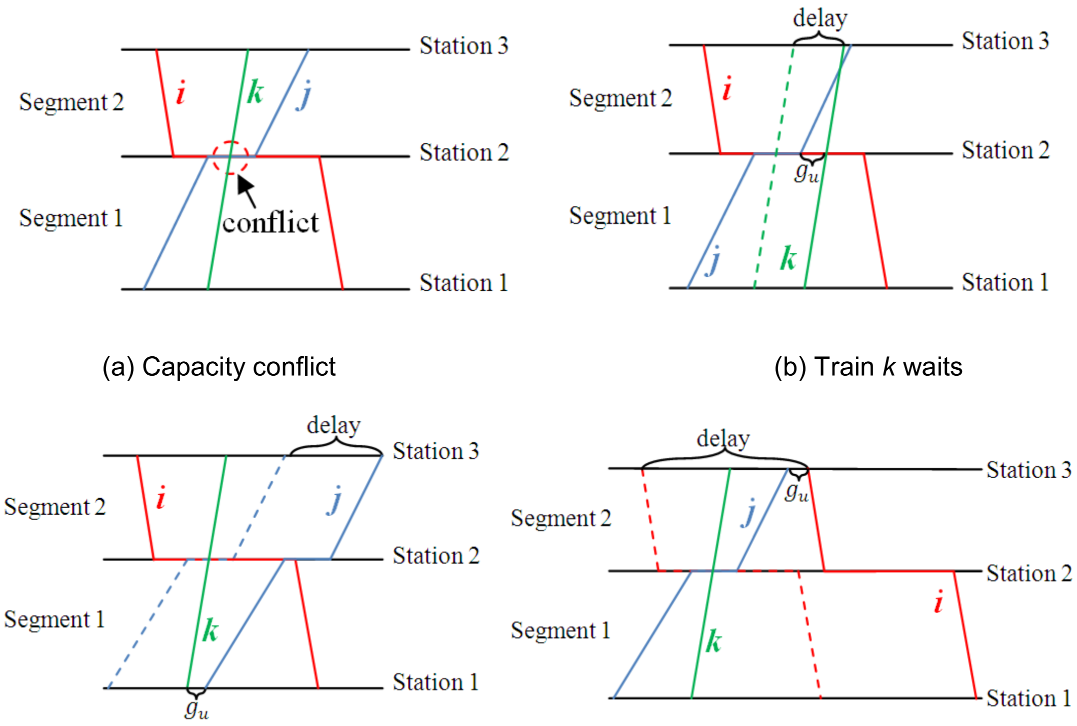
Figure 3.5 Resolution of a conflict at the time intervals at a station.

In this case, there is not much to add considering that this schedule changes are the same as the changes in the meet and pass conflicts, i.e. dwell time extensions and slower running times.

- **Capacity conflict**

The Capacity conflict is by far the most complex of them all. The train chosen to wait for the full station to be available again, is not necessarily the last one in the schedule to arrive. In fact, all of the trains at the station, when the conflict is detected, are candidates to be re-scheduled. Hence, one can say that a conflict in a station with capacity *N*, will have *N + 1* possible solutions. The policy in this situation is to hold one of the trains at the previous station until the first train at the full station departs. To help understand this resolution better, the following example in Figure 3.6 shows a capacity conflict in station 2 with capacity for two trains.

The solution in Figure 3.6 (d) exemplifies another important aspect of this model. Despite station 2 become available immediately after train *j* departs, train *i* waits an extra time for train *j* to arrive at station 3 and hence avoiding a meet conflict. As for Figure 3.6 (b), this is a solution that solves the capacity conflict but generates another future pass conflict between trains *j* and *k*. Even so, this solution is still completely valid.



(a) Capacity conflict

(b) Train *k* waits

(c) Train *j* waits

(d) Train *i* waits

Figure 3.6 Resolution of a capacity conflict

3.5 Towards a feasible solution

Now that the resolutions for each conflict type are defined, it is necessary to select them properly in order to generate the best possible solutions. In the previous section, conflicts were studied as isolated cases, but reality is not that simple. In a complex railway schedule, if one conflict arises, its solution will probably originate more conflicts and the resolution process has to be repeated until the schedule is feasible again.

Now, the objective of this Thesis is not only to provide the dispatcher, with feasible solutions, in a small amount of time, but also to try to obtain the optimal solution if possible. Therefore, two solutions were developed in order to satisfy these requirements: the heuristic approach and the optimum solution. An explanation of each one of the approaches will be presented next.

- Heuristic Approach

Taking into account that this program will work in real-time, there is the need to generate solutions as fast as possible to help the dispatcher. In Chapter 2, the dispatchers' behaviour was explained and classified as myopic and sub-optimal. Nevertheless, these decisions have proven to be effective and keep the railroad safe. Therefore, the objective of this algorithm is to find a feasible solution for the conflict in a short amount of time, and also to reproduce the dispatchers' behaviour in the resolution process. It can be useful to imitate train dispatchers so as to compare the quality of their solutions with the optimal ones.

The proposed decision criterion is based mainly on train priorities and in the FOFI rule. For every conflict that involves only two trains, which means all conflict types except for the capacity conflict, the decision criterion works as follows:

-If one train has higher priority than the other, that's the train that will always go first and the other will be delayed.

-If both trains have the same priority, then the FOFI rule is applied. This rule will check which one of the trains is able to solve the conflict in a faster way. In other words, this rule verifies which decision generates de smallest delay on the stopped train.

As for the capacity conflicts, there are *N + 1* trains to consider and the procedure is as follows:

-Between all the trains involved in the conflict, choose the train with lowest priority in order to be delayed.

-In case there is more than one train with low priority, choose the last one to arrive at the station.

This algorithm will be explained with more detail in the next chapter.

- **Optimal Solution**

This optimal solution was developed to provide the dispatchers with optimal or near optimal solutions, better than the ones they usually take. To do so, a search tree is considered. The nodes of the search tree are the conflicts and the branches are the respective solutions. This means that from the first node to the leaf node is represented a feasible solution of the re-scheduling problem. Because the time available is a key element, the search mode adopted was the depth first search (DFS). The DFS allows reaching the solutions without having to search the entire tree as it would happen in a breath first search.

To better illustrate how this procedure works, the following Figure 3.7 is presented.

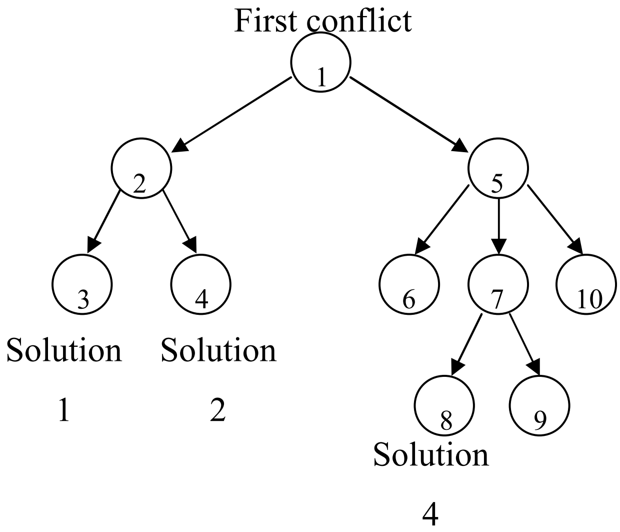


Figure 3.7 Illustration of the search tree.

The numbers inside each node represent the order of the search. For the sake of clarity, only four solutions are indicated, but of course in this example, there are six feasible solutions. In node 5 there are three possible options which mean this is a capacity conflict at a station with capacity for only two trains.

As it was referred before in the introduction chapter, this problem is NP-complete, meaning it increases exponentially with the number of conflicts. Therefore, there is the need to adopt some techniques that allow saving some time. The branch-and-bound was one of the chosen techniques. In addition to this technique, there is also the introduction of a time horizon and a limited time for the search.

More details concerning the implementation of this solution will be discussed on the next chapter.

Chapter 4

Program Implementation

This chapter intends to make a full description of the developed program. Before that, the first section aims to motivate the choice of MATLAB as the programming language. Then, section 4.2 proceeds with the overall structure and in the following sections the explanation of its main modules. The input and output data are described in sections 4.5 and 4.6. Finally, the application and its features are presented in the last section.

4.1 Programming Language Selection

This application was developed in MATLAB. The main motivation for this choice was the developer's experience on this programming language in contrast with a lack of experience in other languages such as C++ or Java.

As the name indicates MATLAB (MATrix LABoratory) is a language specially oriented to the manipulation of matrices, and can perform numerical calculations faster than for example in C. In view of the fact that the timetables are matrices and their manipulation would be a main part of this program, this represented an important advantage.

Because the graphic creation was a decisive issue in order to represent the train diagrams, MATLAB also contains many libraries that allow an easy and versatile graphic manipulation.

And last, being the intent of this program to interact with the dispatcher, the Guide User Interface presented in MATLAB, allows a wide range of possibilities to create a more user friendly application.

4.2 Program Overview

In this program, there are two distinct parts: the Conflict Detection block and the Conflict Resolution block. These parts are indicated below in Figure 4.1 within the dashed rectangles.

- The Conflict Detection block is responsible for the surveillance of the railway network. It is also in charge of updating all the necessary data, so that the conflict detector can work with valid information. This loop is supposed to keep running until a conflict is detected.

- The Conflict Resolution is evidently the part responsible for solving the detected conflicts. This part is much more complex than the previous one. It is divided into two main blocks which are the heuristic solution and the optimal solution. These two algorithms that were already introduced in Chapter 3 will be explained in detail further on. The Conflict Resolution part is also in charge of reporting the achieved solutions to the dispatcher.

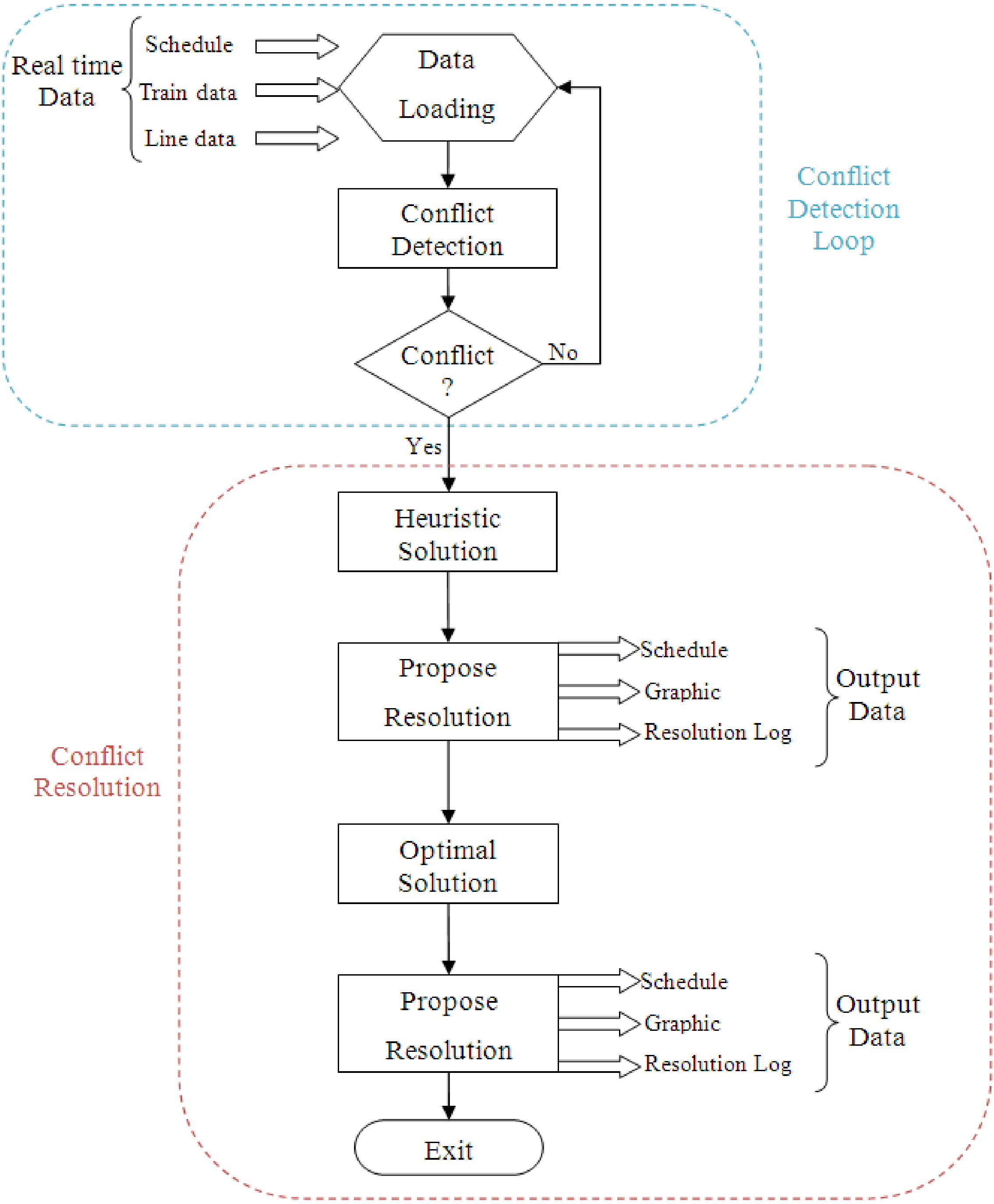


Figure 4.1 Programs' Structure.

4.3 Conflict Detection

The detection of conflicts is executed by the inspection of the network timetable. Two scans at the timetable are performed, the first one tests out for meet and pass conflicts, as for the second scan, it checks for the safety intervals at stations and for capacity conflicts.

Here in Table 4.1 is an example of a timetable with all the arrival and departure times of all the trains from every meetpoint in the railway.

Table 4.1 Example of a timetable.

	Meetpoint1		Meetpoint2		Meetpoint3	
	ARRIVAL	DEPARTURE	ARRIVAL	DEPARTURE	ARRIVAL	DEPARTURE
Train1	0	10	45	54	62	300
Train2	10	15	30	50	65	73
Train3	5	30	36	62	68	120
Train4	146	149	140	141	104	136
Train5	85	300	72	78	0	66
Train6	100	190	110	160	100	105

Entrance times at Segment2

The double line between Train 3 and Train 4 is separating the outbound trains (Train 1 to Train 3) from the inbound trains (Train 4 to Train 6). As explained in Chapter 3, remember that the meetpoints and segment tracks are numbered in the outbound direction. The times presented in the table are in minutes. More details concerning the input data will be discussed in Section 4.5.

