Performance Optimization of Train Dispatching Support System

(Case of Ethiopia)

By:

Nurye Hassen

A Thesis Submitted to Addis Ababa University

Addis Ababa Institute of technology

In Partial Fulfillment of the requirements for the Degree of

Master of Science in

Electrical Engineering for Railway Systems

Advisor: Abebe Teklu (PhD candidate)

Addis Ababa, Ethiopia

March, 2018

Submitted By:

Nurye Hassen	_____	_____
Student	Signature	Date

Approved By:

1.	Mr. Abebe Teklu (PhD Candidate)	_____	_____
	Advisor	Signature	Date
2.	Dr. Celestin Nkundineza	_____	_____
	Internal examiner	Signature	Date
3.	Dr. Yihenew Wondie	_____	_____
	External examiner	Signature	Date
4.	_____	_____	_____
	Chair or School Dean	Signature	Date
5.	_____	_____	_____
	Director of Postgraduate Program	Signature	Date

Declaration

I, the undersigned declare that, this thesis is my original work, has not been presented for a degree in this or other universities. All sources of materials used for this thesis have been fully acknowledged.

Name: Nurye Hassen Signature: _____

Place: Addis Ababa Institute of Technology, Addis Ababa University, Addis Ababa

Date of Submission: _____

This thesis has been submitted for examination with my approval.

Abebe Teklu (PhD Candidate). _____

Thesis Advisor Name Signature

ABSTRACT

One of the significant challenges in the daily operation of train dispatching is making the right decision upon unplanned conflict occurrence. The resolution process by itself will introduce an additional delay on the railway network unless well resolved. Thus, it is a decisive and challenging issue for train dispatchers and railway operation planners to decide which of the trains to stop or to pass from the trains involved in the conflict to bring minimum propagated delay. Such an operation with effective conflict resolution requires an intelligent decision support system that considers minimization of future dwelling time.

To this end, this thesis developed a decision support system that provides an intelligent decision to the train dispatcher by detecting a conflict on a rail network with an optimal resolution of the conflict and cost. The approach addressed the minimization of overall delay due to the conflict resolution in addition to detection and resolution. A mixed integer linear programming approach has been implemented to find optimal combinations of arrival and departure events that bring minimum propagated delay. Optimization toolbox of the commercial software, MATLAB R2015a, was used to develop the solving algorithm and obtain the result. The solution procedure is also clearly illustrated using practical and hypothetical applications. The model has been applied to Ethio-Djibouti Railway enterprise railroad from DIRE-DAOUA to DAOUENLE, which consists of eight stations. The program was able to reach an optimal solution with minimum cost when compared to the manual (heuristic) approach especially for an increased number of trains and stations. The model was also tested based on various hypothetical assumptions and showed that it is a powerful tool to be used for train dispatchers for ensuring operational optimality and safety of the railway line.

Keywords

Conflict Detection and Resolution, Decision Support System, Mixed integer linear programming, Re-scheduling, Train Dispatching.

ACKNOWLEDGMENT

First and foremost, I want to praise the Almighty Allah, for giving me the strength to finish this thesis and next I would like to express my gratitude to several people that helped me along the way and made this thesis possible.

I thank my parents for their continued moral support during my study. I would also like to thank the German academic exchange service, DAAD for financially funding my graduate study and all DAAD Regional members of staff in Nairobi.

Furthermore, I would like to thank my advisor, Abebe Teklu, for his persistence and wisdom. He made the ultimate impact on the research presented in this thesis and has been an excellent advisor and lecturer. I would also like to thank Abdul-Aziz Ahmed, General Director of Ethio- Djibouti Railway Organization for providing essential data used under this thesis. The Last but not the least, I would like to thank everyone who ever contributed to the development of this thesis.

TABLE OF CONTENTS

ABSTRACT ..... **I**

ACKNOWLEDGMENT ..... **II**

TABLE OF CONTENTS ..... **III**

LIST OF FIGURES ..... **VI**

LIST OF TABLES ..... **VII**

LIST OF ACRONYMS ..... **VIII**

CHAPTER ONE INTRODUCTION **1**

1.1. Introduction 1

1.2. Single track railroad operation 2

1.3. Statement of the problem 3

1.4. Objectives 4

1.4.1. General objectives 4

1.4.2. Specific objectives 4

1.5. Research methodology 4

1.6. Algorithm overview 6

1.7. Structure of the Thesis 8

CHAPTER TWO LITERATURE REVIEW **9**

2.1. Theoretical background 9

2.1.1. Basic concepts 9

2.1.2. Definitions of terms used 9

2.1.3. Timetables and train diagrams 10

2.1.4. Decision support system (DSS) 11
2.1.5. Objective Function (Optimization Criteria) 11
2.1.6. Operational and safety constraints 12
2.1.7. Possible train conflicts 12
2.2. Literature Review on Re-scheduling and dispatching models 14

CHAPTER THREE MODEL DEVELOPMENT 17

3.1. System Architecture 17
3.2. Model description 18
3.3. Mathematical formulation 19
3.5. Selection of appropriate Algorithm 23

CHAPTER FOUR ALGORITHM IMPLEMENTATION 24

4.1. Programing language selection 24
4.2. Program overview 25
4.3. Input data 26
4.3.1. "SCHEDULE" sheet 26
4.3.2. "TRAINS_DATA" sheet 27
4.3.3. "TRACK_DATA" sheet 28
4.4. Conflict detection 28
4.4.1. Reduction of complexity in the detection algorithm 28
4.4.2. Crossing and overtake conflict Detection 29
4.4.3. Safety interval at station and capacity conflict detection 31
4.5. Conflict Resolution 32
4.5.1. Optimal Resolution (Exact method) 32
4.5.2. Optimization procedure 34

CHAPTER FIVE RESULT AND ANALYSIS 39

5.1. Result..... 39

5.2. Analysis..... 45

CHAPTER SIX CONCLUSION AND RECOMMENDATION.....47

6.1. Conclusion..... 47

6.2. Recommendations..... 48

REFERENCES.....49

APPENDICES 52

Appendix A: Inputs used..... 52

Appendix B: Test Result 55

Appendix C: Data Collected 57

Appendix D: MATLAB Source code..... 60

LIST OF FIGURES

Figure 1: workflow chart of the thesis 5

Figure 2: Train Diagram Example 10

Figure 3: System Architecture 17

Figure 4: Problem Description 18

Figure 5: Conceptual framework 25

Figure 6: Input Data Format 26

Figure 7: Start and Finish point of trains to meet points 29

Figure 8: General Optimizing procedure for DSS 34

Figure 9: Dispatching support system preview 40

Figure 10: Problem 1 initial train diagram 41

Figure 11: Problem 1 optimal solution 42

Figure 12: Problem 2 initial schedule 43

Figure 13: Problem 2 optimal solution 44

Figure 14: Problem 1 input schedule 55

Figure 15: Problem 1 optimal solution 55

Figure 16: Problem 2 input schedule 56

Figure 17: Problem 2 optimal solution 56

Figure 18: Trains schedule 57

Figure 19: Time Distance diagram 58

Figure 20: Train diagram of three trains working on 26-11-2017 59

LIST OF TABLES

Table 1: Example of a schedule sheet 26

Table 2: Example of first table in TRAINS_DATA sheet 27

Table 3: Example of second table in TRAINS_DATA sheet 27

Table 4: Example of TRACK_DATA sheet 28

Table 5: Segment start and finish times 30

Table 6: Train sorted in order of entrance time to segments 30

Table 7: Meet point sorting 31

Table 8: Input Schedule for problem 1 39

Table 9: Input Schedule Sheet for problem 2 43

Table 10: Priority of each train type 45

Table 11: Set of weights considered 46

Table 12: Variation in optimal solution cost 46

LIST OF ACRONYMS

B&B	Branch and Bound
DSS	Dispatching (Decision) Support System
FOFI	First Out First In
GUI	Graphical User Interface
Intlinprog	Integer Linear Programming
LINDO	Linear Interactive and Discrete Optimizer
LP	Linear Program
MATLAB	Matrix Laboratory
MILP	Mixed Integer Linear Program
MIP	Mixed Integer program

CHAPTER ONE
INTRODUCTION

1.1. Introduction

Due to pre-planning of its operations, railroad systems are considered to be more reliable, cost-effective, and safer than other means of transportation [1]. Conversely, in real time rail operations, the pre-planned schedule (timetable) for trains usually exhibits variations compared to the actual records on the route [2]. The cause of this deviation is due to technical failures like a signal problem, catenary problem, engine breakdown or other problems like staff shortage and weather condition [3]. Such unplanned circumstances result in increased dwelling times, running times, arrival and departing events [4]. Because of train interactions, these variations will also be propagated to other trains and brings interruption of the whole rail network specifically to the scheduled timetable [5].

This interruption inevitably causes infeasibility of the previous schedule. Consequently, it needs to be modified with the correct timetable comprising of new arriving and departing events from/to all stations with all conflicts resolved and normal traffic movement restored. This process demands effective solutions within a short duration of time and is called train dispatching or conflict resolution [1]. A feasible railroad timetable should specify the departure and arrival time of all trains to each location of its route so that the line capacity, safety, and other operational constraints are satisfied [6][7].

In the last few decades, the competition among transport service providers has created awareness of the need for quality service. Under such pressure for improvement, computer tools have been developed to help Dispatchers and planners to do their work more efficiently and quickly [5]. Those tools have also been able to assist train dispatchers in decision making process. However, search for a better and optimal solution is still drawing the attention of researchers [2][3][6].

In Ethiopia, the former railroad network (Ethio Djibouti railway Enterprise) which still runs between Dire Dawa and Djibouti (Daouenle), implements a manual approach to generate train timetable by drawing trains on the time-distance diagram, whereby the train diagram is manually

adjusted so that all associated conflicts resolved. Such a process can be time-consuming furthermore it doesn't ensure operational optimality [4][7].

In this thesis, we have developed a real-time and intelligent dispatching support system that detects a conflict on single track railroad network and provides optimal resolution. The foremost objective of this model was to minimize the overall delay arisen from the stopping of any train while solving those conflicts. The user-friendly characteristic of the GUI makes it easy to be used by any level of skill.

1.2. Single track railroad operation

Railways are built with a different number of tracks, depending on the traffic characteristics of the rail network. The most common railway tracks are single track and double track. Single line tracks can serve only one train at a time while double line tracks can serve up to pairs of a train at a time traveling in an opposite or same direction [8].

Specifically, on single line rail track trains are allowed to overtake(pass) and cross(meet) each other only at particular locations such as sidings and meet points sometimes referred as stations that are pre-established along the track line [9]. From an operation point of view, double line tracks operate very much simpler than that of single line tracks. Trains moving in a same direction use one single track and trains in opposing direction use the other. i.e., opposing trains do not exist if every train keeps its route as per the dispatching plan.

To conclude, in single line track operation, movement of opposing trains and fast train following slow train demands adequate dispatching tool to avoid colliding of a train with others within the network [1].

1.3. Statement of the problem

On single track railroads, where trains moving in both directions share the same track, overtake and cross operation has to be done only at stations or sidings. Meeting of trains outside of this region will eventually lead to rout conflict and cause train collision [9]–[11]. Additionally, for technical or external problems trains continue to exhibit deviation from the pre-developed schedule [3][12]. The disrupted schedule cause unstable railway operation and an increasing propagating delay. Thus, it is crucial to detect and resolve such abnormal conditions precisely to limit the damage likely to happen in the traffic system [4].

A train timetable is well planned to strictly meet rail traffic constraints such as track capacity and operational constraints. Train dispatcher uses this detailed timetable for dispatching trains within the system. The train dispatcher may encounter problems due to the existence of unforeseen circumstances in the journey of the train.

A train breakdown, for example, will cause unavailability of the occupied track for other scheduled trains and therefore, cause conflicts within the network which needs resolution on time and the timetable should be updated accordingly [1]. At this stage, the train dispatcher must reschedule involved trains completely to eliminate emerged conflicts and restore normal traffic movement.

Such a process requires adequate and intelligent dispatching support system that can generate a conflict-free schedule in a reasonable amount of time by considering both operational and safety constraints. The system also aims to minimize the time deviation of trains from the previously planned schedule [1][9].

Driven by such a complex problem, in this thesis real-time and intelligent dispatching support system is developed that detects a conflict on a rail network and provides optimal resolution. Beside detection and resolution of trains, the target of this model further extends to minimize the overall delay resulted from stopping of trains while resolving those conflicts.

1.4. Objectives

1.4.1. General objectives

- The general objective of this thesis is to optimize performance of train dispatching support system for a single-track system.

1.4.2. Specific objectives

- To investigate optimal solution approach for MILP
- To model an automated train dispatching support system.
- To examine conflict detection and resolution mechanisms.
- To investigate train operational optimality and safety.
- To study train dispatching tools.

1.5. Research methodology

The methods followed by the author to achieve the objectives of the research include the following core steps.

- Literature review, a study on optimization problems and their respective appropriate methods as well as depth understanding of the research problem.
- Necessary data and information required as input for the research collected.
- The problem formulated as an optimization problem with proper objective function and constraints.
- Appropriate algorithm and method selected for solving the problem, which is mixed integer linear programming with branch and bound technique.
- Selection of suitable software environment, MATLAB R2015a, was chosen for its advantage of containing built-in methods for solving optimization problems, for writing the program and implementation of the model.
- Performance evaluation carried out, mainly testing the developed application and performing result analysis.

The flow chart depicting the methodology is shown below in Figure 1.