Now, the first scan sorts the train order for each track segment, and then for each pair of trains, according with the conditions from Corollary 3.3, it verifies the meet and pass constraints. In other words, the algorithm compares the safety constraints between two consecutive trains entering a given track segment. This is the application of the two theorems explained in the previous chapter.

Considering the example of Table 4.1, the order of trains entering track segment 2 corresponds to the sort of the highlighted values within the red circles. In this case it would be as indicated in Table 2:

Table 4.2 Order of entrance at Segment Track 2.

	Segment Track 2	
	Entering times	Order
Train 1	54	2
Train 2	50	1
Train 3	62	3
Train 4	136	6
Train 5	66	4
Train 6	105	5

The scan would begin the checking between Train 2 and Train 1, then Train 1 and Train 3 and so on until the last pair, Train 6 and Train 4. This type of approach allows saving precious time checking for meet and pass constraints between all the trains in the timetable, and so reducing the complexity of the algorithm.

When a conflict is found, the scan does not stop because it might not be the earliest one to occur. It is intended in this program to solve conflicts as chronologically as possible. Although this may be not very accurate, the time of conflict is assumed as the instant when the first involved train leaves the meetpoint before the conflict. It is logical to think this way considering that this is the deadline to take a dispatching measure. After that instant, the options considered in Chapter 3 for conflict resolution are not feasible any more.

The second scan is very similar to the first one except that in this case, it is not about entering times in segment tracks, but entering and leaving times from meetpoints. The algorithm sorts out the arrival and departure times for one station and compares all the consecutive times checking if the safety time intervals are respected. Simultaneously, a train count for that same station is performed in order to detect capacity conflicts. The blue shaded columns in Table 4.1 will be the example for this scan.

Table 4.3 Sorting Meetpoint 1.

	Meetpoint 1	
	Arrival Order	Departure Order
Train 1	1	1
Train 2	3	2
Train 3	2	3
Train 4	6	4
Train 5	4	6
Train 6	5	5

4.4 Heuristic Solution

This algorithm is intended to be fast and effective as explained in Section 3.5. Therefore, the detected conflict is evaluated concerning only train priorities and the FOFI rule which is a sub-optimal and myopic rule. A flowchart explaining the solution procedure is presented below in Figure 2.

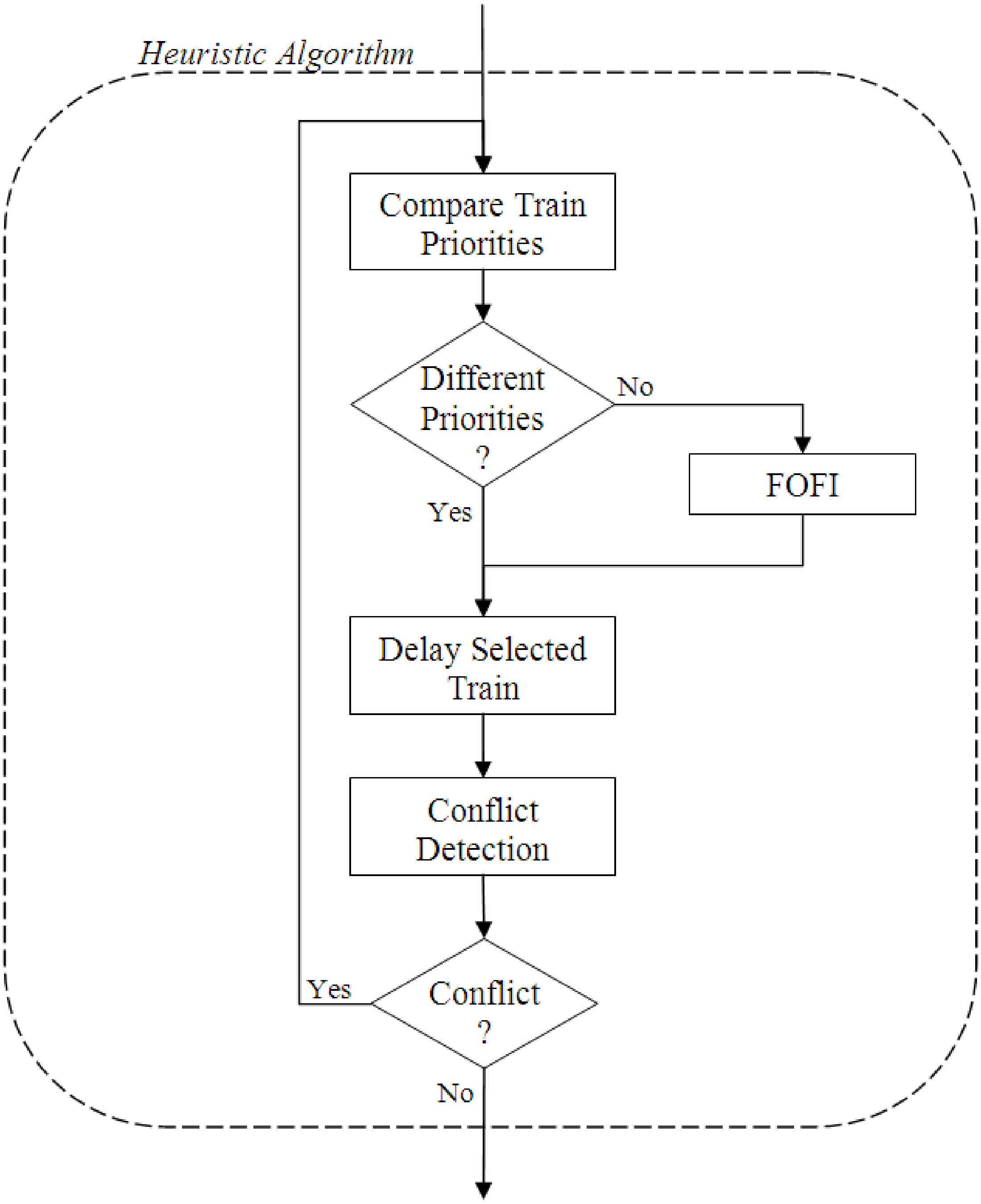


Figure 4.2 Flowchart of the Heuristic algorithm.

It was seen before, in Figure 4.1, that when this module is activated, it means that a conflict was already detected. So, the algorithm begins by comparing the priorities of the trains involved in the conflict, to check if one has a lower priority. If all of the involved trains have the same priority, the program performs the FOFI rule to see which option can cause de smallest delay. Having decided which train to delay, its schedule is re-arranged accordingly and always respecting future dwelling times. Next, it is necessary to check for more conflicts in the schedule. If another conflict is detected, the process is repeated again from the beginning. This cycle continues indeterminately until the schedule becomes free of conflicts, which means a solution was found.

4.5 Optimal Solution

The Optimal algorithm was developed with the objective of reaching an optimal solution. To do so, it should supposedly verify all the possible solutions and then choose the best one among them. Due to the enormous size that the search tree may reach in dense traffic railways, it is unnecessary and also impossible to store the entire tree in the computer memory and to search it in feasible time. The Depth First Search was then chosen because it does not require a lot of memory, and because it provides solutions faster even if they are not optimal. In this program, a stack was used to store the search tree, meaning only one route from the first node to the leaf node is in the computer's memory at a given instant.

Because time is a key factor in this search, there is the need to adopt some techniques to reduce the size of the search tree. One of them is the introduction of an Upper Bound search limit. The Upper Bound stops the depth search, whenever the considered solution is worse than the best one found until that moment. The quality of each solution is calculated by the cost function presented in Chapter 3. Another technique used to reduce the search tree is the time horizon. The conflict detector only reports conflicts earlier than the time horizon, therefore ignoring all the following future conflicts. This horizon enables the algorithm to prune a big part of the search tree and consequently saving precious time.

Even with these features, there are still schedules that require too much time to be optimally solved. Hence, the dispatcher has another possibility which is the Maximum search time. If this time is exceeded, the algorithm will stop the search and report the best solution found until that moment. Therefore, there can be two possible types of final solutions presented. When the search tree is completely explored before the time limit, then the best solution found is the optimal solution. Of course, this solution is optimal but disregarding all the conflicts that occur after the considered time horizon. The other type of solution happens when the maximum search time is exceeded and the search is not complete. In this case the solution is not optimal but can be classified as the best solution found.

In the next figure, the flowchart of this algorithm is presented.

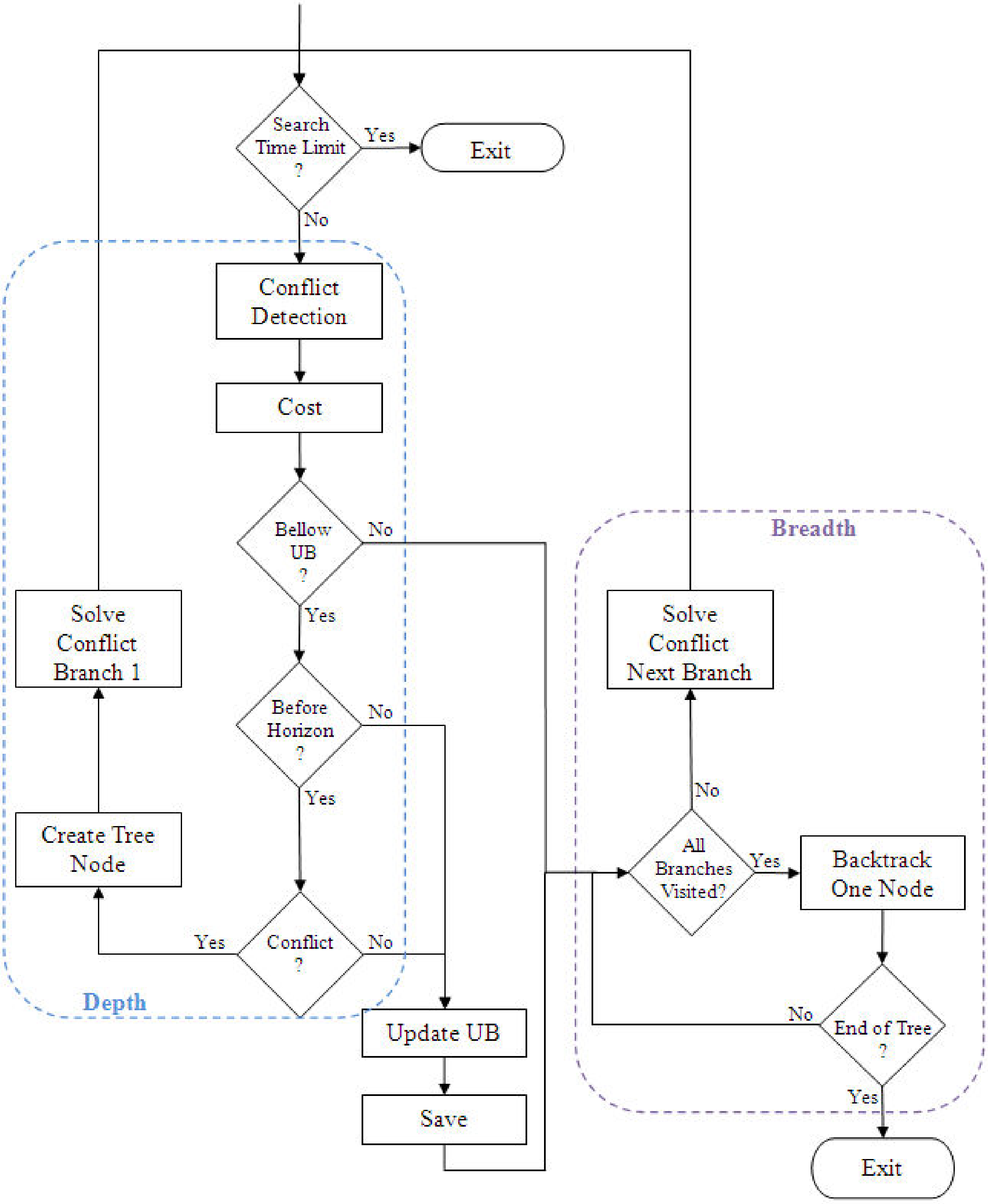


Figure 4.3 Flowchart of the Optimal algorithm.

The algorithm begins by searching the tree in depth that is highlighted in the blue dashed rectangle. This cycle continues until a solution is found which means that a leaf node was reached. If the solution is better than the current upper bound, the upper bound is updated and that solution is saved as the best one found yet. The next step is the breath search cycle represented inside the purple dashed rectangle. First it checks if there are any more branches to visit in the last node. In case there is a branch to visit, the respective solution is executed and the depth search starts again. If not, the algorithm backtracks one node and repeats the previous check.

4.6 Input Data

The necessary information for this program to execute is now presented. The Excel file containing information about the schedules, trains and lines is in the .xlsx format.

All the data is located in one single Excel file in the .xlsx format. This file contains three sheets where the first sheet contains the schedule, the second sheet describes the trains, and the third sheet describes the railway track.

4.6.1 Schedule Description

The schedule sheet must be named as “Schedule” and is presented as a timetable containing all arrival and departure times for every train at every meetpoint. Below, in Table 4.4 is an example of a timetable.

Table 4.4 Example of the Schedule sheet.

	Meetpoint1		**Meetpoint2**		**Meetpoint3**	
	ARRIVALS	**DEPARTURES**	**ARRIVALS**	**DEPARTURES**	**ARRIVALS**	**DEPARTURES**
101	0	10	15	15	22	1000
102	0	15	30	50	65	1000
103	0	30	36	42	48	1000
204	42	1000	36	37	0	32
205	65	1000	52	58	0	46
206	70	1000	10	50	0	5

The first column has to contain a list of all trains exactly in the same order as they appear in the train sheet, starting with the outbound trains and then the inbound trains. The table must also include the stations’ names in the first line and below the words ARRIVALS and DEPARTURES.

The times presented in the timetable are in minutes. The zero value means the train is at the station since the beginning of the simulation. If a train is going to stay at a given station until the end of the simulation period, a significantly large value is inserted (in the example this value was 1000).

4.6.2 Trains Description

In this second sheet, named “Trains”, is supposed to be all the necessary information about the trains in the schedule. This sheet is composed by two tables that are now described. As in Table 4.5, the number of inbound trains must be indicated first followed by the number of outbound trains. Then, in the Table 4.6, the most important characteristics are the train identification and the train priority. The train identification, like the one used by the real train dispatchers, is numeric and cannot contain characters. The train type is an integer number ranging from 1 to 3.

Table 4.5 Example of the first table in the Trains sheet.