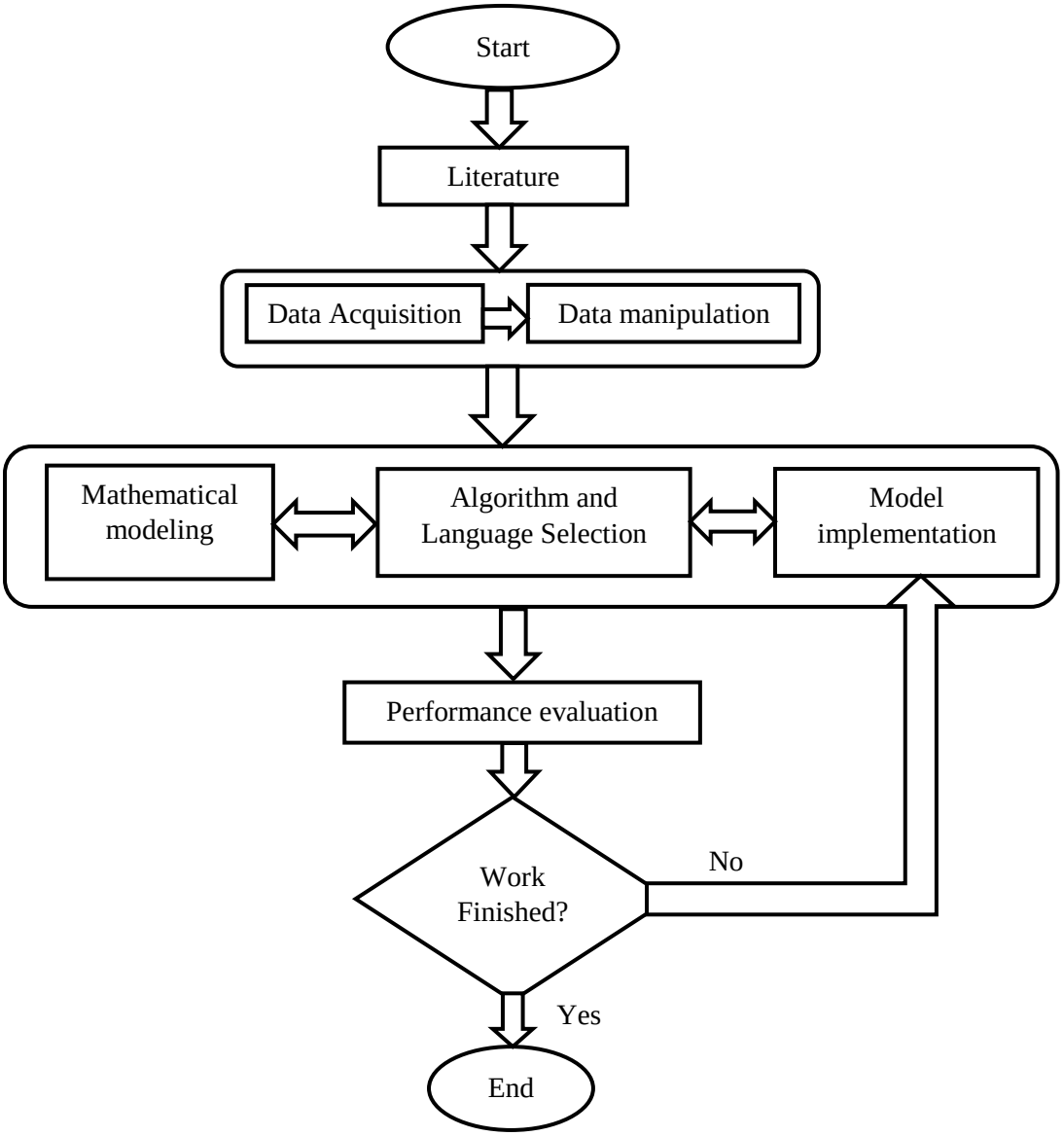

Figure 1: workflow chart of the thesis

1.6. Algorithm overview

In this section, mixed integer linear programming (MILP), an algorithm to be used for solving the MILP problem and solving approach to the solution will be introduced. The problem formulated under this study is of the form, mixed integer linear program. MATLAB built-in function under optimization toolbox named as Intlinprog is the essential solution package. It covers a bunch of core steps where all steps considerably simplify the subsequent move and let users customize the function to meet their need. Detail options on this application can be found by referring MATLAB optimization toolbox user's guide [13].

1.6.1. Mixed-Integer Linear Programming

A mixed-integer linear program is a problem with [13]

- Linear objective function, $f^T x$, where f is a column vector of constants, and x is the column vector of unknowns
- Bounds and linear constraints, but no nonlinear constraints
- Restrictions on some components of x to have integer values

In mathematical terms, given vectors $\boldsymbol{f}$, $\boldsymbol{lb}$ and $\boldsymbol{ub}$, matrices $\boldsymbol{A}$ and $\boldsymbol{Aeq}$ corresponding vectors $\boldsymbol{b}$ and $\boldsymbol{beq}$ and a set of indices **intcon**, find a vector x to solve

$$\min_x f^T x \text{ subject to } \begin{cases} x(\text{intcon}) \text{ are integers} \\ A \cdot x \le b \\ Aeq \cdot x = beq \\ lb \le x \le ub. \end{cases}$$

In order to solve the problem with MATLAB integer linear programming solver, all vectors and matrices should be written as per the equation mentioned on the definition of MILP. To mean, if the objective function is maximization problem it has to be multiplied by '-1' to convert into minimization and all inequality constraints have to be in '≤' form otherwise mathematical operation should be applied to convert in such format and so on [13].

1.6.2. The Branch and Bound method

The method creates a tree involving subsets of the solution set and identifies branches of the tree. These branches are also named as a node. Then, the method checks every node's LP relaxation by using an estimated upper and lower bound on the variables. If the recent Sub-Problem's solution is worse than preceding solutions, the method automatically discards that branch and explores the following branch to find the optimal solution. This iterative process continues until the method gets the integer optimal solution, which minimizes (or maximizes) the objective function [14].

1.6.3. MATLAB 'Intlinprog' Algorithm

intlinprog uses this basic strategy to solve mixed-integer linear programs. intlinprog can solve the problem in any of the stages. If it solves the problem in a stage, intlinprog does not execute the later stages [13].

1. Reduce the problem size using Linear Program Preprocessing [15].
2. Solve an initial relaxed (non-integer) problem using Linear Programming.
3. Perform Mixed-Integer Program Preprocessing to tighten the LP relaxation of the mixed integer problem.
4. Try Cut Generation to further tighten the LP relaxation of the mixed-integer problem [16].
5. Try to find integer-feasible solutions using heuristics [17].
6. Use a Branch and Bound algorithm to search systematically for the optimal solution. This algorithm solves LP relaxations with restricted ranges of possible values of the integer variables. It attempts to generate a sequence of updated bounds on the optimal objective function value. Detail information about the methods can be found from.

1.7. Structure of the Thesis

This research thesis is organized into six chapters, the first one introduces the thesis, and the approach used by the author to accomplish the main objectives of the research.

Chapter 2 presents a literature review on similar and related works regarding train dispatching and rescheduling models to provide an insight about the problem that this thesis is going to address.

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 a description of how the program was implemented and the choice of suitable programming language for this particular problem. This chapter also intends to provide sufficient information regarding the entire process to enable readers to get a complete understanding.

Chapter 5 will give the result of the thesis as well as analysis based on the outcome of the developed model. The last one, Chapter 6 discusses the conclusions and recommendation for future research.

CHAPTER TWO
LITERATURE REVIEW

2.1. Theoretical background

2.1.1. Basic concepts

In this section of the thesis, similar and related works regarding train dispatching and rescheduling models are reviewed to provide an insight about train dispatching and rescheduling problem that this thesis has addressed. Before presenting those most relevant dispatching model's fundamental aspects of train dispatching and definitions of commonly used terms presented.

2.1.2. Definitions of terms used

Some of the terms commonly used in this thesis are defined below.

Re-scheduling: is the modification of the timetable to take account of disturbances such as lateness of trains and breakdowns.

Minimum Headway: is a minimum distance or time separating two trains on a single line track. For trains traveling in opposite direction, minimum headway is the time or distance required for one train to clear the track segment sufficiently before the opposite train enters that track. Most of the time for same directional trains minimum headway is determined using signaling [18].

Crossing loops: is a section of single track at intervals along the line to allow trains running in different directions to pass each other and may also be used to allow trains heading in the same direction at different speeds to overtake. If a single-track line is designed to be used by more than one train at a time, it must contain passing loops (also called passing sidings or crossing loops).

Train Conflict: There are two situations, namely: when two trains approach each other on a single line track and the continuation of either or both trains journey would not be possible; and when a faster train catches up a slower train traveling in the same direction.

2.1.3. Timetables and train diagrams

Timetable and train diagrams are two principal tools used by dispatchers for surveillance of train movement in a railway network. A train timetable defines the planned arrival and departure times of trains to/from yards, terminals, and sidings. Train scheduling plays a vital role in managing and operating complex railroad systems.

Figure 2: Train Diagram Example [22]

The train diagram, or graphical timetable, is another representation of the schedule in a more insightful way. As shown in Figure 2, it is a representation of all train route in a time-distance

graph which is very simple and intuitive to see the timetable as well as to read conflict [10], [19]–[21].

2.1.4. Decision support system (DSS)

A Decision Support System is a computer-based train dispatching support system or a computer-aided interactive tool that assist train dispatchers and railway planners in the construction of dispatching plans(schedule) or the process of decision making. Moreover, it is an automated dispatching support system, which is a real-time traffic management system that aids dispatchers to control the train traffic. It is responsible for rescheduling train movements, minimizing future delay and ensuring the feasibility of the resulting train schedule. The conventional DSS has two principal functions: detect all conflicts, and propose solutions to the conflict found [4][10][23][24].

2.1.5. Objective Function (Optimization Criteria)

When dealing with optimization problems, the choice of the objective function to be minimized or maximized directly influence the quality of the final solution. The size of the problem, the characteristics of the objective function, and constraints are some of the other factors that can directly affect the efficiency and accuracy of the solution to these problems.

The following objective functions classified as the most common [25].

1. Minimization of the maximum tardiness $T_{max} = max\{T_1, T_2, \dots, T_n\}$ [T_n is tardiness of train n]. Although the criterion minimizes the maximum delay, many trips may suffer disturbances. The criterion is acceptable in situations where passenger trains prevail for scheduling.
2. Minimization of the maximum weighted tardiness $WT_{max} = max\{w_1T_1, w_2T_2, \dots, w_nT_n\}$ is an interesting criterion under the mixed traffic conditions [w_n is priority]. The weights are usually the same for all trains of the same category within the given scheduling period.
3. Minimization of the total tardiness $D = \sum_{i=1}^{n} T_i$.
4. Minimization of the total weighted tardiness $WD = \sum_{i=1}^{n} w_i T_i$.
5. Minimization of make span $C_{max} = max\{C_1, C_2, \dots, Cn\}$ [C_n is completion time of train n]. Expresses the wish for the trains to leave as soon as possible the fragment of railway network under consideration. In the case of rescheduling, this objective function gives support to the localization of disturbances.

6. Minimization of the number of late jobs$|LJ|$, where $=LJ = \{J_i \in J \mid C_i - c_i > 0\}$. [$C_i$ is actual and c_i is planned job completion time of job J_i, LJ is late job].

2.1.6. Operational and safety constraints

The train dispatcher uses the DSS to generate a conflict-free and feasible timetable for the railroad system. The timetable allocates departure and arrival times for all trains within the network. To build a feasible, conflict-free timetable and to avoid any potential conflicts, all safety, and operational constraints must be strictly respected.

Some of the significant scheduling constraints used are [26]–[28].

- **Speed constraints**: Trains within the network should not exceed the maximum allowed or maximum safe speed specified for it to use on any track during their journey.
- **Station headway constraints**: There must exist fixed time interval between two trains arriving or departing a station using same track line.
- **Station capacity constraint:** The station capacity should exceed the number of trains at the station at any instant of time.
- **Other topological constraints**: All crossing and overtakes on a single rail network should be taken place on fixed passing and crossing loops or station.

2.1.7. Possible train conflicts

Timetable conflicts occur when the trains within the network no more abide the scheduling rules. A possible cause of this problem can be a breakdown of a train or stochastic events. Whenever such problem is detected, the DSS will use the proposed algorithm for solving the problem and generate a new conflict-free timetable. Possible timetable conflicts are [26][1]:

Trains meeting conflict: Occurs when trains moving in opposite direction on the same track are about to pass the same fixed point before the meet point. One way of solving this problem is by using signal systems. The signal system of the railway system is used by granting priority to one of the trains to use the track at a particular time while the other train waits for the route to be unoccupied before using it.

Pass (overtake) conflict: Occurs when a fast train catches a slower train moving in the same direction. Such conflict can be prevented by keeping the safe time or distance required between the trains to reach the next station.

Station capacity conflict: All stations within the railway system have limited number of tracks for trains available for parking. If a train needs to use the track for some time and all available tracks in the stations are occupied, this problem is said to be station capacity conflict.

Evidently, for solving the conflicts listed so far, a tool having an objective of resolving the conflict and enhancing the performance of the network is ultimately required. When a feasible solution is found using the solution technique, modification of the original timetable will be made. This process is known as rescheduling.

2.2. Literature Review on Re-scheduling and dispatching models

A wide range of studies in the past has conducted focusing on the mathematical formulation and solution algorithm for train scheduling problems. Some of the literature found focused on minimizing train traveling time, estimating train delay from unforeseen incidents based on historical data, traffic distribution, and track topology and so on.