Number of Inbound Trains	3
Number of Outbound Trains	3

Table 4.6 Example of the second table in the Trains sheet.

TRAINS DESCRIPTION		
ID	**TYPE**	**LENGTH**
101	1	1000
102	2	520
103	1	868
204	1	458
205	1	1000
206	3	940

The train length is also included in the file. Although it is not used in the program right now, it can be useful in future improvements.

4.6.3 Railway Description

The railway sheet must be named as "Railway", and includes information concerning stations and segments. Like in the Trains sheet, this sheet also has two tables, one for the stations description and another for the segments description.

Table 4.7 Example of the first table in the Railway sheet.

STATIONS DESCRIPTION			
NAME	**CAP**	**LENGTH**	**DISTANCE**
E1	12	1000	0
E2	3	1000	6
E3	8	1000	20

Table 4.8 Example of the second table in the Railway sheet.

SEGMENT DESCRIPTION
NAME
T1
T2

Just like in Table 4.7, the name of each station is provided in the first column followed by their capacities in the second column. The length of each station in the third column is not used in this program but like the trains length, it can be useful for future improvements. As for the distance in the last column, it indicates how far each station is from the first station of the railway line. This measurement is always made in the outbound direction.

The second table, Table 4.8, is about the segments data and has only one column indicating their identifications. This information is useful for the conflict report so as to help the dispatcher in tracing the conflicts in the graphic.

4.7 Output Data

The output information is made in three different ways. The most important one is the representation of each solution found in the form of train diagrams. The second is the creation of a new Excel file with the final schedules free of conflicts, and the third is the respective creation of a conflict report for the solutions found. These three formats are now presented in detail.

4.7.1 Train Diagrams

An example of a train diagram of this program is represented in Figure 4.4. These train diagrams follow the standard type just like it was explained in Chapter 2. However, there are some additional features now presented. The colours were introduced to represent train priorities being:

1. Green – Fast Passenger Train
2. Blue –Slow Passenger Train
3. Red –Freight Train

There are two lines for each train, a dashed line representing the initial route, and then the solid line with the final train schedule. With these two lines, it is easier to detect what changes were made in the initial schedule. The time horizon is also represented in the graphic as a vertical, light blue, dashed line.

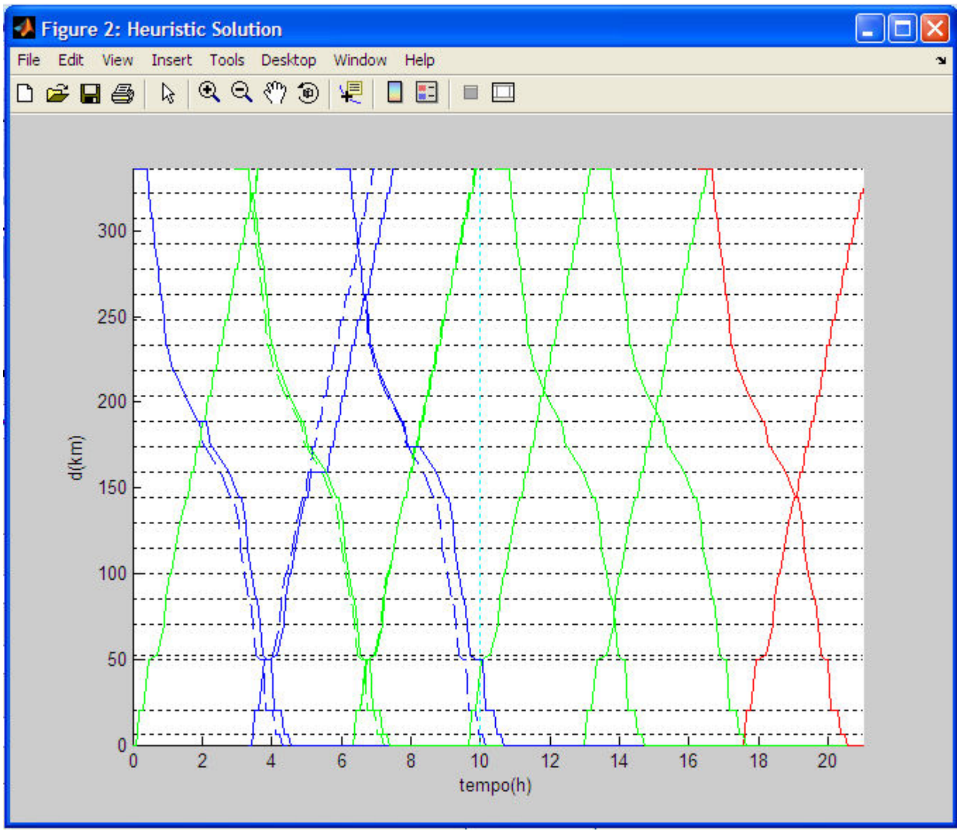


Figure 4.4 Example of the output of a train diagram.

4.7.2 Solution Schedules

For the recording of the final schedules, it is created a new Excel file with the same name of the input file but appended with the string “Solution_” before the file name. For example, if the input file has the name “Problem1.xlsx”, then the created output file will be “Solution_Problem1.xlsx”.

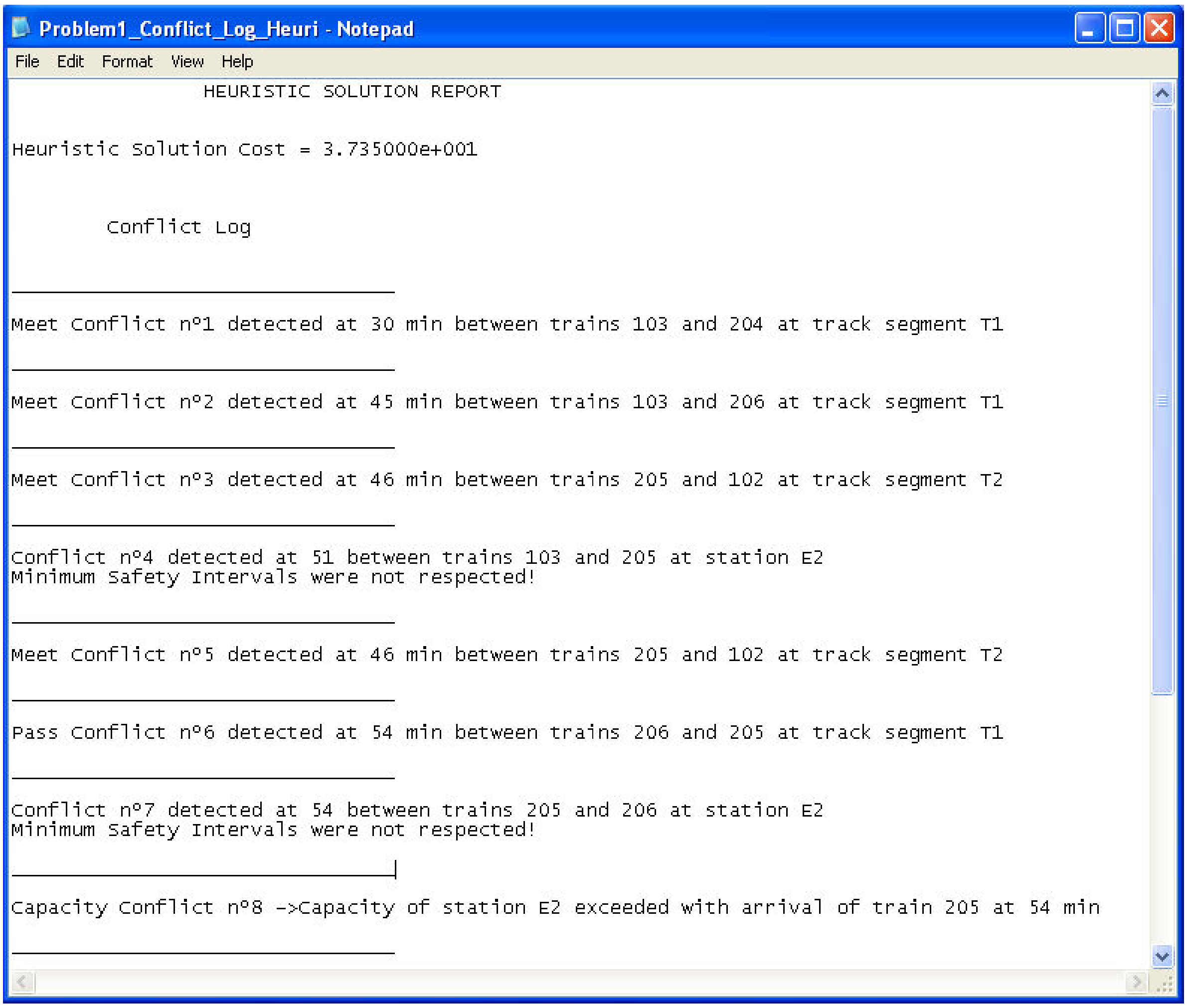
Within this file, there will be two sheets: one named “Heuristic” which contains the final schedule obtained by the heuristic algorithm, and the second sheet named either “Optimal” or “Best”, depending on the situation, that will also contain the respective final schedule.

The format of these timetables is exactly the same as the one described in Section 4.6.1.

4.7.3 Conflict Report

The conflict report is another description of the same solution represented in the final schedule, but in a different format. The conflict report is in the .txt format and the name is also the concatenation of the input file name with the string “_Conflict_Log_Heuri.txt” if it is the report of the heuristic solution. If the solution reported is optimal, then the concatenated string will be “_Conflict_Log_Opti.txt”. Finally if the solution is the best solution found, the added string will be “_Conflict_Log_Best.txt”.

The conflict report first indicates the cost of that solution. Then the conflict log begins with the report of all the conflicts found and solved until the final schedule is achieved. This conflict log is another type of information available for the dispatcher, helping him to trace the conflicts and so, making the reading of the graphical timetable a lot easier. In Figure 4.5 is a small example of a conflict report of a heuristic solution.



Problem1_Conflict_Log_Heuri - Notepad

File Edit Format View Help

```
                HEURISTIC SOLUTION REPORT

Heuristic Solution Cost = 3.735000e+001


        Conflict Log

________________________________

Meet Conflict nº1 detected at 30 min between trains 103 and 204 at track segment T1

________________________________

Meet Conflict nº2 detected at 45 min between trains 103 and 206 at track segment T1

________________________________

Meet Conflict nº3 detected at 46 min between trains 205 and 102 at track segment T2

________________________________

Conflict nº4 detected at 51 between trains 103 and 205 at station E2
Minimum Safety Intervals were not respected!

________________________________

Meet Conflict nº5 detected at 46 min between trains 205 and 102 at track segment T2

________________________________

Pass Conflict nº6 detected at 54 min between trains 206 and 205 at track segment T1

________________________________

Conflict nº7 detected at 54 between trains 205 and 206 at station E2
Minimum Safety Intervals were not respected!

________________________________

Capacity Conflict nº8 ->Capacity of station E2 exceeded with arrival of train 205 at 54 min

________________________________
```

Figure 4.5 Example of a conflict report of a heuristic solution.

4.8 Application Description

This section aims to provide a detailed description of the application's GUI.

The application is composed by a single window, divided into 7 areas, as shown in Figure 4.6 represents the state of the application after the rescheduling process is completed.

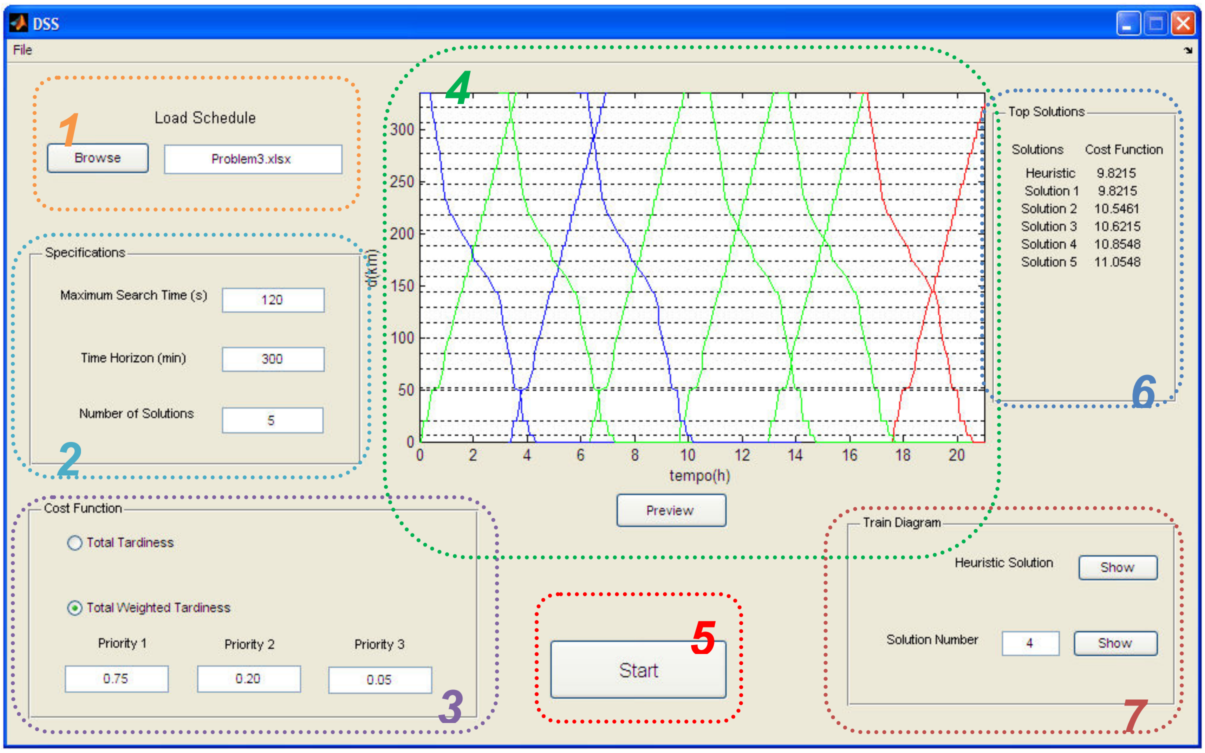


Figure 4.6 Application’s GUI.

The 7 areas will be described next:

1. **Browser** – This button opens a browser so that the user may select the input file.
2. **Specifications** – Panel where the dispatcher specifies the parameters for the search. The maximum Time Search, the Time Horizon and the desired number of final solutions.
3. **Cost Function** – Area reserved for the selection of the intended cost function and the respective weights for each priority.
4. **Preview** – This graphic is just a preview of the initial schedule to verify if the input data is correct.
5. **Start** – As the name indicates, this is where the program begins the re-scheduling process.
6. **Top Solutions** – This chart indicates the ranking of the best solutions found, ordered by their cost. The number of solutions displayed is the number of solutions inserted in the specifications panel.
7. **Train diagram** – Panel where the dispatcher chooses which solution he wants to see in a graphical timetable format. To visualise a solution from the Top Solutions chart, just insert the ranking number in the text box and press show.

Chapter 5

Performance Evaluation

This chapter aims to test the application developed. In each section, a set of tests is made regarding a specific parameter of the program and then the respective result analysis is performed.

The development of any application of this kind is not finished without a serious and meaningful evaluation of its performance. Therefore, this chapter aims to present and analyze the results of the tests carried out to check this model's efficiency. More specifically, the main objective of these tests is to understand how the program is influenced by its input parameters. The parameters in study are:

1. Initial schedule
2. Time horizon
3. Number of Solutions
4. Cost function
5. Maximum search time
6. Upper Bound

Next, each one of these features will then be analyzed individually.

As it was said in Chapter 4, this program was implemented in MATLAB R2007b on the Windows XP operating system, and all the experiments were conducted on a laptop with an Intel Pentium M 1.20 GHz processor and 1.23 GB of RAM.

In order to make these tests more credible, it would have been better to use some real-world timetables. However, such data was not provided and so all the schedules used, were created by the author of this Thesis. There are five schedules, each one with different number of trains and meetpoints in order to evaluate the behavior of the algorithms in different complexity environments. The schedules are presented in detail in Annex 1.

5.1 Initial Schedule

In this first test, it is intended to demonstrate how the increase in the complexity of the input schedule affects the search for a solution. As for the other features, their input was as follows: the time horizon considered for all these experiments was one day (24h); the maximum search time was established at thirty minutes (1800 s); Also the number of requested solutions was one (1); finally, the cost function used was the total weighted tardiness whose weights are the following:

- Priority 1 = 0.75
- Priority 2 = 0.20
- Priority 3 = 0.05

In Table 5.1 are presented the simulation results. In the first column is the identification number of the input schedules which are ordered by their complexity. The number of initial conflicts indicates how disturbed the initial schedule is at the beginning of the simulation. In the last two schedules, 4 and 5, the optimal solution is not found before the maximum search time meaning the solutions presented are not optimal but the best ones found in 1800 s. The limits columns presents values of the number of times that each limit was used to stop the search. The train diagrams from all the solutions in Table 5.1 are presented in Annex 2

Table 5.1 Results for different initial schedules.

Input Schedules	Trains	Meetpoints	Initial Conflicts	CPU Time (s)		Weighted Tardiness		Limits	
				Heuristic Solution	Optimal Solution	Heuristic Solution	Optimal Solution	Time Horizon	Upper Bound
1	6	3	8	2,63	2,58	37,35	23,80	79	429
2	12	6	11	2,52	3,97	101,55	44,30	10	353
3	12	24	22	3,01	944,89	46,54	43,72	62	141694
4	20	24	38	2,73	1800*	52,91	52,14	330	250591
5	40	24	240	6,09	1800*	321,9	382,27	669	205350

* Optimal Solution not found

Through the analysis of Table 5.1, the first and most important conclusion is that the more complex and disturbed the schedule is, the longer it takes for the program to find the optimal solution. Quite surprisingly, the algorithm that finds the heuristic solution is barely affected by this complexity increase. Another good result is the quality of the proposed heuristic solution that revealed a near optimal solution in most cases. In Schedule 5, the quality of the heuristic solution could even overcome the thirty minute search of the optimal algorithm. This aspect will be analyzed later on section 5.6. As for Schedule 2, this is a case where the optimal solution, with half the cost of the heuristic solution, is found in just four seconds. This variety from schedule to schedule reinforces the fact that independently of their complexity, schedules may change a lot from one another making greedy algorithms less reliable. As for the limits used in the search for the optimal solution, the upper bound limit revealed more effective than the time horizon limit. Nonetheless, the time horizon was at its maximum value and for shorter time horizons it is obviously more useful.

5.2 Time Horizon

With this second test, it is expected to show if the variation in the time horizon can accelerate the search for the optimal solution. Once again, it is important to clarify that the indicated solutions are optimal but only within the time horizon limit. They are not the optimal solutions for the whole schedule.

For this second set of simulations, the maximum search time, number of solutions and cost function are the same of the previous tests.

Table 5.2 Variations in the time horizon.

Input Schedules	Time Horizon (h)	CPU Time (s)		Weighted Tardiness	
		Heuristic Solution	Optimal Solution	Heuristic Solution	Optimal Solution
3	2	2,62	2,44	1,754	1,754
	5	2,81	2,74	9,822	9,822
	10	3,72	27,88	25,30	24,61
	24	3,01	944,89	46,54	43,72
4	2	2,90	2,42	1,762	1,762
	5	2,63	2,73	11,734	11,734
	10	3,09	29,67	29,53	26,72
	24	2,73	1800*	52,91	52,14
5	2	3,17	2,9	0,438	0,438
	5	2,93	1800*	70,42	64,31
	10	3,53	1800*	152,64	186,80
	24	6,09	1800*	321,90	382,27

* Optimal Solution not found

By the inspection of Table 5.2, one can conclude that the decrease in the time horizon reduces drastically the search time need to obtain an optimal solution. This result is of course expected since that a short time horizon will ignore the majority of the future conflicts and so reducing the search space. For Schedules 3 and 4, with a time horizon of ten hours, which is more than acceptable, the algorithm is able to solve the conflicts optimally in less than thirty seconds. As for Schedule 5, the densest schedule, the heuristic algorithm completely outperforms the optimal one for long time horizons. Once more, this heuristic values lower than the optimal ones will be explained in Section 5.6.

5.3 Number of Solutions

This test aims to check if the increase in the number of requested solutions has a big influence on the program performance. This hypothesis comes from the fact that, the higher the number of requested solutions is, the higher the upper bound will be, and thus, less restrictive.

Once again the input parameters remain the same except for the time horizon that will be reduced to ten hours.

Table 5.3 Variations in the number of solutions.

Input Schedules	Number of Solutions	CPU Time (s) Optimal Solution
3	1	27,88
	5	30,78
	10	32,79
4	1	29,67
	5	33,39
	10	35,16

The tests reveal that the number of solutions is not relevant in the behavior of the program. Differences like six seconds are not significant when talking about train dispatching. Besides, there is probably more interest in having a set of feasible solutions available so that the dispatcher can have more choice options.

5.4 Cost Function

The purpose of this test is to show how the heuristic solution may have different quality performances with different cost functions. To do so, four different sets of weights were considered:

Table 5.4 Sets of weights for the weighted tardiness function.

	Priority 1	Priority 2	Priority 3
Set 1	0.7	0.2	0.1
Set 2	0.6	0.3	0.1
Set 3	0.5	0.4	0.1
Set 4	0.5	0.3	0.2

In order to be more realistic, there were two conditions that the above sets had to verify:

1) $Priority_1 > Priority_2 > Priority_3$;

2) $Priority_1 + Priority_2 + Priority_3 = 1$;

The case where all the weights are the same is also considered with the total tardiness cost function. The time horizon is set to 10 h and the maximum search time was set to 300 s. Bellow in Table 5.6 are the results of this experiment.

Table 5.5 Variations in the cost function.

Input Schedules	Cost Function	Sets of Weights	Solution Cost	
			Optimal	Heuristic
4	Weighted Tardiness	Set 1	23,97	24,66
		Set 2	29,97	31,27
		Set 3	31,76	37,88
		Set 4	27,23	30,00
	Total tardiness	_	69,47	94,52
5	Weighted Tardiness	Set 1	182,07	169,92
		Set 2	170,94	174,50
		Set 3	158,93	179,14
		Set 4	161,52	209,23
	Total tardiness	_	433,75	771,02

Analyzing the results, it may be concluded, as expected, that the gap between the heuristic solution and the proposed optimal solution increases when the weights are more evenly distributed. The heuristic algorithms performs as if all the weight is in the train with higher priority and hence these results. It is now up to the dispatcher to analyze the different produced solutions and work with different cost functions in order to decide which set of weights fits better to reality.

5.5 Maximum Search Time

The interest in studying the maximum search time is mainly to observe how fast does the algorithm converge into an optimal solution. The input parameters are the same as in the first test.

To better illustrate how this algorithm behaves in time, the graphic in Figure 5.1 was made. The solutions quality, or cost, is not presented in its absolute value but instead it is normalized to make their comparison easier. Considering R_{opt} the quality of the optimal solution and R_i the quality of the current solution, then the normalized quality R is defined as being R_i/R_{opt} . In Schedules 4 and 5, the optimal solution considered was the one obtained at 1800 s.

In Table 5.6 are indicated the absolute cost values obtained in this experiment.

Table 5.6 Variations in the maximum search time.

Input Schedules	Maximum Search Time (s)					
	15	30	60	300	600	1800
3	49,73	49,22	48,12	46,29	46,29	43,72*
4	59,79	56,22	55,72	54,12	54,12	52,14
5	386,22	386,12	385,84	385,62	382,27	382,27

*Optimal Solution found in 944s

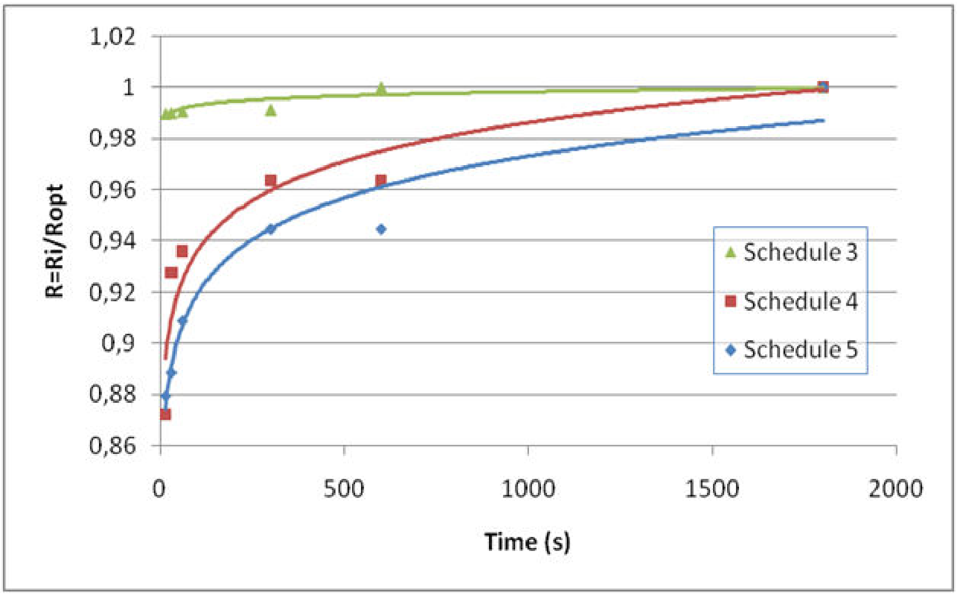


Figure 5.1 Optimal algorithm performance.

As it was concluded in the first test, the more complex the schedule is, the longer it takes for the program to obtain the optimal solution. This conclusion is once more evidenced in this graphic where the lines are logarithmic tendency lines. These tendency lines are slightly overestimated because in the normalization, the best solutions found at 1800 seconds, were considered as being the optimal ones.

5.6 Upper Bound

With this set of tests, it is intended to study the effect of the initial upper bound value, in the performance of the optimal algorithm. By default, the optimal algorithm does not take into consideration the cost value of the heuristic solution. The initial value of the upper bound is set to infinite. This decision was made so that the program could also consider solutions worse than the heuristic solution and let the dispatcher analyze them in the end. The set back of this option is that, by giving a high initial upper bound, the algorithm's performance is affected and so, it might not be as effective as it could.

Again, the input parameters are the same as in the first test.

Table 5.7 Variations in the upper bound initial value.

Input Schedules	**Initial Upper Bound**	**Maximum Search Time (s)**					
		15	**30**	**60**	**300**	**600**	**1800**
3	Infinite	49,73	49,22	48,12	46,29	46,29	43,72 (944s)
	Heuristic Solution = 46,54	46,54	46,54	46,54	46,54	46,54	43,72 (903s)
4	Infinite	59,79	56,22	55,72	54,12	54,12	52,14
	Heuristic Solution = 52,91	52,91	52,91	52,91	52,91	52,91	51,95

To better illustrate the difference between the two situations, the graphics below were created. Like in the previous section, these values are normalized to the optimal solution or in the case of schedule 4 to the best solution found in 1800s.

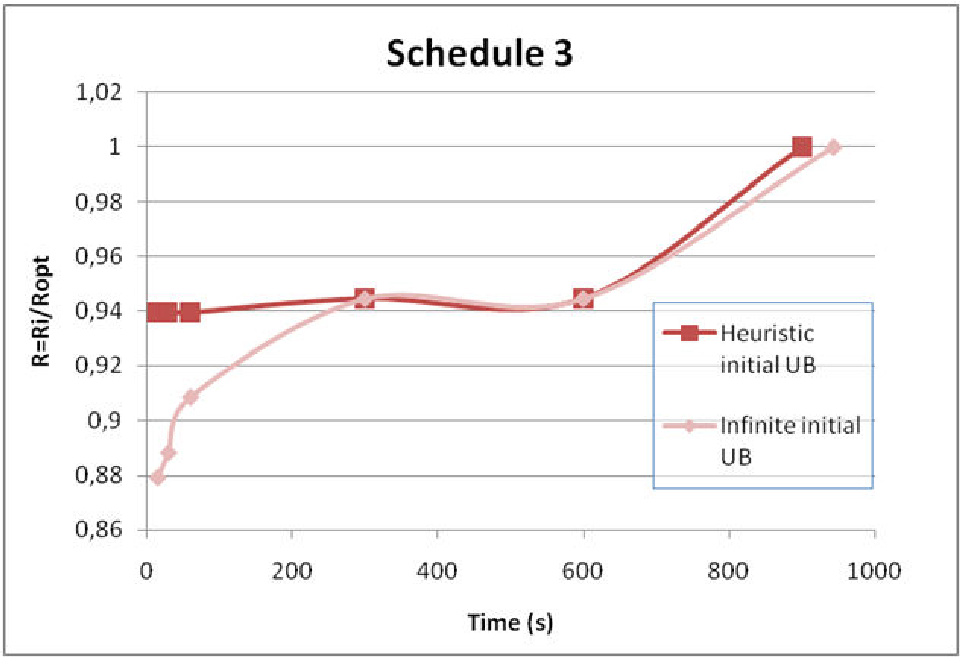


Figure 5.2 Optimal algorithm performance for schedule 3 with enhanced upped bound.