One way or another, researchers in the past have studied issues related to the development of computer-based train dispatching support system, or computer-aided tools that assist train dispatchers and railway planners in the construction of dispatching plans (schedule) in simple term Decision support system (DSS). Here under researches most relevant to the issue of mathematical formulation, solution algorithms and DSS development are presented in chronological orders:

Higgins et al. [21] formulated train scheduling problem on a single line as a mixed integer linear program aiming to minimize total conflict delay. They proposed a heuristic approach to solve the optimization problem because of the problem being NP-Hard and the difficulty to obtain an optimal solution within a reasonable duration of time. The heuristics considered were local search heuristics, genetic algorithm, tabu search, and two hybrid algorithms. Comparisons between those methods were also made to investigate the performance of each algorithm. On the thesis, the two hybrids have shown a better result than the other heuristics. For more than ninety percent of the test results, the second hybrid named HA2 has produced solutions within the range of one percent from the optimal.

Sahin [3] studied a system approach for the construction of heuristic algorithm to reschedule trains. It compares expected time of arrival at next meet point (or actual arrival time if the train is at that meet point) with the planned arrival time from the pre-established schedule for detection of any deviation from the schedule. They have used discrete event simulator for determining the contrary trains and the meat section where the immediate conflict is going to occur. The time of occurrence and exact location has determined by using continuous simulation. To resolve a conflict; the algorithm stops the train with the higher future consequential delay. Then the proposed heuristic approach continues to detect and resolve one conflict at a time.

Adenso-Díaz et al. [3] developed a real-time decision support system for the Spanish National Railway company. They studied the average duration of the incident based on historical data on the regional network in Asturias and the solution alternative to the problem. They also presented a MIP (mixed integer program) model with the objective of maximizing the number of passengers transported with several practical constraints. The complexity of the resolution problem shown using integer programming on account of the size it acquires especially for resolving real cases. Considering this fact design of constructive-type procedure based on exploration of solutions space was taken as the only practical alternative. The system was implemented in 1998 on the Austrian regional rail network and has been said to offer a set of useful solutions in less than five (5) minutes.

On the study by Corman et al. [29] an approach for solving the problem of coordinating the task of multiple dispatchers in the presence of disturbances was presented. The problem was formulated as a bilevel program moreover the objective of the formulation was to minimize delay propagation. An aggregate coordinator graph is adopted to model coordination constraints while detailed dispatcher graphs model the problem in each dispatching area. They used a branch and bound algorithm to solve the mathematically formulated problem given on the study. They find distributed approaches to be able to deliver better solutions than a centralized approach from the computational results. Good resolutions were offered in a short amount of computation time, compatible with real-time management. However, in the case of huge instances, the distributed approaches are said to fail in finding a globally feasible.

Zhou and Zhong [21] studied single track timetabling problem with the objective of minimizing total train traveling time subjected to a set of safety and operational constraints. In the research, three different approaches were adopted to reduce the solution space effectively. The branch and bound solution procedure were presented for obtaining optimal schedules through implicit enumerations. The search algorithm chronologically adds precedence relation constraints between opposing trains to eliminate conflicts, and the resulting sub-problems are solved by the longest path algorithm to determine the earliest start times for each train in different segments.

Dotoli, et al. [30] studied A real time traffic management model for regional railway networks under disturbances. They addressed real time traffic management under disturbances of regional

rails with a centralized traffic control system. they solved the rescheduling problem by revisiting a finite time horizon mixed integer linear programming model from the related literature. First, they adapted the framework to regional networks, in which stations are close and the network is mainly constituted by single tracks; second, to solve train conflicts that may occur in the rescheduled timetable after the chosen time horizon, they enhanced the model by an iterative heuristic algorithm that solves such conflicts. The presented approach is applied to a real data set related to a large portion of a regional network in Southern Italy, showing its effectiveness in providing a physically realizable rescheduled solution in a very short computational time.

Another interesting work was presented by Afonso and Bispo [10] which studied use of Branch and bound approach to solve Meet and pass problems. On the research paper two separate models namely, Heuristic and search for an optimal solution were developed. They have designed a separate algorithm for both models. The Heuristic Approach is fast and effective but is a myopic rule which evaluates a given conflict concerning only train priorities and FOFI rule. The second one, the search for an optimal solution, uses a branch and bound technique with the method called depth-first search, to save computer memory and to reach the solution faster even if it is not the optimal one. On the thesis, good results were revealed using Hypothetical tests.

To conclude, most of the research papers in the literature used different types of heuristic approaches that attempts to find near-optimal solution quickly for finding the solution to the rescheduling problems. The characteristics of the problem of being NP-Hard mentioned as the main reason behind this. Search for an optimal solution is identified as the gap in the literature.

But given the recent environment, where everything is updating consistently, the author of this thesis used MATLAB R2015b which consists of a built-in function called “intlinprog” for solving MILP problems optimally and effectively. The solver was added to the MATLAB optimization toolbox in 2014.

CHAPTER THREE

MODEL DEVELOPMENT

3.1. System Architecture

An optimal solution approach toward decision-making problems involves three core steps. Constructing an appropriate mathematical model, selecting an efficient algorithm for solving that model, and finally implementing the solution obtained by the algorithm. These core steps are discussed under one by one.

The overall system architect is depicted in Figure 3 below, showing the type of information exchanged with the dispatcher and the incorporation of the decision support system with the whole dispatching process.

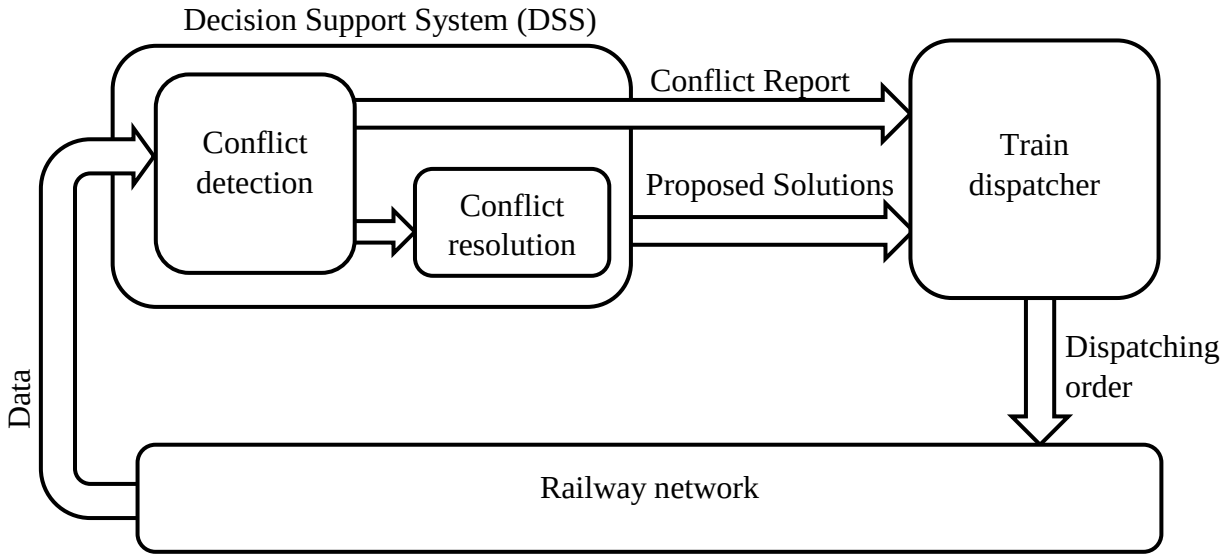

Figure 3: System Architecture[31][10]

This model is a very useful tool for dispatchers to enable them to analyze the consequence of their decision before implementing and provides all other feasible solutions. Thus, train dispatchers will have more option to consider and to select the one which best fits with the rules and regulation predefined by the railway authority.

3.2. Model description

Before modeling the system, it is necessary to define and discuss the problem environment and assumptions. This thesis considers a single-track railway that serves trains traveling in both directions on a single rail line. In normal operations, trains are allowed to meet or pass each other only at stations or sidings otherwise conflict is said to have occurred. Thus, a meet point is not just station but also sidings or any other point on the track where two trains meet or pass each other. This assumption makes the traffic control and management, complex, realistic, and a very crucial issue for safety reasons.

Directions are defined as westbound for trains going from right to left and eastbound for the reverse movement. Numbering for meet point and track segment is made in eastbound directions.

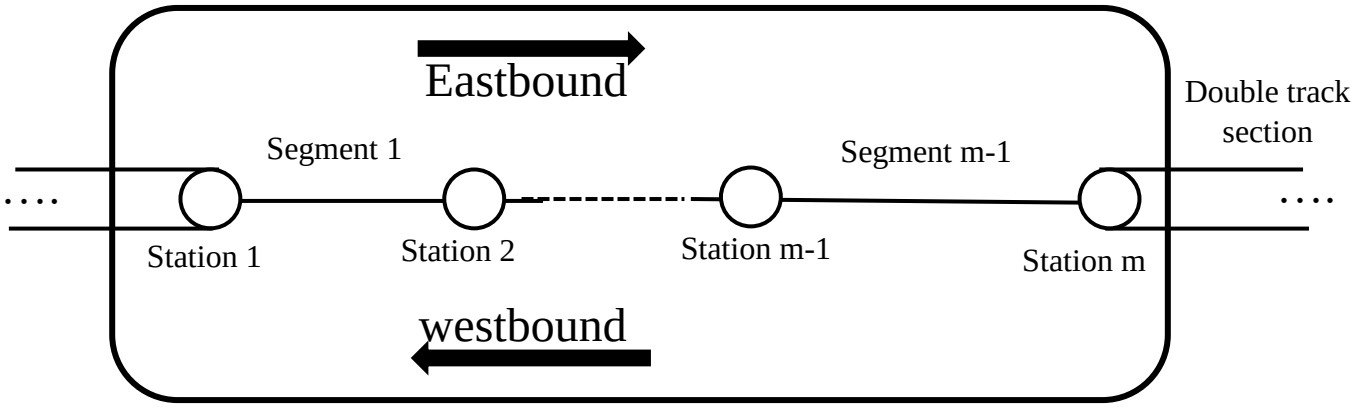

Figure 4: Problem Description[10]

As shown Figure 4, the first and last meet points are either interface to a double track or to a denser rail network. Either way, safety time interval between arrival and departure at these stations remains unchecked. The safety technology considered is fixed block i.e. trains can follow each other on track segment keeping minimum safety headway. Three types of train priorities are chosen for this model namely, fast passengers train, slow passengers train, and a freight train.

3.3. Mathematical formulation

Variable Notations

p_i	Weighted priority of train i
d_i	Total tardiness of train i
$a_i(m_e)$	Actual arrival time of train i at terminal meet point(m_e)
$r_i(s)$	Running time of train i at segment s
$w_i(m)$	Dwelling time of train i at meet point m
$D_i(m)$	Departure times of train i at meet point m
$A_i(m)$	Arrival times of train i at meet point m
$C(m)$	Number of trains at meet point m

Parameters

n	Number of trains
m	Number of meet points
s	Number of track segments
m_e	Terminal meet point
a_i	Actual arrival time of train i
s_i	Scheduled time of trains
π_i	Minimum running time of train i
X_i	Maximum running time of train i
μ_i	Minimum dwelling time of trains
α_m	Minimum allowed headways b/n arrival and departure time at meet point m
β_s	Minimum allowed headways b/n arrival and departure time at segment s
$w(m)$	Capacity of meet point m
I_o	Eastbound train

I_i Westbound train

I Integer

Subscripts (Indices)

i =1, 2… n.

m=1, 2 … m_e, m = m_e.

s= 1, 2 …$m_e - 1$.

3.4. Constraints and optimizing criteria

The selected objective function for this particular problem is a minimization of total weighted tardiness [4][10]. It aims to minimize the total delay of all trains arriving at terminal stations. As shown in Equation (1) the objective function is the multiplication of delay of all trains with their corresponding priority. The weight of the priority ensures that a train with higher priority if involved in a conflict, will be given priority to go first.

Minimize

$$\sum_{i=1}^{n} p_i * d_i \tag{1}$$

The total weighted delay is further defined as a difference of actual arriving time at the terminal station from scheduled arrival time. If a train arrives before the scheduled time, then the delay is considered as zero. Actual arriving time is also defined as summations of all dwelling and running times at stations until the terminal station.

where

$$d_i = \begin{cases} a_i(m_e) - s_i(m_e), & if\ a_i(m_e) > s_i(m_e) \\ 0, & other\ wise \end{cases} \tag{1a}$$

$$a_i(m_e) = \sum_{m=1}^{e} w_i(m) + \sum_{i=1}^{m-1} r_i(s) \tag{1b}$$

$$w_i(m) = D_i(m) - A_i(m) \tag{1c}$$

$$r_i(s) = A_i(m+1) - D_i(m) \quad (1d)$$

Subjected to:

Speed constraints

$$r_i(s) \geq \pi_i(s) \wedge\ r_i(s) \leq X_i(s), i\ = 1, 2, \dots, \mathrm{n}, \mathrm{s} = 1,2, \dots \mathrm{n}-1 \quad (2)$$

Arrival and departure constraints

$$A_i(m+1) \geq D_i(m) + \pi_i(s)\ \text{, for } i \in \mathrm{I} \quad (3)$$

Minimum Dwell time constraint

$$w_i(m) \geq \mu_i(m)\ \text{, for } i \in \mathrm{I} \quad (4)$$

Station entrance constraint

$$A_i(m) \geq A_{i'}(m) + \alpha_m \ \oplus\ A_{i'}(m) \geq A_i(m) + \alpha_m \text{ for } i \in I, i \neq i' \quad (5)$$

Trains Crossing (meeting) constraint

$$D_i(m+1) \geq A_{i'}(m+1) + \alpha_m \ \oplus\ D_{i'}(m) \geq A_i(m) + \alpha_m \text{ for } i \in I, i' \in I_o, i \in I_i \quad (6)$$

Trains passing constraint

Case one; if both trains are eastbound.

$$\begin{aligned}(\ D_i(m) \leq D_{i'}(m) + \beta_s\ \wedge A_i(m+1) \leq A_{i'}(m+1) + \beta_s\ ) \oplus (\ D_{i'}(m) \\ \leq D_i(m) + \beta_s\ \wedge A_{i'}(m+1) \leq A_i(m+1) + \beta_s\ )\ \ i, i' \in I_o\end{aligned} \quad (7)$$

Case two; if both trains are westbound.

$$\begin{aligned}(\ A_i(m) \leq A_{i'}(m) + \beta_s\ \wedge D_i(m+1) \leq D_{i'}(m+1) + \beta_s\ ) \oplus (\ A_{i'}(m) \\ \leq A_i(m) + \beta_s\ \wedge D_{i'}(m+1) \leq D_i(m+1) + \beta_s\ )\quad i, i' \in I_i\end{aligned} \quad (8)$$

Station capacity constraint

$$C(m) \geq w(m) \quad (9)$$

Constraint (2) ensures that running time at each track segment should be greater than the minimum allowed running time and all trains within the network should not exceed the maximum speed limit or maximum safe speed specified for it to use on any track during its journey. Constraint (3) checks for an operational requirement that, the arrival of any train at any station should always be later than its departure from the previous station. Constraint (4) ensures that all train should depart a station only after completing their minimum dwell time at a station. Constraint (5) assures that, no simultaneous arrivals at a station without keeping minimum safety interval.

Constraint (6), (7) and (8) are the crucial ones to consider because they guarantee that all trains crossing and overtakes happen at intermediate stations or sidings without problems or accidents for a smooth operation of the system. Specifically, constraint (6) assures that crossing of trains toward different direction shall happen only at a station and only after elapsing the arrival of the other train and the safety time interval together. Constraint (7) and (8) allows two eastbound or westbound trains to follow each other at any given segment but only if they respect safety headway throughout the entire segment track until they arrive the next station. Constraint (9) monitors the maximum number of trains that the station can handle.

Generally, a feasible schedule should respect all of the above restriction. If anyone of these restrictions is violated then conflict is detected and the schedule is no more feasible. The Dispatching support system is responsible for the detection of any violations and optimally resolving of those conflicts as well as providing a quick but “good enough” options for train dispatchers to empower them to deal with real-world rescheduling problems.

3.5. Selection of appropriate Algorithm

Optimization techniques are used to find a set of design parameters, $\boldsymbol{x = \{x_1, x_2 \ldots xn\}}$, that can in some way be defined as optimal. In the context of this thesis, the design parameters are set of all events that give an optimal schedule. It consists of all arrivals, departures, dwell times, running times and delay times. The objective is to minimize sum of delays that arise from all trains measured on their destination subject to constraints listed earlier. These constraints can be categorized under equality constraint, inequality constraints, and parameter bounds, xl, xu (lower and upper bounds).

An efficient and accurate solution to this problem depends not only on the size of the problem in terms of the number of constraints and design variables but also on characteristics of the objective function and constraints. When both the objective function and the constraints are linear functions of the design variable, the problem is known as a Linear Programming (LP) problem. If some of the design parameters are needed to be integers, the problem will be a Mixed integer linear programming (MILP). This is exactly the characteristics of our problem, i.e. the objective function as well as all constraints are linear and for practical reasons, all elements in the train schedule should be integers. Apparently, the mathematical formulation presented earlier is called a mixed integer linear program (MILP**)**.

MATLAB has introduced the solver called "*intlinprog*" for solving mixed integer linear programs in its optimization toolbox since 2014. The MILP has become a much-known mathematical language in the literature since the 1960th. With this in mind, there are also many other optimization software's for solving MILP problems besides, the capability of the intlinprog solver to deal with a large number of variables and availability of easily customizable and vast options suits to be a choice for this problem.

CHAPTER FOUR

ALGORITHM IMPLEMENTATION

4.1. Programing language selection

In this thesis, MATLAB is used to solve the Optimization problem and to develop the overall interactive train dispatching support system. Hereunder some of the reasons are discussed.

MATLAB is a software and programming language specially oriented for matrix manipulations thus it performs faster numerical calculations faster than other programming languages. In the rescheduling process, the most task is to manipulate timetables (other forms of matrices). This present one significant advantage of using this software.

MATLAB optimization toolbox contains easily customizable options for users and built-in heuristic procedures that may simplify the solution procedure and decrease solution time in some cases. Its user friendless and capability to deal with a large number of variables made it the choice for this problem.

For example, among other optimization software's like LINDO, Gurobi [32], Mathcad [33], MOSEK, GAMS, OptimJ and AMP, if we see the classic free version of LINDO software, according to their official website, it can only handle problems with maximum of 150 constraints, 300 variables and 30 integer variables in an ordinary linear optimization problem.

Being train diagram creation and graphical user interface development important feature of this work, the GUI inside MATLAB and its libraries will ease and make the application more interactive to train dispatchers.

4.2. Program overview

The overall dispatching support system contains two major sections, conflict detection, and conflict resolution. The conflict detection section of the program first takes all the necessary inputs mainly the initial train schedule. Using the detection algorithm, it will search whether there is any potential conflict or not. The conflict resolution section of the program as illustrated in Figure 5 deals with solving the conflict and generating an optimal conflict-free schedule.

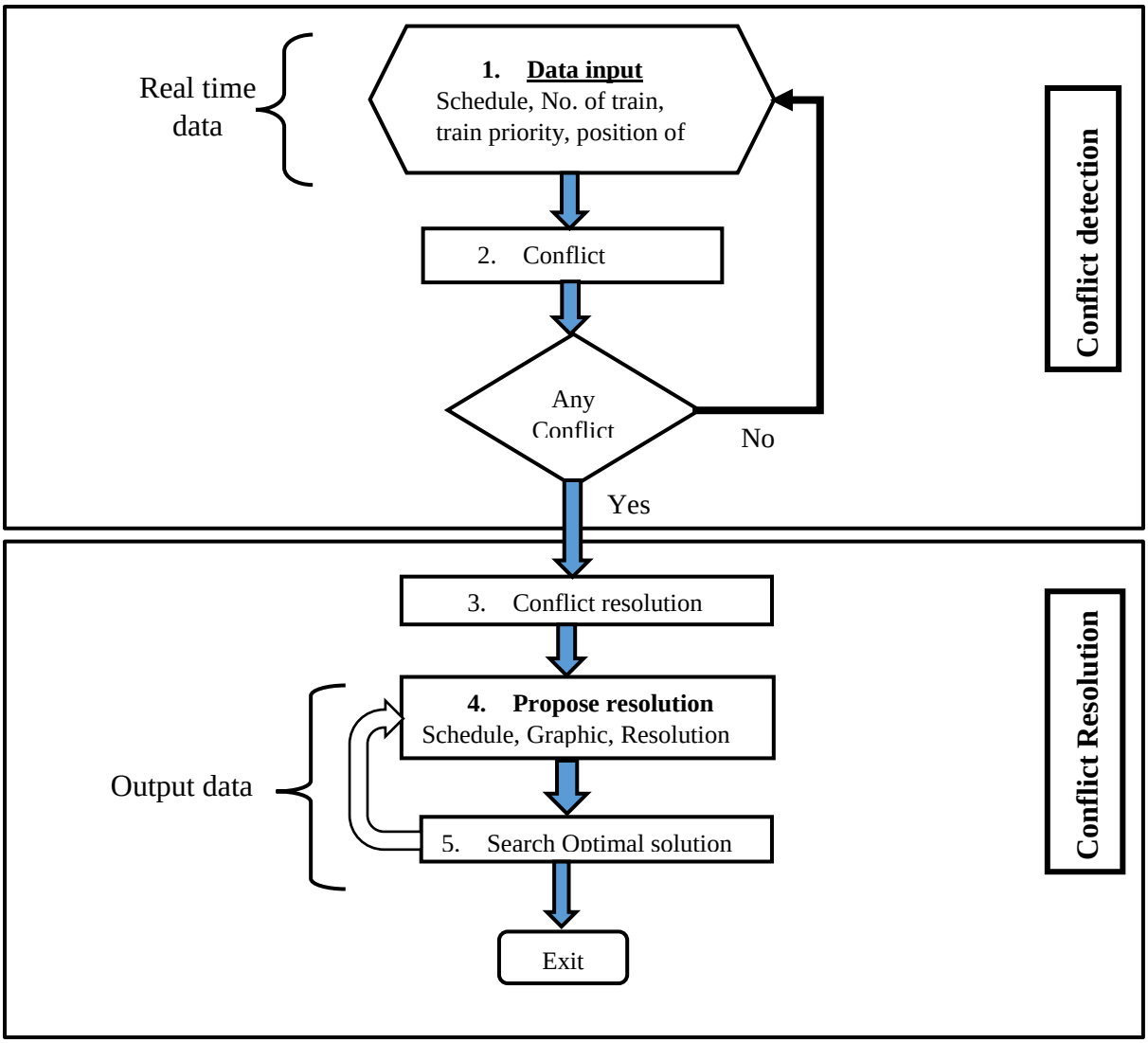

Figure 5: Conceptual framework

The program in Figure 5 contains:

1. Input data manipulation
2. Conflict detection
3. Conflict resolution
4. Optimal resolution

Section 4.3. will explain the feature of each program division.

4.3. Input data

This section of the program is intended to make all data inputs available and easily accessible. All input data formats are in Excel (.xlsx) and should be placed on a single Excel file but with separate sheets. Typical input data format is given in Figure 6.

This file contains three sheets namely SCHEDULE, TRAINS_DATA, and TRACK_DATA that comprises data for schedule, trains, and the track information.

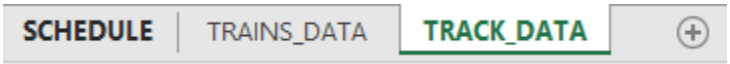

Figure 6: Input Data Format

4.3.1. "SCHEDULE" sheet

Example of a schedule sheet is given in Table 1 below. It should be noted that the interactive dispatching support system accepts a schedule with any number of trains and stations. But the order of eastbound and westbound trains should be as per the table below.

Table 1: Example of a schedule sheet

	Train No.	Meetpoint1		Meetpoint2		Meetpoint3	
		ARRIV1	DEPART1	ARRIV2	DEPART2	ARRIV3	DEPART3
eastbound trains							
Train 1	424	0	10	15	15	22	1000
Train2	10	0	15	30	50	65	1000
Train3	12	0	30	36	42	48	1000
westbound trains							
Train4	17	42	1000	36	37	0	32
Train5	105	65	1000	52	58	0	46
Train6	15	70	1000	10	50	0	5

As illustrated in Table 1, In addition to the train number, arrival time and departure time, the schedule contains information about the number of stations at the upper side of Table 1. The 'train number' in the upper left side of the table indirectly carries information about the type of trains and directions. All times presented in the table are in minutes. If a train is at a station during the start of the simulation zero is assigned as a start time also if a train waited at a station for a long, large number is allotted as departure time (In this example it is **1000**).

4.3.2. "TRAINS_DATA" sheet

The input excel file also have to contain all the necessary data needed regarding train information on its second sheet named, TRAINS_DATA as illustrated on Table 2 and Table 3. The information includes type of train (fast passenger (1), slow passenger (2) or freight train (3)) (refer Table 3) number of eastbound and westbound trains (refer to Table 2), train number (train identification). This sheet consists of two separate tables as shown in Table 2 and Table 3.

Table 2: Example of the first table in TRAINS_DATA sheet

number of westbound trains	**3**
number of eastbound trains	3

Table 3: Example of the second table in TRAINS_DATA sheet

TRAINS DESCRIPTION	
ID	TYPE
424	1
10	2
12	1
17	1
105	1
15	3

Here increase or decrease in the number of trains does not affect the developed program, only the user needs to modify the input on the excel file and rerun the application for any updates.

4.3.3. "TRACK_DATA" sheet

This part of the input as displayed in Table 4 should contain station information including the name of each station, the capacity of the station and distance between stations. It is placed on the third sheet and has only one table.

Table 4: Example of TRACK_DATA sheet

STATIONS DESCRIPTION			
NAME	**CAPACITY**	**DISTANCE**	**SEGMENT**
E1	12	0	--
E2	3	6	S1
E3	8	20	S2

Table 4 shows sample hypothetical data. Distance between station is measured in kilometers from trains starting station and capacity is measured with number of trains at the station.

4.4. Conflict detection

In chapter two, possible types of timetable conflicts were discussed in detail. This section will present the detection mechanisms for any conflict or in another term detection of any violations of the traffic rules. To recall four conflicts were considered in this thesis, crossing (trains meeting) conflict, overtake (trains passing) conflict, capacity conflict and safety interval at station conflict.

4.4.1. Reduction of complexity in the detection algorithm.

To detect crossing and overtake take conflicts, there are two critical actions taken for reducing the complexity of the problem for saving time and computer memory. i.e., a blind scan was not performed to search a conflict upon all trains in the schedule. Instead, adequate and intelligent search for potential candidates of conflict has carried as stated below.

1. The first approach toward the complexity reduction is to make the algorithm to check whether the potential trains are in the same direction or not. If they are toward the same direction only potential overtake conflict will be carried. Otherwise, for trains moving in different directions a potential crossing conflict alone will be examined.

The pseudocode is given below.

```
for all segments;
 for all candidate combination; %only candidate combinations
    % check if the two trains are in the same direction
    if (the two trains are in the same direction)
        % check only for overtake conflict
      If (overtake conflict)
         Report overtake conflict;
      end

    else % the trains are in a different direction.
        % check only for crossing conflict
      if (crossing conflict)
          Report crossing Conflict;
      end
 end
end
```

2. The second approach: it is unnecessary to check each constraint to detect conflicts between all trains in the schedule. Preferably, it is enough to sort trains in entering orders into segments and checking the constraints only between consecutive trains[10]. Using this approach, the original problem which was a combinatorial problem (i.e., for n trains there were C_2^n comparisons) it can be reduced into n-1 comparisons only.