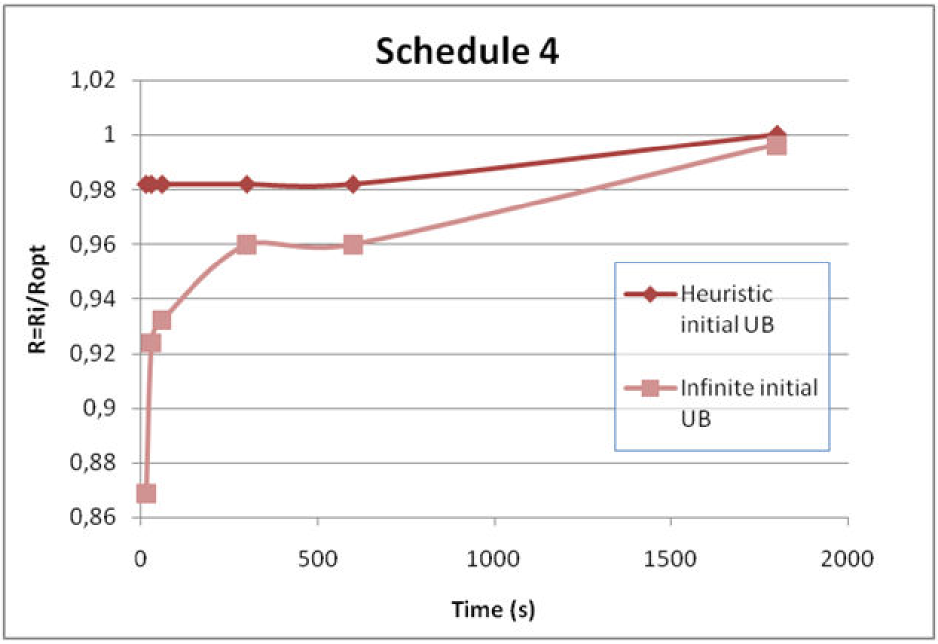


Figure 5.3 Optimal algorithm performance for schedule 4 with enhanced upper bound.

For these two schedules, the improvement of the upper bound was not very obvious. In schedule 3, the algorithm took about 40 seconds less to obtain the optimal solution in a 15 min time search which is not very relevant. As for Schedule 4, after the 1800 s time limit, the solution produced by the enhanced upper bound was slightly better than the other.

Maybe, for longer time searches and with some more experiments, the results could have been more conclusive.

Chapter 6

Conclusions and Future Work

This chapter finalizes this report by presenting to the reader a brief summary of the solutions developed as well as some conclusions followed by prospects for future work.

6.1 Summary and Conclusions

Chapter 1, introduced the problem addressed in this Thesis, highlighting that the growing interest in railway lines, and the need to make them more profitable, justifies the development of a system that optimizes train control operations. Chapter 2 structured the problem at hand, by dividing the models into two main types: fixed speed models, and variable speed models. As it was intended for the developed program to be as real as possible, a variable speed model was chosen.

Chapter 3 first presented the architecture of the developed solution and its relation with the dispatcher. After the models assumptions and inputs are defined, a mathematical formulation of this optimization problem was formally introduced. In order to validate the model created for conflict detection, two theorems and the consequent corollary were defined and demonstrated. It was therefore concluded, that this conflict detection method is valid, and reduces the complexity of the detection algorithm.

Still in Chapter 3, all types of conflict resolutions were explained in detail. In particular, the capacity conflicts revealed the most complex ones to solve, but with the advantage of giving more realism to the model.

Finally, two resolution methods for re-scheduling were proposed. One was the heuristic solution, which is a greedy algorithm that provides a solution for the problem in a short amount of time by means of a local decision criterion. The second method was the optimal solution, an algorithm that checks for all the possibilities, by using a search tree and a branch-and-bound technique. In order to make the optimal search less exhaustive and more useful in real-time, the time-horizon and the maximum search time were adopted. With a proper management of this search parameters, the dispatcher may solve a conflict in a short amount of time, pushing the conflicts further on, and gaining precious time for another search, looking for better results.

Chapter 4 described in detail the algorithms of the main modules of the program: the conflict detection, the heuristic solution and the optimal solution. In order to offer all the information needed by the user, to make a proper use of the application, the input and output data was shown as well as the applications' GUI.

In Chapter 5, tests were executed to evaluate the performance of the program and then the results were analyzed. The heuristic algorithm performed fast in all situations, independently of the complexity of the initial schedule. The quality of its solutions was mainly near optimal but, as expected for a greedy algorithm, there were cases were the proposed solution was far from optimal.

The optimal algorithm depends very much on the complexity of the initial schedule. Nonetheless, the time horizon parameter revealed very effective, with the tradeoff of shortening the validity of the solution. The number of requested solutions has almost no effect on the performance and provides the dispatcher with a set of useful different solutions.

In summary, it can be said that the tests revealed good results and that with the correct choice of parameters, this application represents a powerful tool to help the dispatcher.

Despite the positive results, some improvements can be made in the developed application. These improvements were left for future work and are discussed in the next section.

6.2 Future Work

Despite the encouraging results obtained, the solution developed still leaves room for improvements:

Introduction of a good lower bound estimator – This improvement may accelerate the optimal search and so, enabling the program to achieve better results in a shorter amount of time

Length restriction – To make this model even more realistic, there could be additional restrictions related with the train length. For example, in trains longer than the meetpoints' length, there can be no meet or pass.

More tests – So as to give more credibility to this model, it would be necessary to run tests with real cases and compare them with the dispatcher's decisions.

Cost functions – There are many cost functions that may suit better to reality, depending on the situation. Therefore, some more cost functions should be introduced, to give the dispatcher a wider range of options.

Heuristic Algorithm – The heuristic solution has also a lot of room for improvements. It could weight its decisions based on the selected cost function to improve its quality.

In conclusion, there is still a lot of research to do in the field of re-scheduling, and the objective of finding optimal solutions in feasible time will continue to be a difficult task.

Annex 1

Input Files

This appendix includes some of the input files used in the tests

Schedule 1

Schedule Sheet:

	Meetpoint1		Meetpoint2		Meetpoint3	
	ARRIVALS	DEPARTURES	ARRIVALS	DEPARTURES	ARRIVALS	DEPARTURES
101	0	10	15	15	22	1000
102	0	15	30	50	65	1000
103	0	30	36	42	48	1000
204	42	1000	36	37	0	32
205	65	1000	52	58	0	46
206	70	1000	10	50	0	5

Trains sheet:

Number of Inbound Trains	3
Number of Outbound Trains	3

TRAINS DESCRIPTION		
ID	TYPE	LENGTH
101	1	1000
102	2	520
103	1	868
204	1	458
205	1	1000
206	3	940

Railway Sheet:

STATIONS DESCRIPTION			
NAME	CAP	LENGTH	DISTANCE
E1	12	1000	0
E2	3	1000	6
E3	8	1000	20

SEGMENT DESCRIPTION
NAME
T1
T2

Schedule 2

Schedule Sheet:

E1		E2		E3		E4		E5		E6	
ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART
0	5	25	30	61	65	80	82	105	115	170	180
240	245	265	270	301	305	320	322	345	355	410	420
410	415	435	440	471	475	490	492	515	525	580	590
640	645	665	670	701	705	720	722	745	755	810	820
810	815	835	840	871	875	890	892	915	925	980	990
960	965	985	990	1021	1025	1040	1042	1065	1075	1130	1140
160	500	140	147	105	110	90	90	61	65	8	8
320	660	300	307	265	270	250	250	221	225	168	168
435	775	415	430	385	390	365	370	336	340	283	283
575	915	555	570	525	535	505	510	476	480	423	423
665	1005	645	660	615	625	595	600	566	570	513	513
785	1125	765	780	740	740	715	720	686	690	633	633

Trains sheet:

Number of Inbound Trains	6
Number of Outbound Trains	6

TRAINS DESCRIPTION		
ID	TYPE	LENGTH
101	3	1000
102	2	520
103	2	868
104	1	541
105	1	1002
106	3	1546
201	2	124
202	1	652
203	2	658
204	1	458
205	1	1000
206	3	940

Railway Sheet:

STATIONS DESCRIPTION			
NAME	CAP	LENGTH	DISTANCE
E1	12	1000	0
E2	3	1000	25
E3	8	1000	85
E4	7	1000	134
E5	4	1000	160
E6	12	1000	230

SEGMENT DESCRIPTION
NAME
T1
T2
T3
T4
T5

Schedule 3

Schedule Sheet:

	E1		E2		E3		E4		E5		E6		E7		E8		E9		E10		E11		E12	
	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART
101	0	5	8	8	10	18	28	30	40	43	55	55	58	58	65	68	75	75	83	83	92	97	102	107
102	200	205	208	208	210	218	228	230	240	243	255	255	258	258	265	268	275	275	283	283	292	297	302	307
103	375	380	383	383	385	393	403	405	415	418	430	430	433	433	440	443	450	450	458	458	467	472	477	482
104	575	580	583	583	585	593	603	605	615	618	630	630	633	633	640	643	650	650	658	658	667	672	677	682
105	775	780	783	783	785	793	803	805	815	818	830	830	833	833	840	843	850	850	858	858	867	872	877	882
106	1050	1055	1058	1058	1060	1068	1078	1080	1090	1093	1105	1105	1108	1108	1115	1118	1125	1125	1133	1133	1142	1147	1152	1157
201	260	500	248	255	230	243	220	225	213	213	208	208	200	203	193	193	185	185	183	183	170	175	153	153
202	435	675	423	430	405	418	395	400	388	388	383	383	375	378	368	368	360	360	358	358	345	350	328	328
203	610	850	598	605	580	593	570	575	563	563	558	558	550	553	543	543	535	535	533	533	520	525	503	503
204	885	1125	873	880	855	868	845	850	838	838	833	833	825	828	818	818	810	810	808	808	795	800	778	778
205	1060	1300	1048	1055	1030	1043	1020	1025	1013	1013	1008	1008	1000	1003	993	993	985	985	983	983	970	975	953	953
206	1235	1475	1223	1230	1205	1218	1195	1200	1188	1188	1183	1183	1175	1178	1168	1168	1160	1160	1158	1158	1145	1150	1128	1128

E13		E14		E15		E16		E17		E18		E19		E20		E21		E22		E23		E24	
ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART	ARRIV	DEPART
111	116	121	125	130	135	140	144	149	154	159	163	168	173	177	182	187	192	196	201	206	210	215	220
311	316	321	325	330	335	340	344	349	354	359	363	368	373	377	382	387	392	396	401	406	410	415	420
486	491	496	500	505	510	515	519	524	529	534	538	543	548	552	557	562	567	571	576	581	585	590	595
686	691	696	700	705	710	715	719	724	729	734	738	743	748	752	757	762	767	771	776	781	785	790	795
886	891	896	900	905	910	915	919	924	929	934	938	943	948	952	957	962	967	971	976	981	985	990	995
1161	1166	1171	1175	1180	1185	1190	1194	1199	1204	1209	1213	1218	1223	1227	1232	1237	1242	1246	1251	1256	1260	1265	1270
123	125	113	115	90	90	70	72	58	60	54	54	48	50	45	45	38	38	33	35	28	28	0	25
298	300	288	290	265	265	245	247	233	235	229	229	223	225	220	220	213	213	208	210	203	203	175	200
473	475	463	465	440	440	420	422	408	410	404	404	398	400	395	395	388	388	383	385	378	378	350	375
748	750	738	740	715	715	695	697	683	685	679	679	673	675	670	670	663	663	658	660	653	653	625	650
923	925	913	915	890	890	870	872	858	860	854	854	848	850	845	845	838	838	833	835	828	828	800	825
1098	1100	1088	1090	1065	1065	1045	1047	1033	1035	1029	1029	1023	1025	1020	1020	1013	1013	1008	1010	1003	1003	975	1000

Trains sheet:

Number of Inbound Trains	6
Number of Outbound Trains	6

TRAINS DESCRIPTION		
ID	**TYPE**	**LENGTH**
101	1	1000
102	2	520
103	1	868
104	1	541
105	1	1002
106	3	1546
201	2	124
202	1	652
203	2	658
204	1	458
205	1	1000
206	3	940

Railway Sheet:

STATIONS DESCRIPTION			
NAME	**CAP**	**LENGTH**	**DISTANCE**
E1	12	1000	0
E2	3	1000	6
E3	8	1000	20
E4	7	1000	50
E5	4	1000	52
E6	4	1000	70
E7	7	1000	85
E8	4	1000	100
E9	8	1000	114
E10	7	1000	129
E11	5	1000	144
E12	4	1000	159
E13	8	1000	174
E14	7	1000	188
E15	3	1000	203
E16	5	1000	218
E17	4	1000	233
E18	8	1000	248
E19	5	1000	262
E20	8	1000	277
E21	8	1000	292
E22	4	1000	307
E23	6	1000	322
E24	15	1000	336

SEGMENT DESCRIPTION
NAME
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T12
T13
T14
T15
T16
T17
T18
T19
T20
T21
T22
T23

Annex 2

Test Results

In this annex are the train diagrams of the tests from Section 5.1.