4.4.2. Crossing and overtake conflict Detection

The procedure to detect either of the two mentioned conflicts is first to sort the timetable according to entering times in ascendant order into segments (track section between two meet points) and secondly to check the cross and overtake constraints between consecutive trains. Figure 7 pictures the segments, meet points, start and finish times.

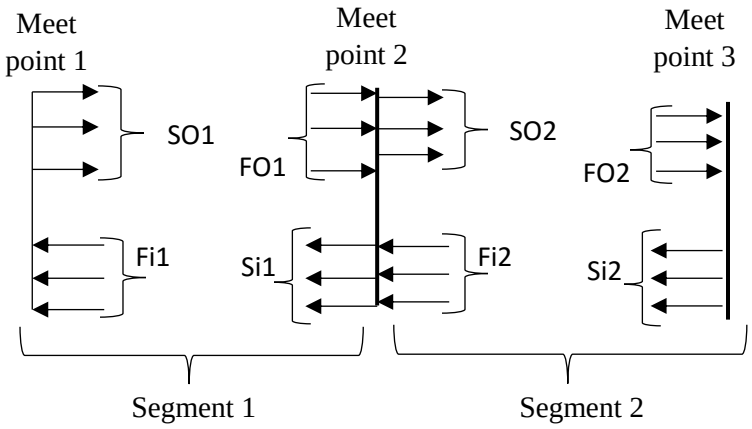

Figure 7: Start and Finish point of trains to meet points

SO1, FO1 = Start and finish time of eastbound trains from/at segment one respectively
Si1, Fi1 = Start and finish time of westbound trains from/at segment one respectively
SO2, FO2 = Start and finish time of eastbound trains from/at segment two respectively
Si2, Fi2 = Start and finish time of westbound trains from/at segment two respectively.

For the case of the previous example in Table 1, there are three meet points. Thus, there exist two track segments because the segment track is always one less than the total meet points as illustrated in Figure 7. The example in Table 1, was written in the form of arrival and departure time from meeting points. The detection algorithm proceeds by writing the table in a track segment arrangement as in Table 5 and sort according to entering orders to the segments as displayed in Table 6.

Table 5: Segment start and finish times

	Train No.	Segment 1		Segment 2	
		Start	**Finish**	**Start**	**Finish**
Eastbound trains					
Train 1	424	**10**	**15**	**15**	**22**
Train2	10	**15**	**30**	**50**	**65**
Train3	12	**30**	**36**	**42**	**48**
Westbound trains					
Train4	17	**37**	**42**	**32**	**36**
Train5	105	**58**	**65**	**46**	**52**
Train6	15	**50**	**70**	**5**	**10**

A track segment is part of the railway line that is bounded by two meet points or stations. Therefore, both eastbound and westbound trains start their journey from either side of the segment. For single track railways entering times to segment matters a lot. Opposing trains are not authorized to enter the same track segment at a time. Additionally, trains running in the same direction are not entitled to enter the same track segment without minimum safety headway.

Track segment one is bounded by meet point one from the left and meet point two from the right. Therefore, from Table 1 only eastbound trains (trains 1-3) from the left and westbound trains (trains 4-6) from the right start their journey at this specific track segment. For track segment two, it is bounded by meet point two from the left and meet point three from the right. Here under Table 6 shows trains sorted according to entering time to each track segment.

Table 6: Train sorted in order of entrance time to segments

Segment one			Segment two		
	Train no.	Entering Time		Train no.	Entering Time
Train 1	424	10	Train 2	10	50
Train 2	10	15	Train 1	424	54
Train 3	12	30	Train 3	12	62
Train 4	17	149	Train 5	105	66
Train 6	15	190	Train 6	15	105
Train 5	105	300	Train 4	17	136

Here above in Table 6, trains have been sorted according to their entering orders to the track segment. Next, for each pair of trains as suggested in section 4.4.1 for reduction of complexity, the successive pairs of trains are examined for the occurrence of any pre-defined cross and overtake conflict.

4.4.3. Safety interval at station and capacity conflict detection.

To detect safety interval at station conflict, first both arrival and entering times of trains to meeting points will be sorted in ascending orders as illustrated in Table 7, then three types of possible conflicts will be examined. These conflicts may arise between consecutive two departing trains, two arriving trains and within one arriving and one departing trains.

Table 7: Meet point sorting

Train No.	Meetpoint1		Meetpoint2		Meetpoint3	
	Arriving order	Departing order	Arriving order	Departing order	Arriving order	Departing order
Eastbound trains						
424	1	1	2	1	4	4
10	2	2	3	4	6	5
12	3	3	4	3	5	6
Westbound trains						
17	4	4	5	2	1	2
105	5	5	6	6	2	3
15	6	6	1	5	3	1

Table 7 shows order of the trains while arriving and departing from or to meet points.

For the arriving and departing order given in Table 7,, after correctly sorted, safety interval between the first arriving and first departing trains will be checked and also for simultaneous departure and arrival trains. For the case of detecting capacity conflict, train count will be taken at the station simultaneously with the detection of safety headway at the station.

4.5. Conflict Resolution

For conflicts found so far on the detection section of the program, the exact method that finds an optimal solution to the problem no matter how long it takes was deployed to solve each conflict. Detail solution procedures are discussed next.

4.5.1. Optimal Resolution (Exact method)

As described in chapter three, the mathematical model to be solved is a mixed integer linear program. Based on the definition given for MILP in section 1.6, the objective function and all constraints should be in the form of matrices and vectors for the problem to be solved by the intlinprog method.

For any optimization problems the decision variables should also be known first to solve the problem, the challenge of this particular problem is that the variables are dynamic with a change in input. i.e., for each change in input timetable, the number of variables, the size of matrices and vectors of A, Aeq, b, beq, f, and intcon changes exponentially. Thus, the decision support system should be intelligent enough to understand the change in input and to act accordingly.

There are only three forms for the equations in the definition of MILP given in section 1.6 namely equality, inequality, and bounds. Rearranging our formulation given in section 3.3 to change into those forms the final formulation becomes as follows.

Minimize

$$\sum_{i=1}^{n} p_i * d_i \qquad (10)$$

Subjected to:

$$d_i - a_i(m_e) + s_i(m_e) = 0 \quad (11)$$

$$a_i(m_e) > s_i(m_e) \quad (12)$$

$$d_i \geq 0 \quad (13)$$

$$a_i(m_e) - \sum_{m=1}^{e} w_i(m) - \sum_{i=1}^{m-1} r_i(s) = 0 \quad (14)$$

$$w_i(m) - D_i(m) + A_i(m) = 0 \quad (15)$$

$$r_i(s) - A_i(m+1) + D_i(m) = 0 \quad (16)$$

$r_i(s) \geq \pi_i(s)$, i =1, 2… N, s=1, 2 …N-1 (17)

$r_i(s) \leq X_i(s)$, i =1, 2… N, s=1, 2 …N-1. (18)

$-A_i(m+1) + D_i(m) + \pi_i(s) = 0$, for $i \in$ I. (19)

$w_i(m) \geq \mu_i(m)$, for $i \in$ I. (20)

$$-A_i(m) + A_{i'}(m) + \alpha_m = 0 \quad \oplus \quad -A_{i'}(m) + A_i(m) + \alpha_m = 0 \text{ For } i \in I,\ i \neq i' \quad (21)$$

$$-D_i(m+1) + A_{i'}(m+1) + \alpha_m = 0 \quad \oplus \quad -D_{i'}(m) + A_i(m) + \alpha_m = 0 \text{ For} i \in I,\ i' \in Io, i \in Ii \quad (22)$$

$$(\ D_i(m) - D_{i'}(m) = \beta_s \ \wedge A_i(m+1) - A_{i'}(m+1) = \beta_s\ ) \oplus$$

$$(\ D_{i'}(m) - D_i(m) = \beta_s \ \wedge A_{i'}(m+1) - A_i(m+1) = \beta_s\ ) \quad i, i' \in Io \quad (23)$$

$$(\ A_i(m) - A_{i'}(m) = \beta_s \ \wedge D_i(m+1) - D_{i'}(m+1) = \beta_s\ ) \oplus$$

$$(\ A_{i'}(m) - A_i(m) = \beta_s \ \wedge D_{i'}(m+1) - D_i(m+1) = \beta_s\ ) \quad i, i' \in Ii\ . \quad (24)$$

$$C(m) \geq w(m) \quad (25)$$

A thorough look at the mathematical model reveals that there are matrices of groups of five decision variables that need to be optimized. Four of the variables build the timetable and the fifth one, will assure optimality. These groups of variables are arrivals, departures, running times, dwelling times, and delay of trains at the destination.

4.5.2. Optimization procedure

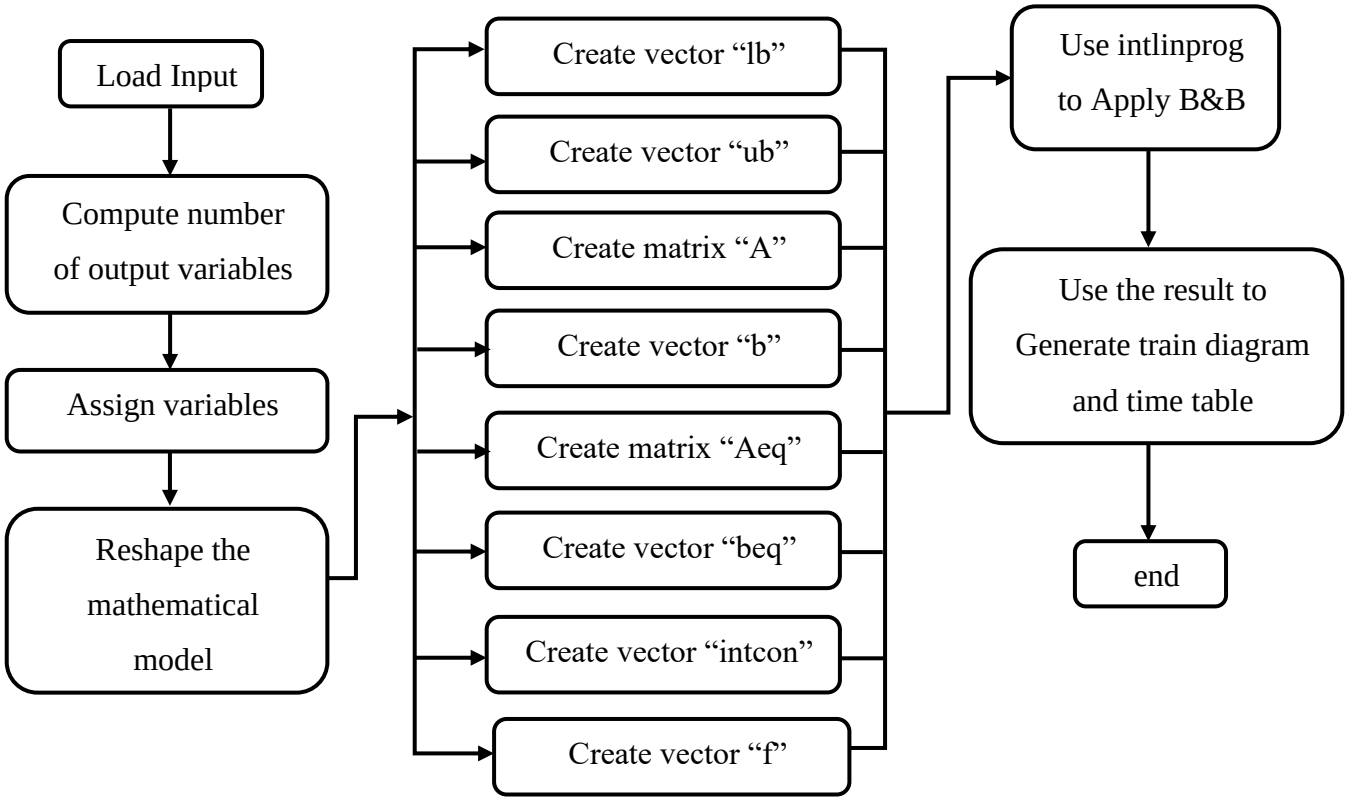

Figure 8: General Optimizing procedure for DSS

Figure 8 demonstrates the overall optimization procedure. Firstly, the model will gather the necessary inputs, most importantly, it takes the initial train schedule. Based on the schedule information the algorithm calculates decision variables (this will be discussed in the coming section on size determination). Again, depending on the input information, the mathematical model is updated. At this stage, the size of the matrices and the vector is already determined. Next, values for appropriately sized matrices and vectors are assigned. Now we have the number of equations and unknowns written in the form of matrices and vectors (A, Aeq, b, beq). Using those matrices and vectors, the intlinprog solver is recalled to calculate the value of the output variable and finally, the train diagram and timetable is constructed from the result of the algorithm.

i. Choosing decision variables.

After correctly loading the input data (the number of trains and meet points) the necessary parameters will be calculated. Here is the calculation of the number of arrivals, departures, running times, dwelling times and delays from input number of trains and meeting points.

```
% define variable parameters, number of trains and meet points are known.
Delays=trains;
Arrivals = trains*meetpoints;
Departures = trains*meetpoints;
RunningTimes = trains*segments;
DwellTimes = trains*meetpoints;


NofVar=trains+3*(trains*meetpoints) + (trains*segments); % generalized from the above.
```

For example, if there are six trains and six meet points, the space allotted for each of the individual variables is given below.

```
Total no. of arrivals =36
Total no. of departures =36
Total no. of running times =30
Total no. of dwell times =36
Delays = 6
```

The general formula for calculating the total number of decision variables is as follows.

```
%Compute total number of variables

NofVar=trains+3*(trains*meetpoints) + (trains*segments);
```

For the example above

```
NofVar=6+3*(6*6) + (6*5);
```

The number of variables will be one hundred forty-four (144). Each of these variables should be chosen carefully to get an optimal solution.

ii. Space allocation

After calculation of the number of variables and parameters, appropriate column matrix is pre-allocated to each parameter from 1 to NofVar as follows

```
% for NofVar columns allocate the place of each variables.
v1= 1: trains;
v2= (v1 (end) +1) :( v1 (end) +Arrivals);
v3= (v2 (end) +1) :( v2 (end) +Departures);
v4= (v3 (end) +1) :( v3 (end) +RunningTimes);
v5= (v4 (end) +1) :( v4 (end) +DwellTimes);
```

As presented in the optimization guideline of DSS, the next task is the creation of vectors and matrices. For all variables, column matrices called lower bound and upper bound, with a size of the number of variables (NofVar), is pre-allocated (0's for lower bound and ∞'s for upper bound). Each constraint having bounds will be assigned with their value according to the location of the variable.

iii. Constraint categorization

On the definition of MILP, matrix A is reserved for coefficients of inequality constraints and matrix Aeq is reserved for equality constraints. Therefore, the crucial step forward is to separately categorize inequality constraints and equality constraints as well as lower and upper bounds.

Upper bound

$$r_i(s) \leq X_i(s) \text{ , } i = 1, 2\ldots \text{ N, s=1, 2 } \ldots \text{N-1.} \tag{18}$$

Lower bounds

$$a_i(m_e) > s_i(m_e) \tag{12}$$

$$d_i \geq 0 \tag{13}$$

$$r_i(s) \geq \pi_i(s) \text{ , } i = 1, 2\ldots \text{ N, s=1, 2 } \ldots \text{N-1} \tag{17}$$

$$w_i(m) \geq \mu_i(m) \text{ , for } i \in \text{I.} \tag{20}$$

$$C(m) \geq w(m) \tag{25}$$

Equality constraints

$$d_i - a_i(m_e) + s_i(m_e) = 0 \tag{11}$$

$$a_i(m_e) - \sum_{m=1}^{e} w_i(m) - \sum_{i=1}^{m-1} r_i(s) = 0 \tag{14}$$

$$w_i(m) - D_i(m) + A_i(m) = 0 \tag{15}$$

$$r_i(s) - A_i(m+1) + D_i(m) = 0 \tag{16}$$

Inequality constraints

$$-A_i(m+1) + D_i(m) + \pi_i(s) \leq 0 \text{ , for } i \in \text{I.} \tag{19}$$

$$-A_i(m) + A_{i'}(m) + \alpha_m \leq 0 \;\oplus\; -A_{i'}(m) + A_i(m) + \alpha_m \leq 0 \text{ For } i \in I,\ i \neq i' \tag{21}$$

$$-D_i(m+1) + A_{i'}(m+1) + \alpha_m \leq 0 \;\oplus\; -D_{i'}(m) + A_i(m) + \alpha_m \leq 0 \text{ For} i \in I,\ i' \in Io, i \in Ii \tag{22}$$

$$\begin{aligned} &(D_i(m) - D_{i'}(m) \leq \beta_s \wedge A_i(m+1) - A_{i'}(m+1) \leq \beta_s) \oplus \\ &(D_{i'}(m) - D_i(m) \leq \beta_s \wedge A_{i'}(m+1) - A_i(m+1) \leq \beta_s) \quad i, i' \in Io \end{aligned} \tag{23}$$

$$\begin{aligned} &(A_i(m) - A_{i'}(m) \leq \beta_s \wedge D_i(m+1) - D_{i'}(m+1) \leq \beta_s) \oplus \\ &(A_{i'}(m) - A_i(m) \leq \beta_s \wedge D_{i'}(m+1) - D_i(m+1) \leq \beta_s) \quad i, i' \in Ii \text{ .} \end{aligned} \tag{24}$$

iv. Size determination

Since matrices A and Aeq are their respective number of constraints by the total number of variables sized matrices, their size is determined by the number of variables which is also dependent on the input schedule as the formula presented on the previous section of this thesis and their respective constraints.

Their respective rows should be calculated as follows for the placement of their constraints.

```
%calculate number of rows for equality constraints

EqRow= trains*(3+meetpoints+segments);

%create Matrix 'Aeq' which denotes coefficients of variables in equality constraints

Aeq=zeros (EqRow, NofVar);

%Create Column Vector 'beq' which denotes the right hand side of equality constraints

Beq=zeros (EqRow, 1);
```

For example, timetable having six trains and six meet points, the size of matrix Aeq is 84*144. The size varies exponentially with a variation in input. Thus, it is difficult for a human brain to consider all constraints listed earlier manually and to obtain an optimal solution.

The vectors b and beq have the same numbers of rows as A and Aeq respectively because they contain the right-hand side of the constraints. The vector f has the same size as the number of variables. Each decision variables will divide and hold the space allotted for f according to their size. The intcon is a row vector containing the address of each integer constraints.

v. Placing constraints

After correctly creating all vectors, it is necessary to place constraints as per their place and order to get the correct output from the algorithm. The categorization of constraints generated earlier will facilitate the placement job. Refer to appendix D section B.

CHAPTER FIVE

RESULT AND ANALYSIS

5.1. Result

To evaluate the efficiency of the model, in optimally solving conflicts, sample problems were tested on sample single-track rail network and optimal solutions were found. The tool developed under this thesis is not supposed to replace the train dispatcher but help them in making an optimal decision. As shown in Figure 9, the proposed solution is generated and given to the dispatcher then the dispatchers have all the necessary information including the optimal solution, rules and regulation and other situation on the ground to be able to solve real-time dispatching problems.

The sample problem contains a different number of stations and trains. For movement of trains from source to destination, the sample railway network considers only a single track which is more complicated and difficult for solving dispatching problems as well as a widely available type of track formation in regional networks than a double track [1]. During the resolution, the headway time considered between trains is 20 minutes, while the interval between departure and arrival is 5 minutes. Sample problems are given in Figure 8 and Figure 9 and their respective optimal solution is discussed here under.

Problem 1: Input Schedule Sheet

Table 8: Input Schedule for problem 1

Trains No.	Dire Dawa		Chenele		Harraouah	
	Arrival	Departure	Arrival	Departure	Arrival	Departure
Eastbound trains						
10	0	10	15	15	22	1000
11	0	15	30	50	65	1000
12	0	30	36	42	48	1000
Westbound trains						
406	42	1000	36	37	0	32
408	65	1000	52	58	0	46
422	70	1000	10	50	0	5

The developed interactive model has an option to generate a train diagram depicted in readily distinct form by dispatchers more specifically those of cross and overtake conflicts as shown in Figure 9. Other conflict types are not easily traceable by the dispatcher from the train diagram. Thus, the dispatcher needs to get them from the conflict report section of the dispatching support system. After properly loading the input schedule, the preview option of the graphical user interface (GUI) generates the following interactive window shown in Figure 9.

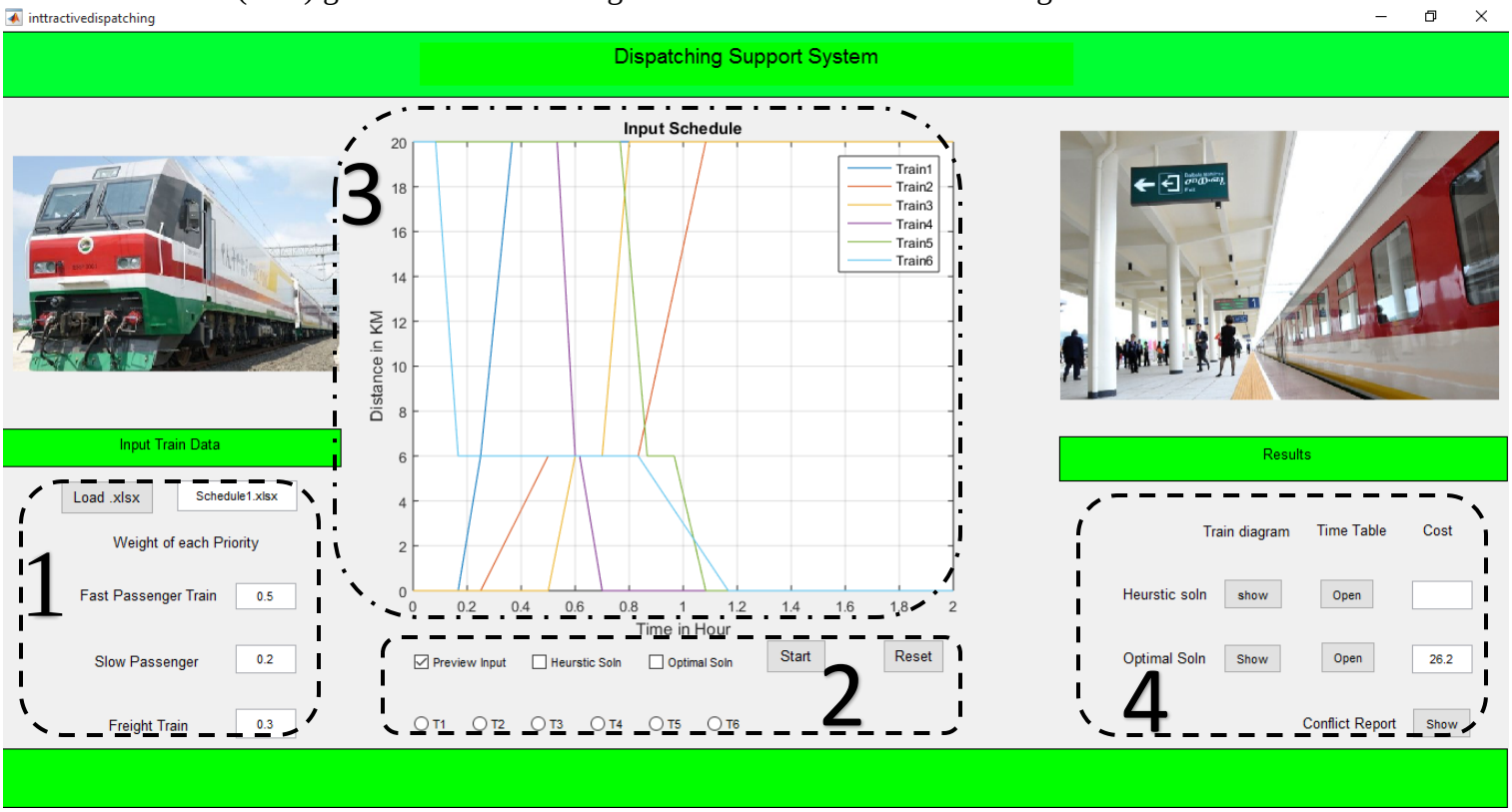

Figure 9: Dispatching support system preview

The GUI is comprised of four sub-sections:

1. This area is called the input section. The user is allowed to choose the input file when pressing the button and is a place where the dispatcher specifies the weight of each train type.
2. This section is a place where the dispatcher can preview the input or look at the optimal resolution by selecting the desired radio button and pressing start. There are also few other options here like reset and drawing the diagram for a specific train.
3. Here a preview of the input or the plan of the optimal resolution can be seen.
4. Result panel, the dispatcher gets the conflict report and the timetable in a separate window.

Initial Train diagram (Schedule):

The dispatcher has an option to see the input data from excel both on the GUI and out of the GUI window. This enables train dispatchers to verify if the input data is correct. Figure 10 shows the train diagram separate from the GUI.

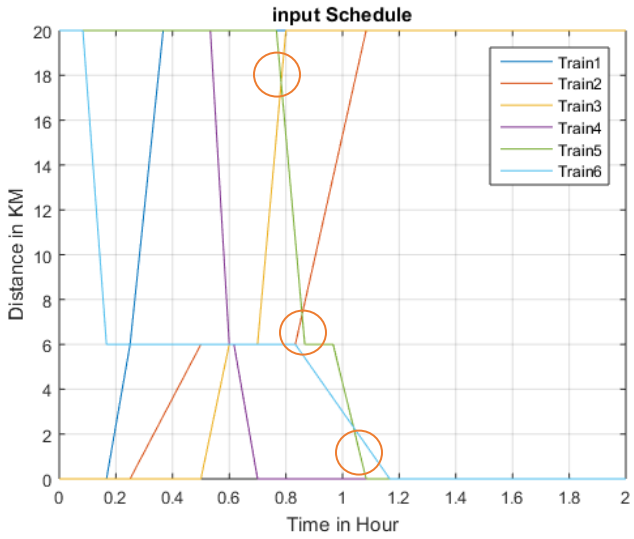

Figure 10: Problem 1 initial train diagram

For this train diagram, there are three stations (meet points) located at 0, 6, and 20 kilometers. If two trains meet other than this station, a conflict has had occurred. As clearly seen from the train diagram plot Figure 10, for the initial schedule, there are two meet conflicts and one pass conflict between trains mentioned in the conflict report below. The conflict report contains other forms of conflict as well.

Conflict report

The conflict report as presented below displays both initial conflicts and those conflicts raised during the resolution process by itself. Finally, the program automatically saves this report into the folder provided as a text file.

Pass conflict b/n trains 6 and 5 at Track segment1
Meet conflict b/n trains 3 and 5 at Track segment2
Meet conflict b/n trains 5 and 2 at Track segment2
Minimum headway not respected b/n trains 3 and 4 at station 2
Arrival and Departure interval not respected b/n train 1 and 6 at station 2
Minimum headway not respected b/n trains 2 and 6 at station 2
Arrival and Departure interval not respected b/n train 6 and 4 at station 3

Published with MATLAB® R2015a

Optimal solution:

Selecting the radio button for the option of optimal resolution generates the following train diagram in the interactive window with its corresponding cost.