Schedule 1

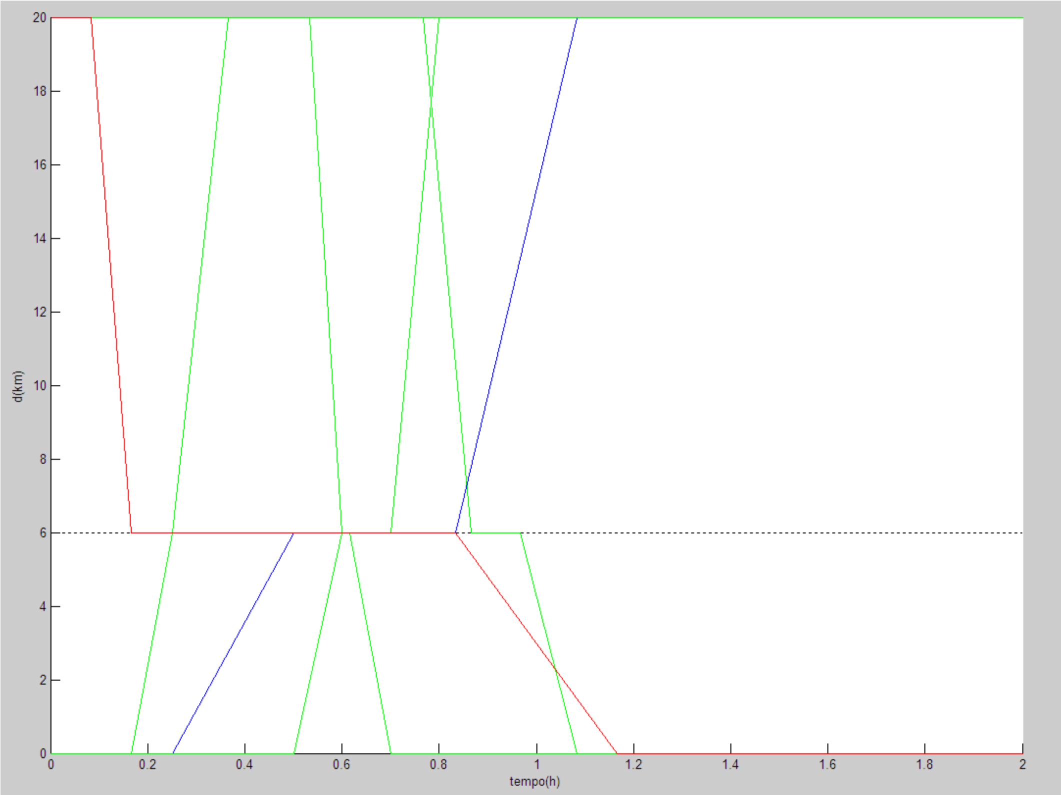


Figure A2.1 Problem 1 Initial Schedule.

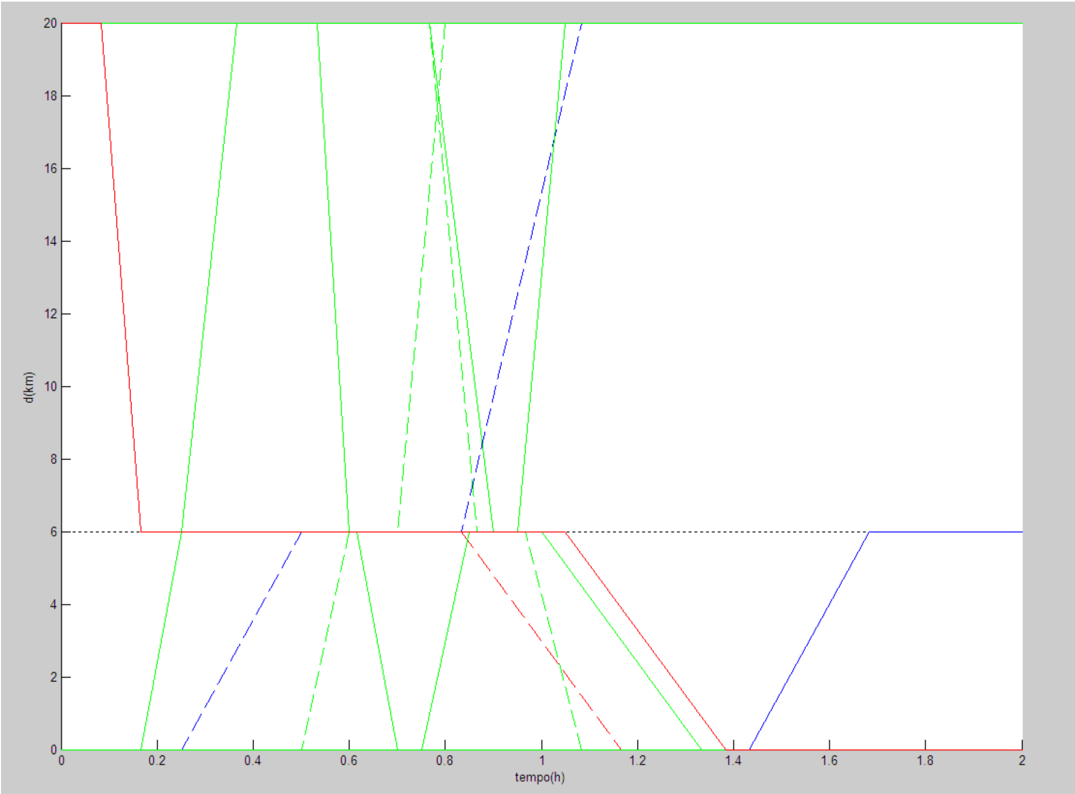


Figure A2.2 Problem 1 Heuristic Solution.

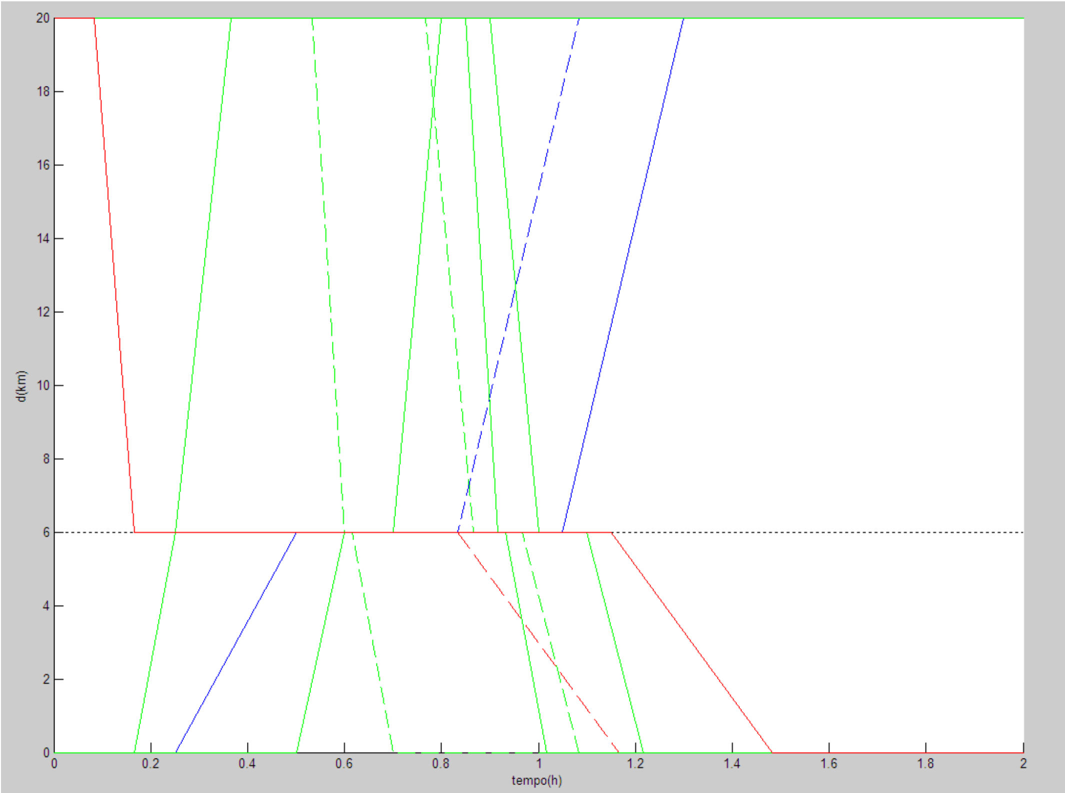


Figure A2.3 Problem 1 Optimal Solution.

Schedule 2

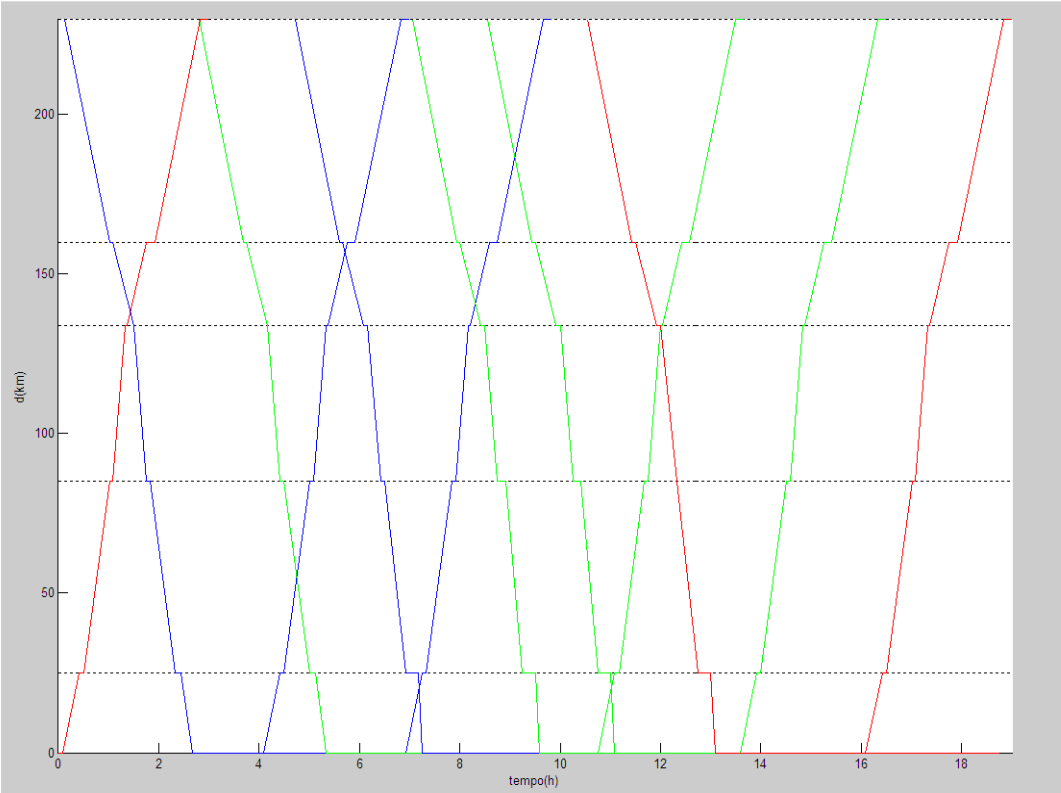


Figure A2.4 Problem 2 Initial Schedule.

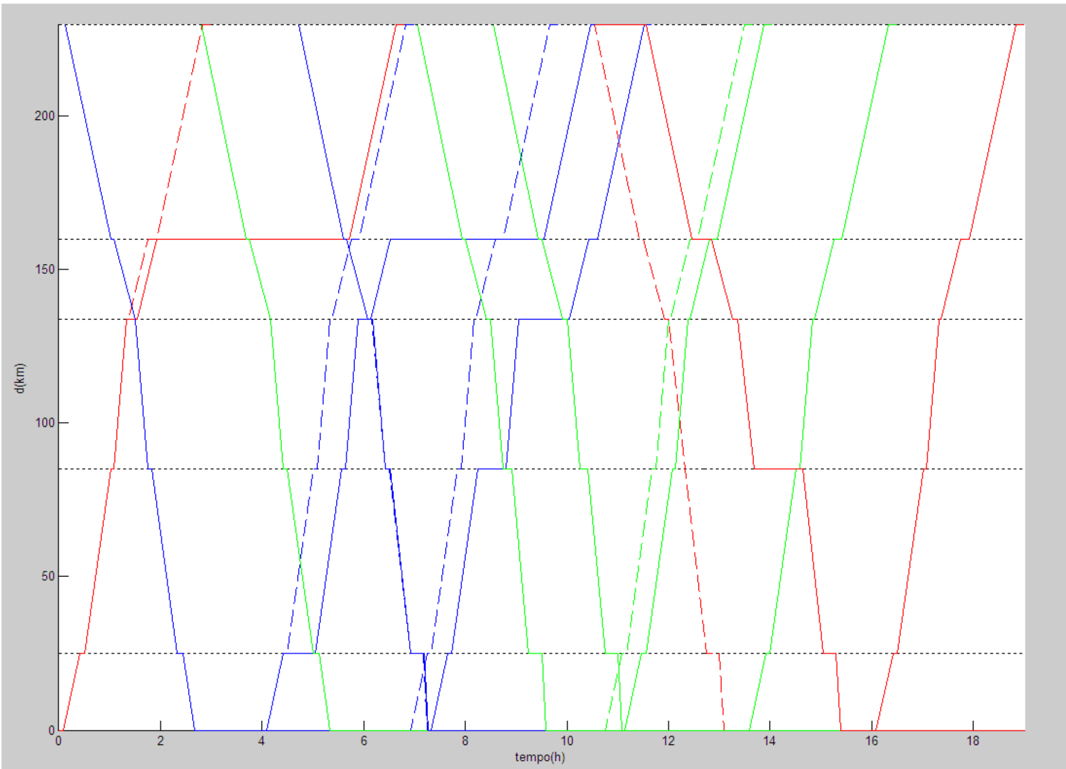


Figure A2.5 Problem 2 Heuristic Solution.

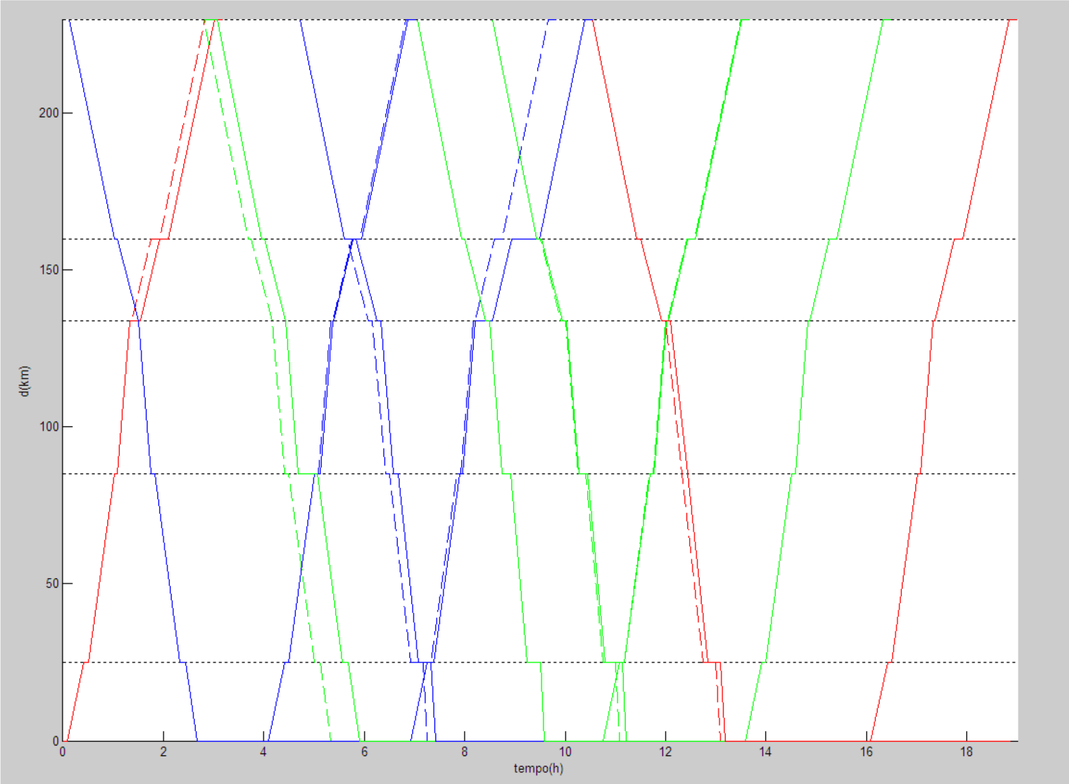


Figure A2.6 Problem 2 Optimal Solution.