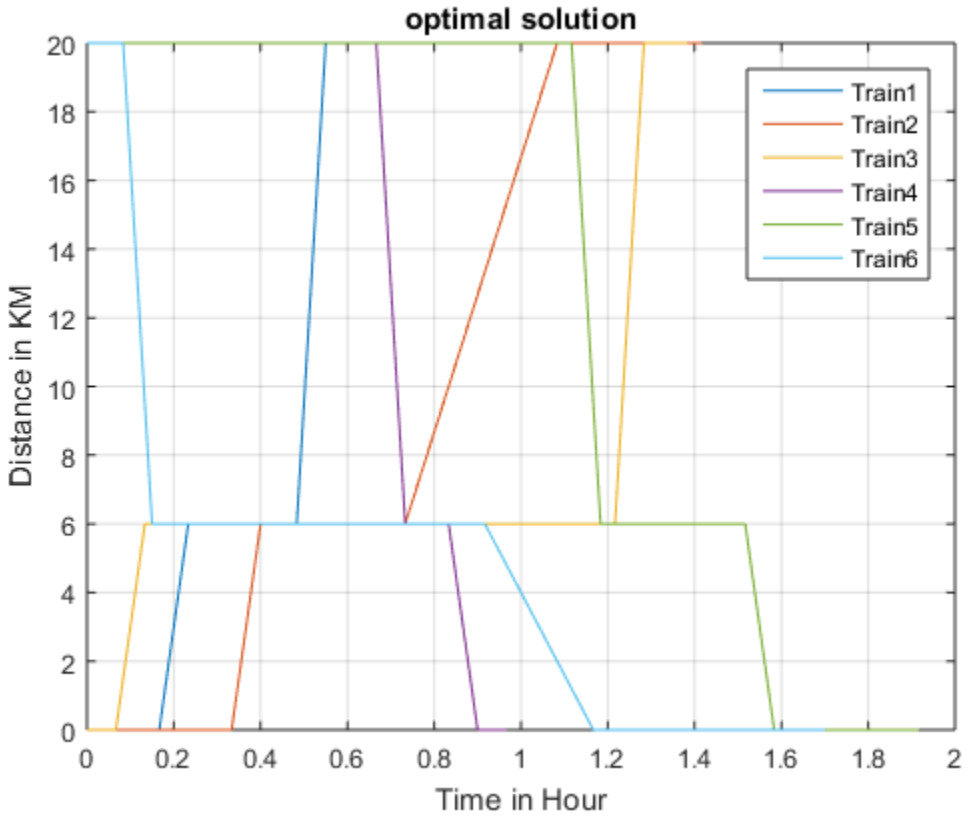

Figure 11 : Problem 1 optimal solution

In Figure 11, there are six trains running through three stations. Three of the trains are traveling eastbound while the others are traveling in westbound. The train diagram plot for the optimal solution as clearly seen from Figure 11, trains are only meeting on stations (0,6, and 9 kilometers). Thus, there are no cross, overtake or other forms of conflicts. This shows that the dispatching support system has solved the conflict neatly and optimally.

Problem 2: Initial Schedule Sheet

Table 9: Input Schedule Sheet for problem 2

M1		M2		M3		M4		M5		M6	
ARRIV	DEP	ARRIV	DEP	ARRIV	DEP	ARRIV	DEP	ARRIV	DEP	ARRIV	DEP
0	5	25	30	61	65	80	82	130	170	195	215
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	60	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

Train diagram of Initial schedule:

From the generated input train diagram in Figure 12, some of the conflicts that can be easily detected by looking at the picture are circled for an explanation. However, station conflicts and other conflicts due to safety constraints cannot be seen directly from the diagram. The conflict report demonstrates all conflicts in detail.

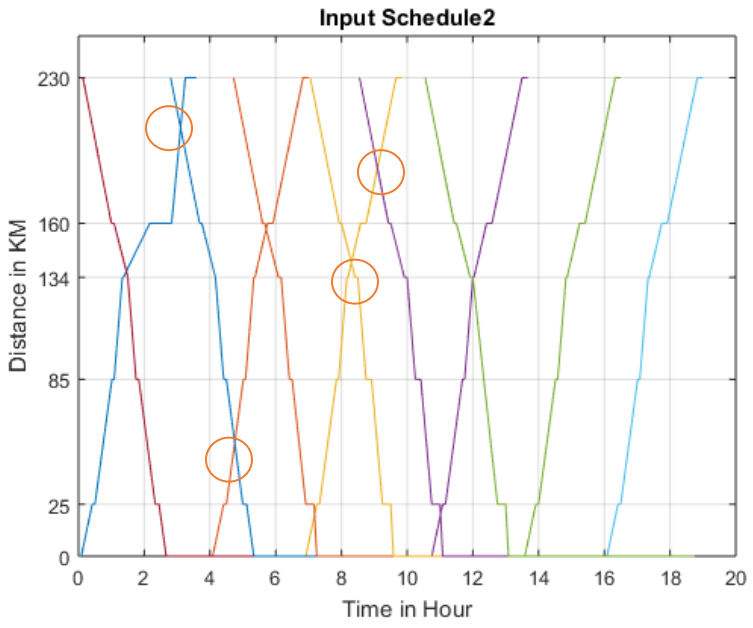

Figure 12: Problem 2 initial schedule

As in Figure 12, the train diagram shows that there is one cross(meeting) conflict on track segment two between trains two and eight, on track segment four cross conflict between trains three and ten, two cross conflicts on track segment five between trains one and eight and between trains three and eleven. It should be noted that these are not the only conflicts in the timetable

Optimal solution

In Figure 13 below, a train diagram for the optimal solution is sketched. The resolution of the conflicts depicted by circles on the input train diagram and hidden conflicts was successful. To achieve this the developed solution algorithm and all the constraints presented in this thesis have been strictly followed.

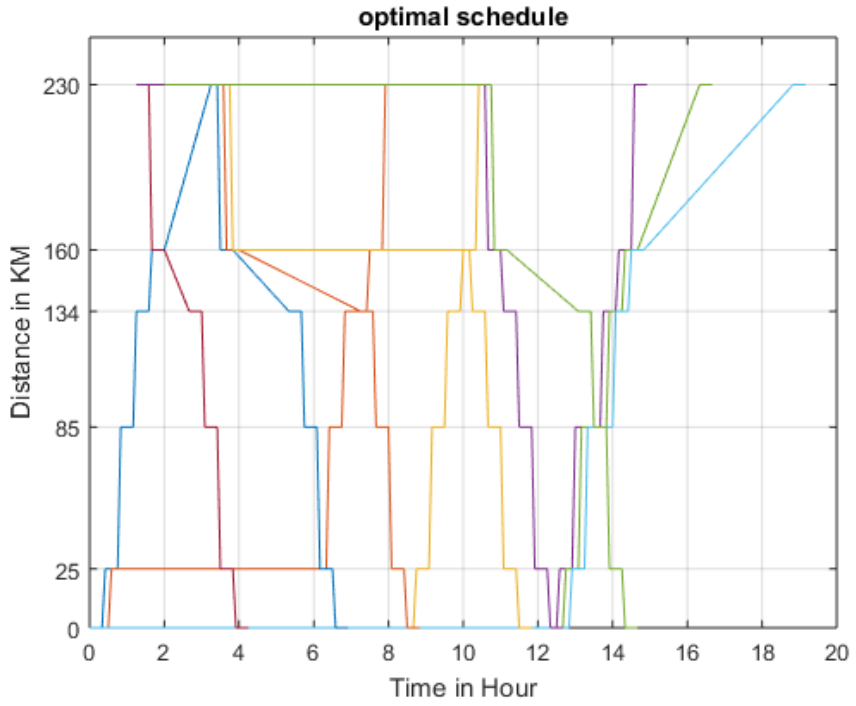

Figure 13: Problem 2 optimal solution

For this particular problem, there are six meet points and twelve trains. Six of the trains are traveling in the eastbound direction, and the others are traveling in the opposite direction in a single-track railway. The stations are located at 0,25,85,134,160, and 230. On the input train diagram plot on Figure 12 there were several conflicts between this station, but here in Figure 13, there are no conflicting movements in between beforementioned locations. Other station conflicts are also resolved and can be seen in the text conflict report.

5.2. Analysis

A schedule with the form of a real-world rail network was used to test and evaluate the developed application. The sample rail network problem considered both complex and manageable schedule to see the effect of variation in the number of train and station on the result of the developed model.

The following test is carried out to see the effect of the weight of the priority of trains on the optimal solution according to the following assumptions.

Assumptions 1:

$$priority\ 1\ >\ priority\ 2\ > priority\ 3$$

Assumption 2:

$$\%\ (priority\ 1) +\ \%\ (priority\ 2) + \%\ (priority\ 3) = 100\%$$

In table 10. We assigned each train type (in this thesis three types of trains are considered fast passenger, slow passenger, and freight train) with respective priorities as given in the table.

Table 10: Priority of each train type

Train type	Weight
fast passenger train	Priority 1
slow passenger train	priority 2
freight train	Priority 3

As shown in Table 11, for each priority separate weight have been given to test the impact of priority on the optimal solution. The dispatcher has an option to set the weigh at the GUI.

Table 11: Set of weights considered

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

Table 12 hereunder shows the test result from the developed model based on the assumption presented above.

Table 12: Variation in optimal solution cost

Input schedule	Set of weights for each priority	Optimal solution cost
1	Set 1	41
	Set 2	49.2
	Set 3	57.4
2	Set 1	85.5
	Set 2	96
	Set 3	95.5

Based on these results it can be said that, for most of the cases, the more evenly distributed set of priorities the less cost the optimal solution will be. However, this is not the case sometimes as for set 2 and set 3 in schedule 2 showed different result from this conclusion. Therefore, it will be up to the train dispatcher to decide which set of weights best fit the rules and regulation of the railway company as well as the reality on the ground. In addition, as expected the complexity of the timetable makes the program to take longer to reach an optimal solution.

CHAPTER SIX

CONCLUSION AND RECOMMENDATION

6.1. Conclusion

A depth study on train dispatching problems has been done by formulating the train dispatching problem as a mixed integer linear program. The corresponding optimization criteria (objective function) takes priority of train and total tardiness into consideration. The formulation consists of a carefully chosen practical rail network constraints. The method called “intlinprog” was used to solve the problem and the overall interactive dispatching support system was developed on MATLAB R2015b. The algorithm produced an optimal solution for all sample data from the tests carried out on the different sample data. The result was finally presented using an interactive GUI that accepts all necessary inputs from the dispatcher and process using the developed algorithm and then return the outputs as train diagram, timetable, and conflict report.

The aims and objectives of this work, which was to obtain an optimal resolution for train dispatching problem and to develop intelligent dispatching, was achieved as it helps the dispatchers in making decisions in solving dispatching problems they face by finding optimal solution. Finally, independent of the complexity of the problem the algorithm was able to reach an optimal solution.

6.2. Recommendations

This research focused on the development of intelligent dispatching support system but only for train dispatchers. However, more work on the same problem can be done for railway planners as well. Inevitably, it requires depth study of the impact of infrastructure change on the train schedule. (i.e., by increasing and decreasing number of stations, distance b/n stations, number of trains, and so on.) The data used for the development of the model are based on hypothetical assumptions, considering real-world situations. The results achieved were also satisfactory based on the tests carried on its performance. However, by using pure actual inputs on a specific railway network, the results found so far can be improved to a better one. "The science of better is not a one-time job." There is also a room for improvement on the formulation of the mathematical model. The objective functions can be modified to suit the situation on the ground based on additional and practical tests.

To conclude, there are lots of available space for improvement on the train dispatching problems. Therefore, improving practical constraints and coming with optimal resolution persists to be a complex problem.

REFERENCES

[1] S. A. Afeez, “Train Dispatching: Heuristic Optimization,” Dalarna University, Sweden, 2006.

[2] P. Pellegrini, G. Marlière, and J. Rodriguez, “A detailed analysis of the actual impact of real-time railway traffic management optimization,” *J. Rail Transp. Plan. Manag.*, vol. 6, no. 1, pp. 13–31, 2016.

[3] B. Adenso-Díaz, M. Oliva González, and P. González-Torre, “On-line timetable re-scheduling in regional train services,” *Transp. Res. Part B Methodol.*, vol. 33, no. 6, pp. 387–398, 1999.

[4] S. Ismail, “Railway traffic control and train scheduling based on inter-train conflict management,” *Transp. Res. Part B*, vol. 33, pp. 511–534, 1999.

[5] A. D. Ariano, D. Pacciarelli, and M. Pranzo, “Discrete Optimization A branch and bound algorithm for scheduling trains in a railway network,” vol. 183, pp. 643–657, 2007.

[6] V. Cacchiani and P. Toth, “Robust Train Timetabling,” *Int. Ser. Oper. Res. Manag. Sci.*, vol. 268, no. 3, pp. 93–115, 2018.

[7] G. Caimi, L. Kroon, and C. Liebchen, “Models for railway timetable optimization: Applicability and applications in practice,” *J. Rail Transp. Plan. Manag.*, vol. 6, no. 4, pp. 285–312, 2017.

[8] “Single Line Operation | The Railway Technical Website | PRC Rail Consulting Ltd.” [Online]. Available: http://www.railway-technical.com/signalling/single-line-operation.html. [Accessed: 23-Oct-2018].

[9] C. B. C. Güvenç Şahin, Ravindra K. Ahuja, “New Approaches for the Train Dispatching Problem,” *Transp. Res. B*, pp. 1–14, 2005.