Schedule 3

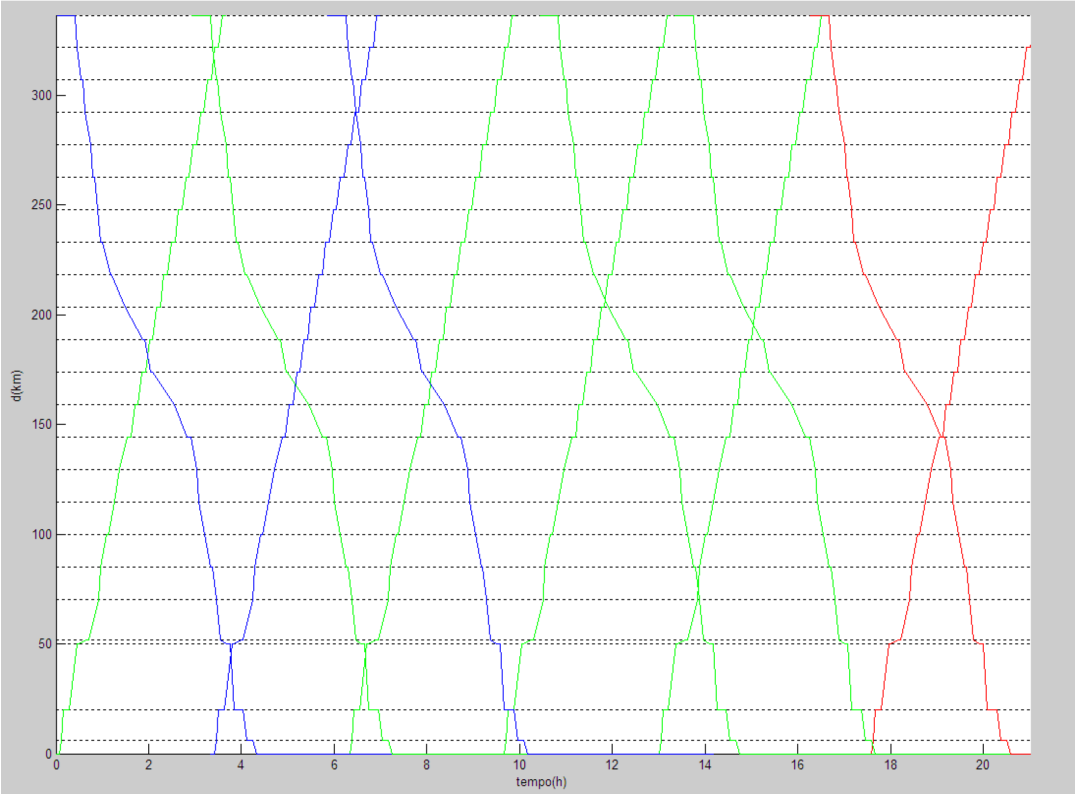


Figure A2.7 Problem 3 Initial Schedule.

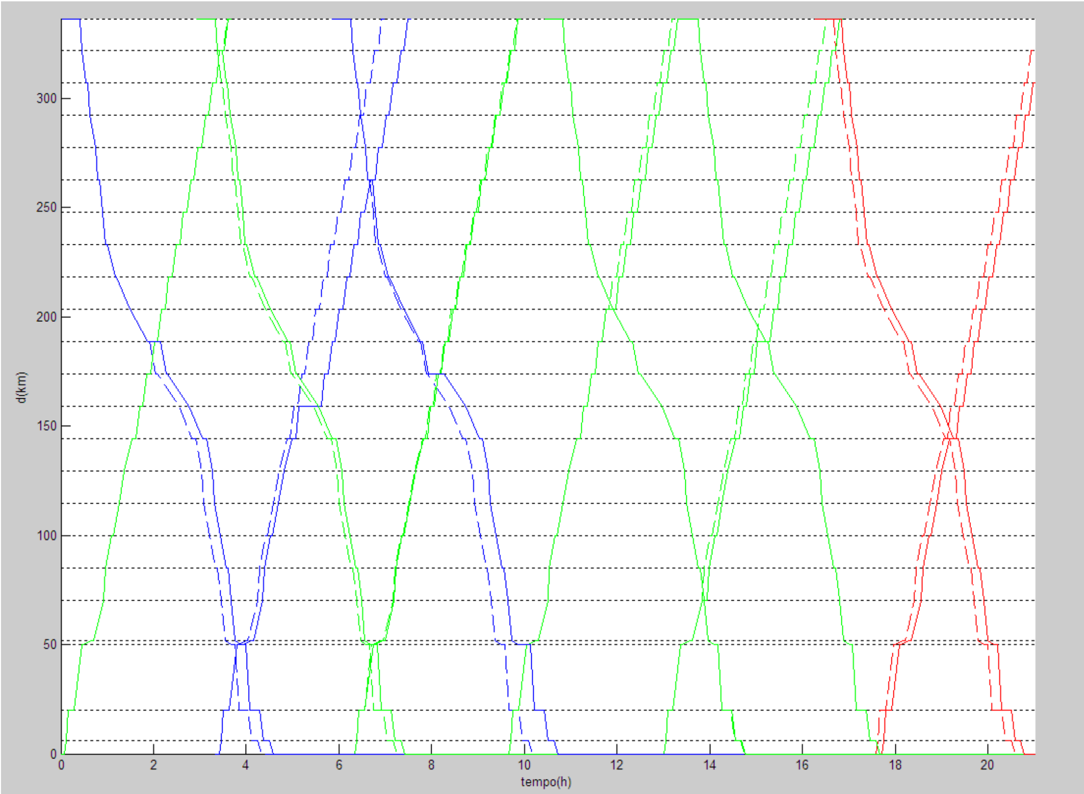


Figure A2.8 Problem 3 Heuristic Solution.

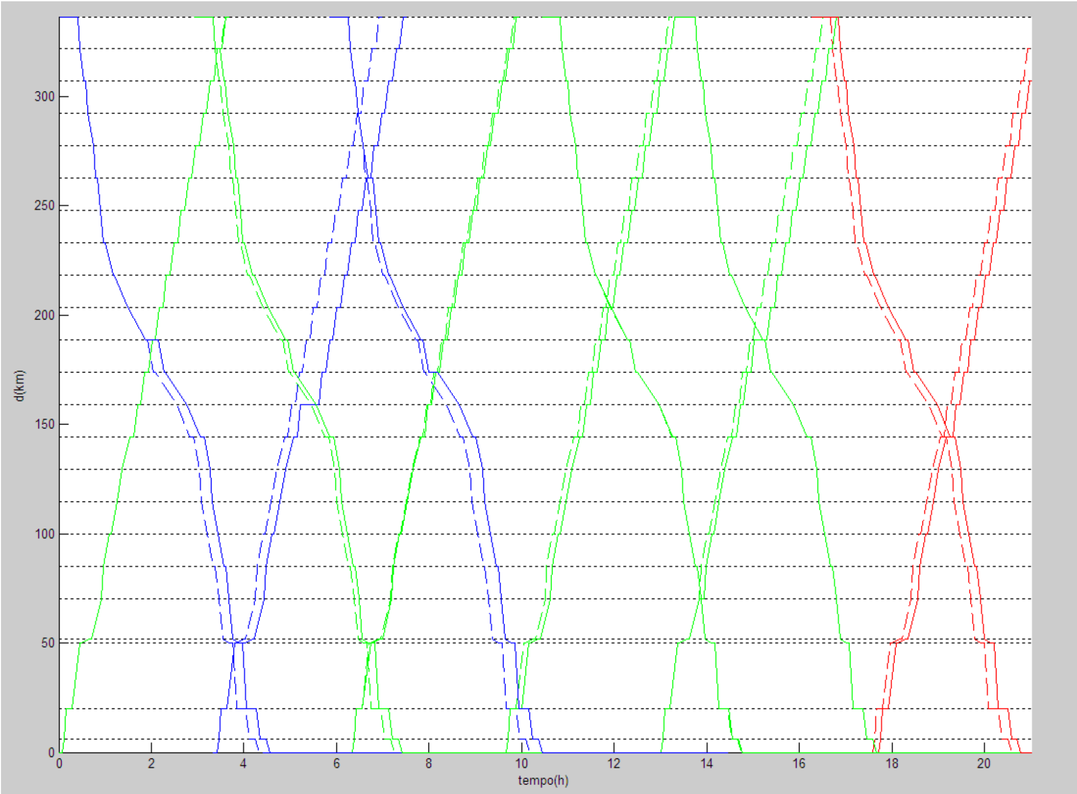


Figure A2.9 Problem 3 Optimal Solution.

Schedule 4

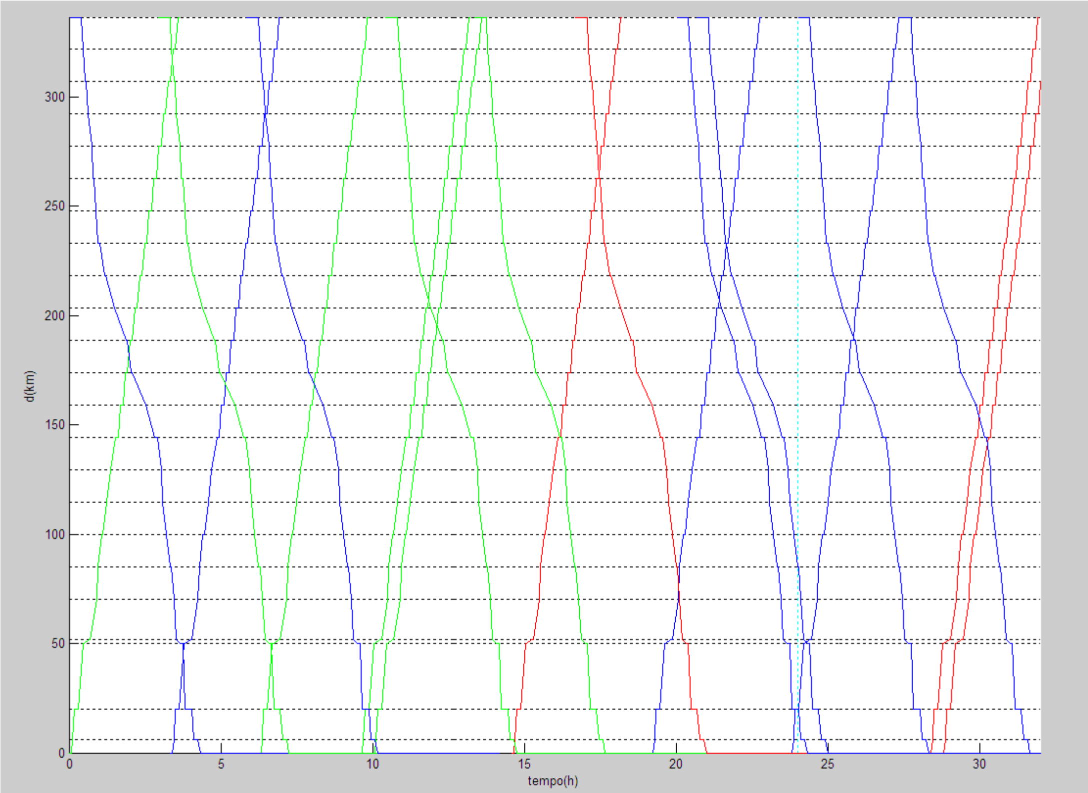


Figure A2.10 Problem 4 Initial Schedule.

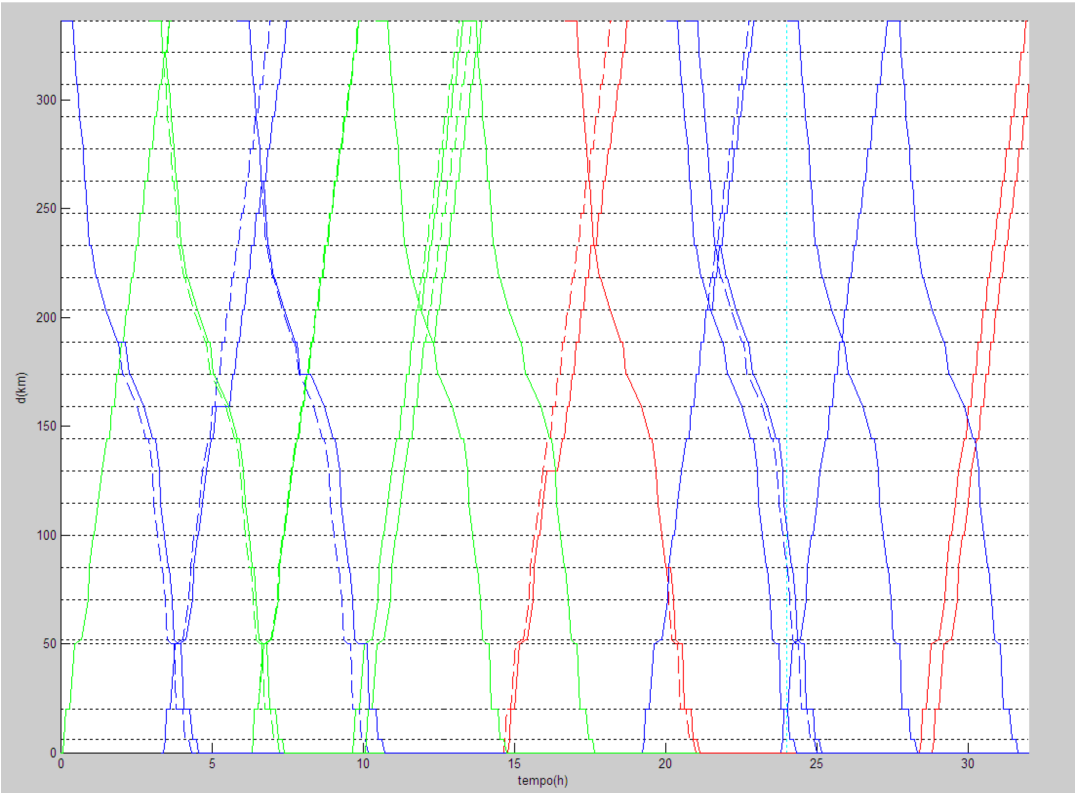


Figure A2.11 Problem 4 Heuristic Solution.

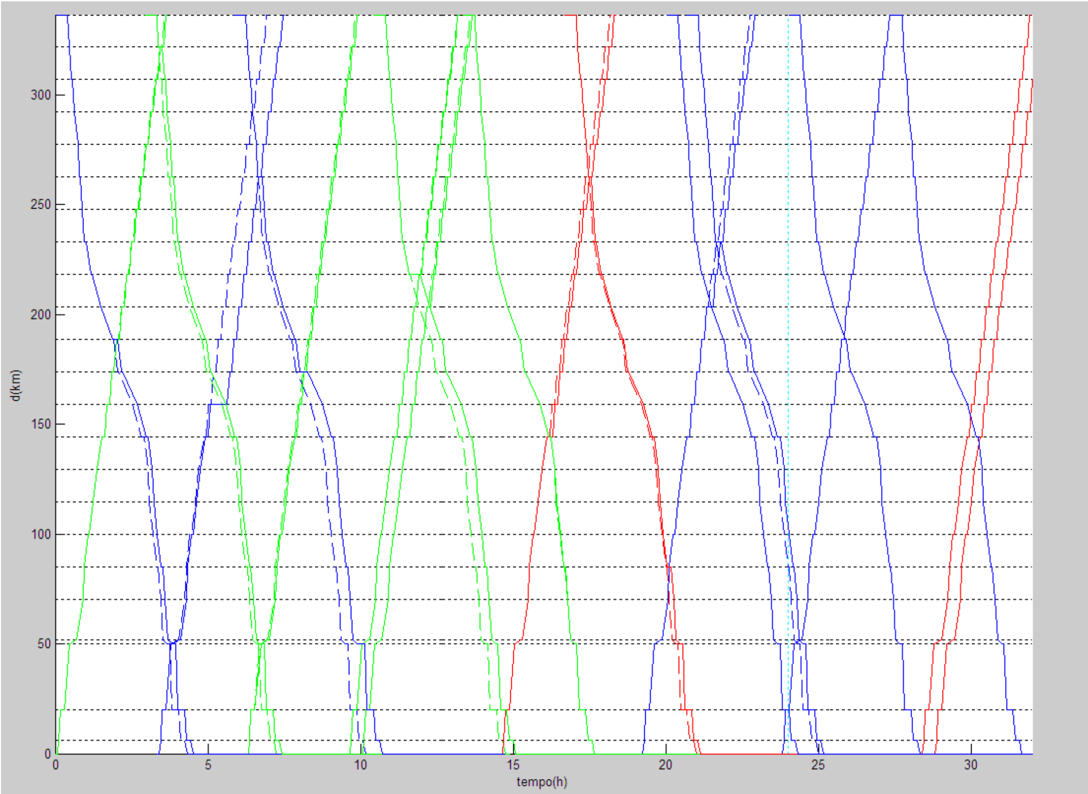


Figure A2.12 Problem 4 Best Solution Found.

Schedule 5

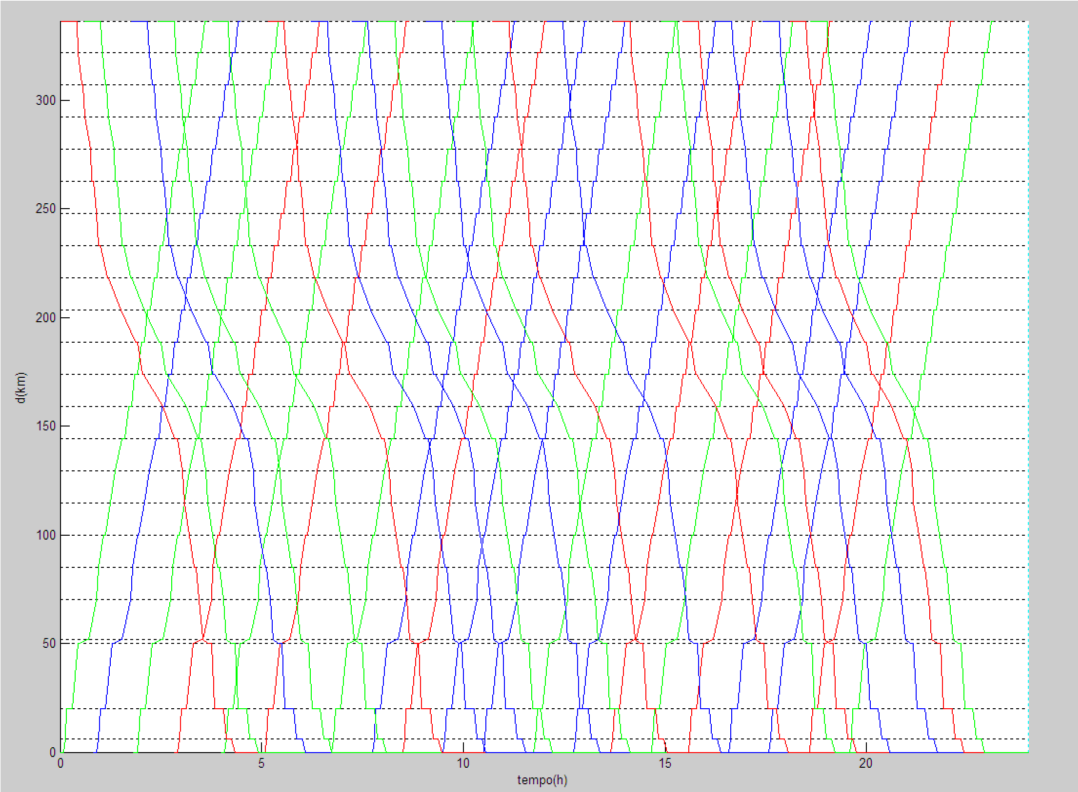


Figure A2.13 Problem 5 Initial Schedule.

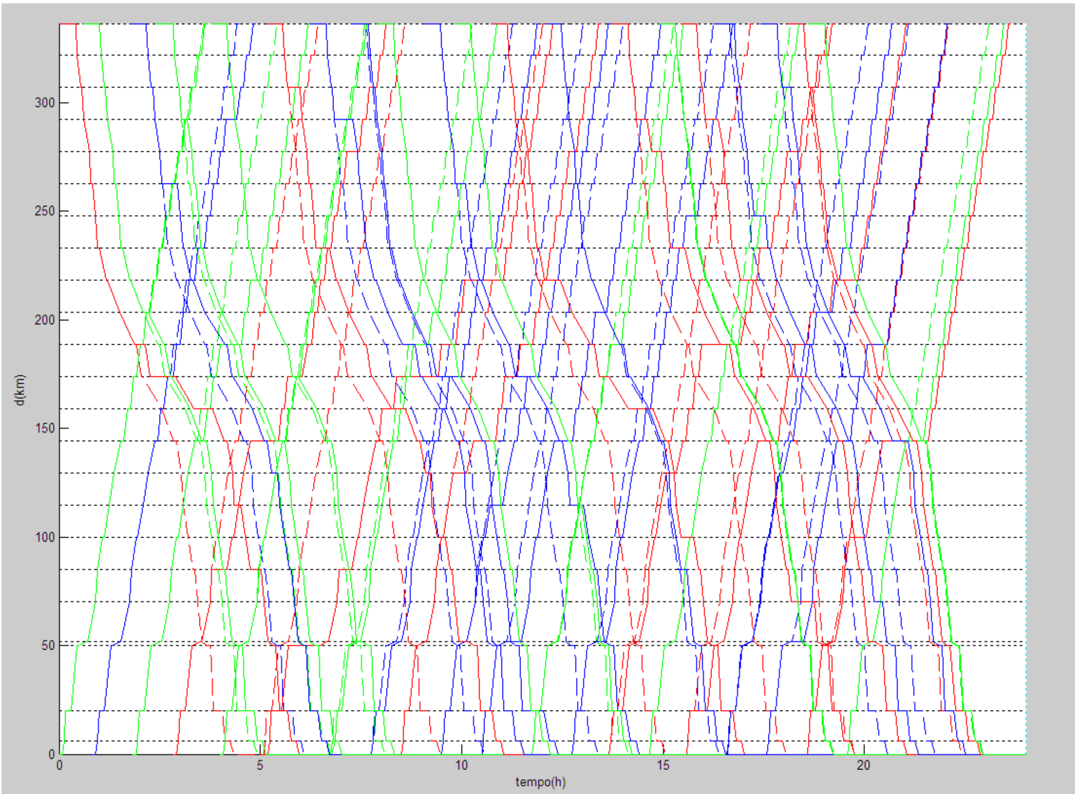


Figure A2.14 Problem 5 Heuristic Solution.

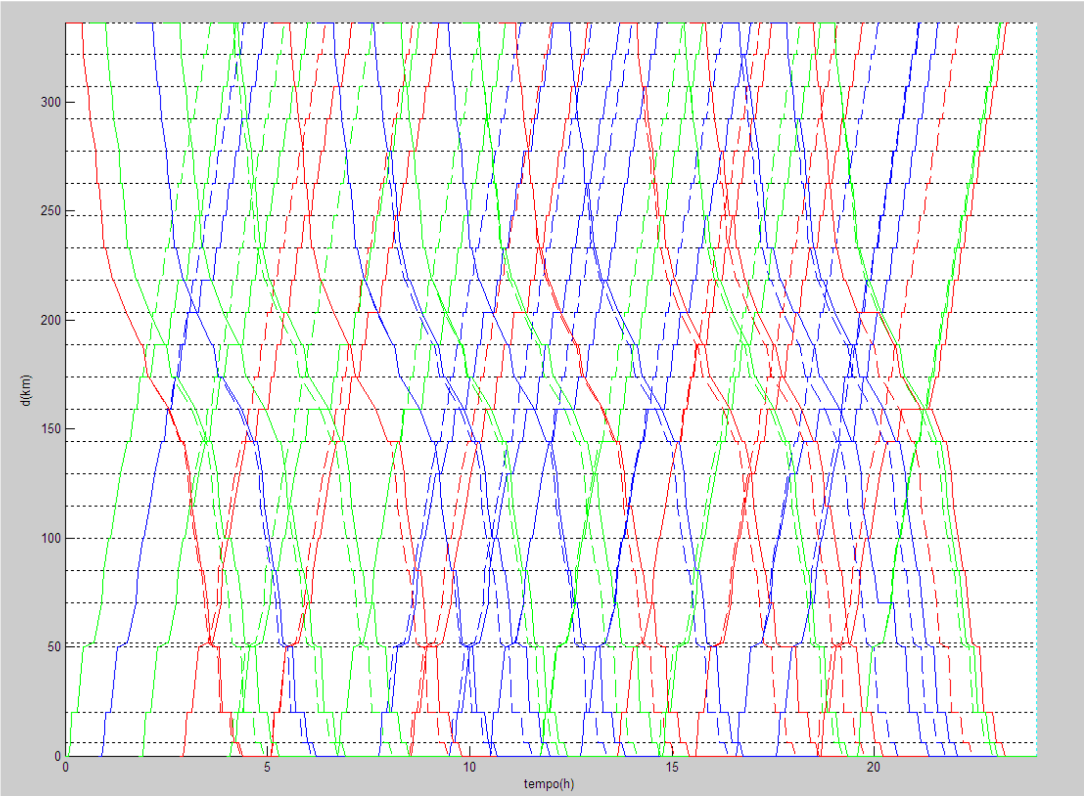


Figure A2.15 Problem 5 Best Solution Found.

References

[Adenso-Díaz99] Adenso-Díaz B., González M.O., González-Torre P., "On-line timetable re-scheduling in regional train services", Transp. Res.- Part B, vol. 33, no.6, pp. 387-398, Aug.1999.

[Cordeau98] Cordeau J.F., Toth P., Vigo D., "A survey of optimization models for train routing and scheduling", Transp. Sci., vol.32, no. 4, pp. 380-404, 1998.

[D'Ariano07a] D'Ariano A., Pranzo M., Hansen I.A., "Conflict resolution and train speed coordination for solving real-time timetable perturbations", IEEE Transaction on intelligent transportation systems, vol.8, NO.2, june 2007.

[D'Ariano07b] Andrea D'Ariano, Dario Pacciarelli, Marco Pranzo, "A branch and cound algorithm for scheduling trains in a railway network", European Journal of Operational Research 183 (2007) 643-657.

[Hellström99] Hellström P., Sandblad B., Frej I., Anders Gideon, "An evaluation of algorithms and systems for Computer-Aided Train Dispatching", School of Engineering, S-78188 Borlänge, Sweden.

[Higgins95] Higgins A.,, Ferreira. L., Kozan E., "Modelling single line train operations", Transportation Research Record 1498, Journal of the Transportation Research Board, Railroad Transportation Research, pp 9-16.

[Higgins96] Higgins A.,, Ferreira. L., Kozan E., "Optimal scheduling of trains on a single line track", Transportation Research – Part B, 30B (2), 147-161.

[Mascis02a] Alessandre Mascis, Dario Pacciarelli, Marco Pranzo, "Models and Algorithms for Traffic Management of Rail Networks", RT-DIA-74-2002.

[Mascis02b] Alessandro Mascis, Dario Pacciarelli, "Job-shop scheduling with blocking and no-wait constraints", European Journal of Operational Research 143 (2002) 498-517.

[Medanic02] Medanic J., Dorfman M.J., "Efficient Scheduling of Traffic on a Railway Line", Journal of Optimization Theory and Applications: Vol.115, No. 3, pp. 587-602, December 2002.

[Sahin99] Sahin I., "Railway traffic control and train scheduling based on inter-train conflict management", Transp. Res.- Part B, vol. 33, no.7, pp. 511-534, Sep. 1999.

[Zhou2006] Zhou X., Zhong M."Single-track train timetabling with guaranteed optimality", Transportation Research – Part B, 41 (2007), 320-341.