[10] P. A. Afonso and C. F. Bispo, “Railway Traffic Management : Meet and Pass Problem,” *J. Syst. Manag. Sci.*, vol. 1, no. 1, pp. 1–26, 2011.

[11] A. L. M.A. Salido, M. Abril, F. Barber, L. Ingolotti, P. Tormos, “Topological Constraints in Periodic Train Scheduling,” in *Planning, Scheduling and Constraint Satisfaction: From*

Theory to Practice, 2005, pp. 1–4.

[12] M. J. C. M. Vromans, R. Dekker, and L. G. Kroon, "Reliability and heterogeneity of railway services," *ERASMUS Res. Inst. Manag. Rep. Ser.*, pp. 1–4, 2003.

[13] C. Mathworks, *Optimization Toolbox User's Guide*, 2017a ed. 2017.

[14] T. Cokyasar, "A Mixed Integer Linear Programming Approach for Developing Salary Administration," University of Tennessee, 2016.

[15] C. Mészáros and U. H. Suhl, *Advanced preprocessing techniques for linear and quadratic programming*, vol. 25, no. 4. Springer-Verlag, 2003.

[16] G. Cornuéjols, *Valid inequalities for mixed integer linear programs*, vol. 112, no. 1. Springer-Verlag, 2008.

[17] E. Danna, E. Rothberg, and C. Le Pape, *Exploring relaxation induced neighborhoods to improve MIP solutions*, vol. 102, no. 1. Springer-Verlag, 2005.

[18] "Signalling | The Railway Technical Website | PRC Rail Consulting Ltd." [Online]. Available: http://www.railway-technical.com/signalling/.

[19] F. Corman, A. D. Ariano, A. D. Marra, D. Pacciarelli, and M. Samà, "Integrating train scheduling and delay management in real-time railway traffic control," *Transp. Res. Part E*, 2016.

[20] J. Harbering and A. Ranade, "Single Track Train Scheduling," *Plan. a Public Transp. Syst. with a View Towar. Passengers' Conv.*, p. 151, 2015.

[21] X. Zhou, "Single-track train timetabling with guaranteed optimality: Branch-and-bound algorithms with enhanced lower bounds," *Transp. Res. Part B*, vol. 41, pp. 320–341, 2007.

[22] Ethio Djibouti railway enterprise, "Train diagram." .

[23] J. Törnquist, "Computer-based decision support for railway traffic scheduling and dispatching : A review of models and algorithms," in *OASIcs-OpenAccess Series in Informatics,vol. 2.*, 2006.

[24] V. Cacchiani *et al.*, "An overview of recovery models and algorithms for real-time railway rescheduling," *Transp. Res. Part B Methodol.*, vol. 63, pp. 15–37, 2014.

[25] MLADENOVIĆ, Snežana, Mirjana ČANGALOVIĆ, "Heuristic approach to train rescheduling," *Yugosl. J. Oper. Res.*, vol. 17, no. 1, pp. 9–29, 2007.

[26] K. Mohamed, "Simulation of Train Dispatching Problem," Hogskolan Dalarna University, Sweden, 2004.

[27] Y. W. L. C.K. Chinu, C. MChou, J. HM. Lee, Leung, "A Constraint Based Interactive Train Rescheduling Tool," University of Hong Kong, Shatin, N. T, Hong Kong, 2002.

[28] C. George, "A Theory of Distributed Train Rescheduling," UNU/IIST (International Institute for Software Technology), 1996.

[29] F. Corman, A. D'Ariano, D. Pacciarelli, and M. Pranzo, "Optimal inter-area coordination of train rescheduling decisions," *Transp. Res. Part E Logist. Transp. Rev.*, vol. 48, no. 1, pp. 71–88, 2012.

[30] M. Dotoli, N. Epicoco, M. Falagario, A. Piconese, F. Sciancalepore, and B. Turchiano, "A real time traffic management model for regional railway networks under disturbances," in *IEEE International Conference on Automation Science and Engineering*, 2013.

[31] A. D. Ariano, M. Pranzo, and U. Siena, "Conflict resolution and train speed co-ordination for solving timetable perturbations," *Energy*, vol. 2005, no. June, pp. 1–21, 2005.

[32] "Gurobi Optimization - The State-of-the-Art Mathematical Programming Solver." [Online]. Available: http://www.gurobi.com/.

[33] "PTC Mathcad | PTC." [Online]. Available: https://www.ptc.com/en/products/mathcad.

APPENDICES

Appendix A: Inputs used

Problem 1:

"SCHEDULE" Sheet

		Dire Dawa		Chenele		Harraouah	
		Arrival	Departure	Arrival	Departure	Arrival	Departure
3	101	0	13	18	18	25	1003
4	102	0	18	33	53	68	1003
5	103	0	33	39	45	51	1003
6	204	45	1003	39	40	0	35
7	205	68	1003	55	61	0	49
8	206	73	1003	13	53	0	8

"TRAINS DATA" Sheet

1	Number of westbond trains	3
2	Number of eastbond trains	3

	TRAINS DESCRIPTION		
	ID	TYPE	LENGTH
5	101	1	1000
6	102	2	520
7	103	1	868
8	204	1	458
9	205	1	1000
10	206	3	940

"TRACK DATA" Sheet

	STATIONS DESCRIPTION				SEGMENT DESCRIPTION
	NAME	CAP	LENGTH	DISTANC	NAME
3	E1	12	1000	0	T1
4	E2	3	1000	6	T2
5	E3	8	1000	20	

Problem 2:

"SCHEDULE" Sheet

	A	B	C	D	E	F	G	H	I	J	K	L
1	M1		M2		M3		M4		M5		M6	
2	ARR	DEP	ARR	DEP	ARR	DEP	ARR	DEP	ARR	DEP	ARR	DEP
3	0	8	28	33	64	68	83	85	133	173	198	218
4	243	248	268	273	304	308	323	325	348	358	413	423
5	413	418	438	443	474	478	493	495	518	528	583	593
6	643	648	668	673	704	708	723	725	748	758	813	823
7	813	818	838	843	874	878	893	895	918	928	983	993
8	963	968	988	993	1024	1028	1043	1045	1068	1078	1133	1143
9	163	503	143	150	108	113	93	93	63	68	11	11
10	323	663	303	310	268	273	253	253	224	228	171	171
11	438	778	418	433	388	393	368	373	339	343	286	286
12	578	918	558	573	528	538	508	513	479	483	426	426
13	668	1008	648	663	618	628	598	603	569	573	516	516
14	788	1128	768	783	743	743	718	723	689	693	636	636

"TRAINS DATA" Sheet

	A	B	C	D	E	F
1	Column1	Column2				
2	number of inbond trains	6				
3	number of outbond trains	6		Columr	Columr	Column3
4				TRAINS DESCRIPTION		
5				ID	TYPE	LENGTH
6				101	1	1000
7				102	2	520
8				103	1	868
9				104	1	541
10				105	1	1002
11				106	3	1546
12				201	2	124
13				202	1	652
14				203	2	658
15				204	1	458
16				205	1	1000
17				206	3	940

"TRACK DATA" Sheet

	A	B	C	D	E
1	STATIONS DESCRIPTION				SEGMENT DESCRIPTION
2	NAMI	CAP	LENGTH	DISTANC	Name
3	E1	12	1000	0	T1
4	E2	3	1000	25	T2
5	E3	8	1000	85	T3
6	E4	7	1000	134	T4
7	E5	4	1000	160	T5
8	E6	12	1000	230	

Appendix B: Test Result

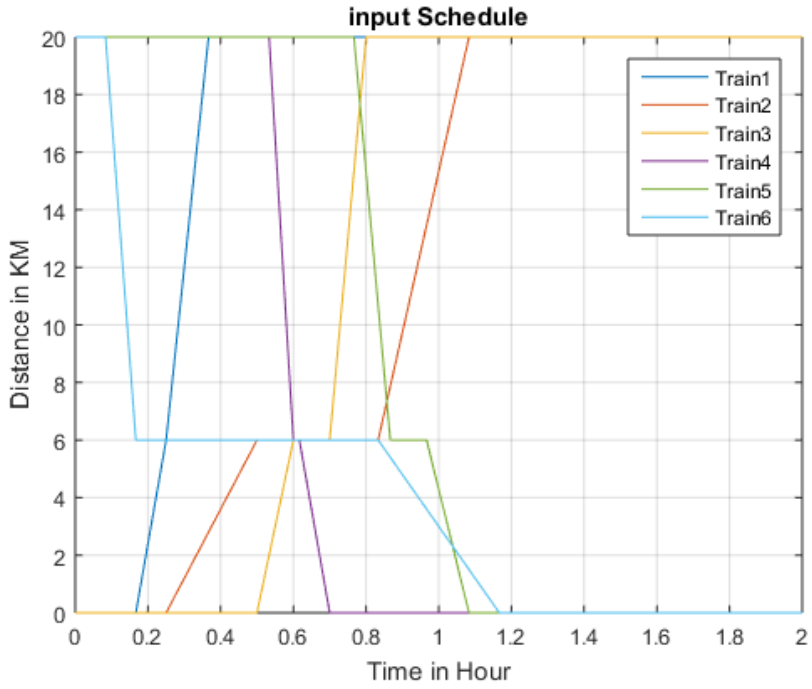

Figure 14: problem 1 input schedule

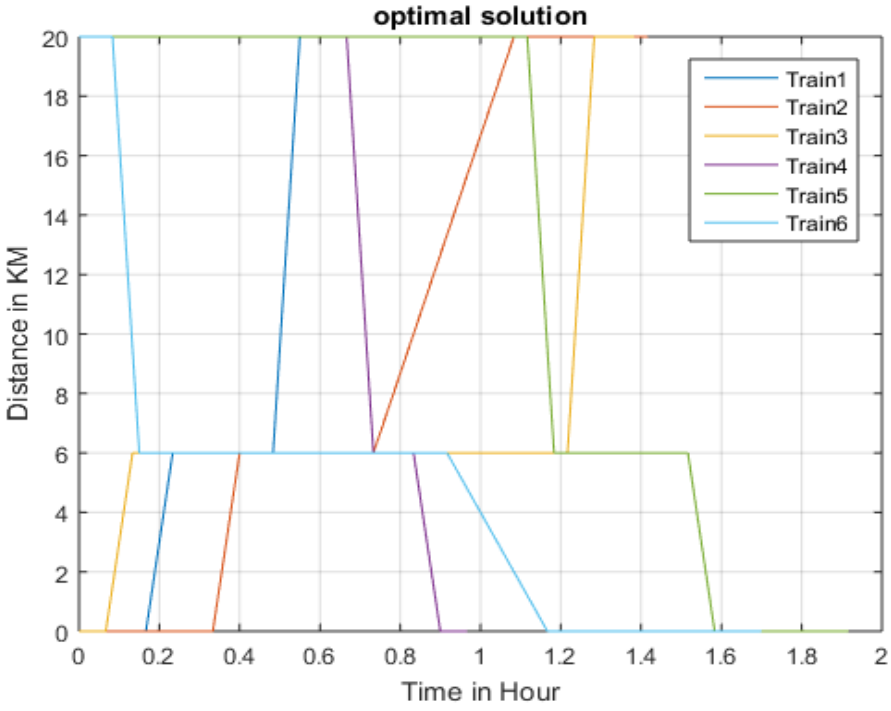

Figure 15: problem 1 optimal solution

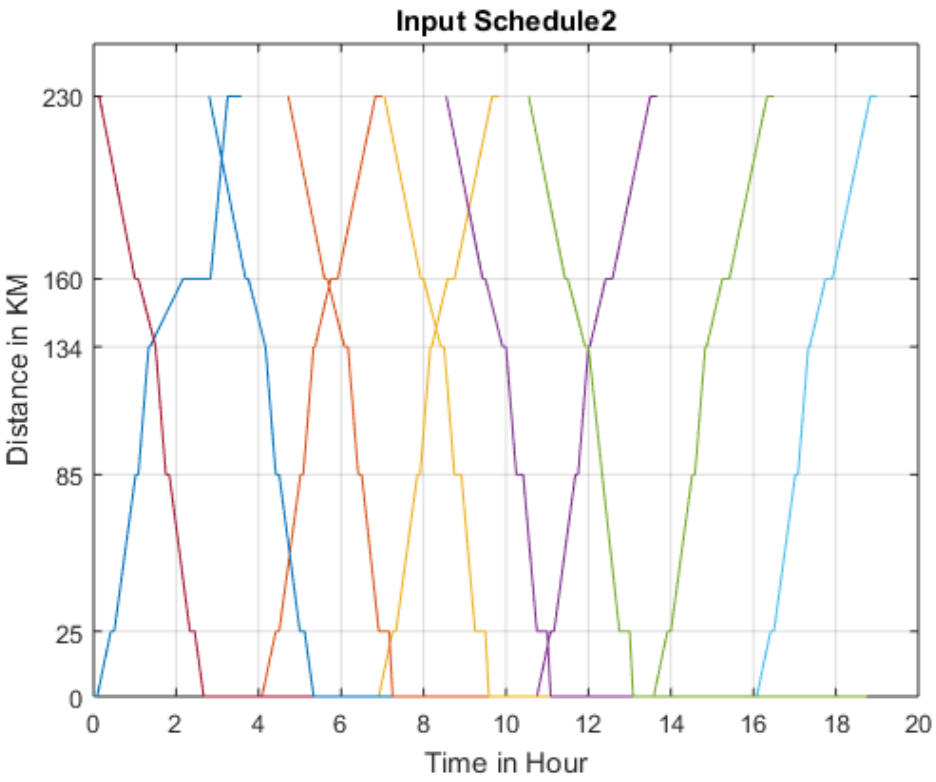

Figure 16: problem 2 input schedule

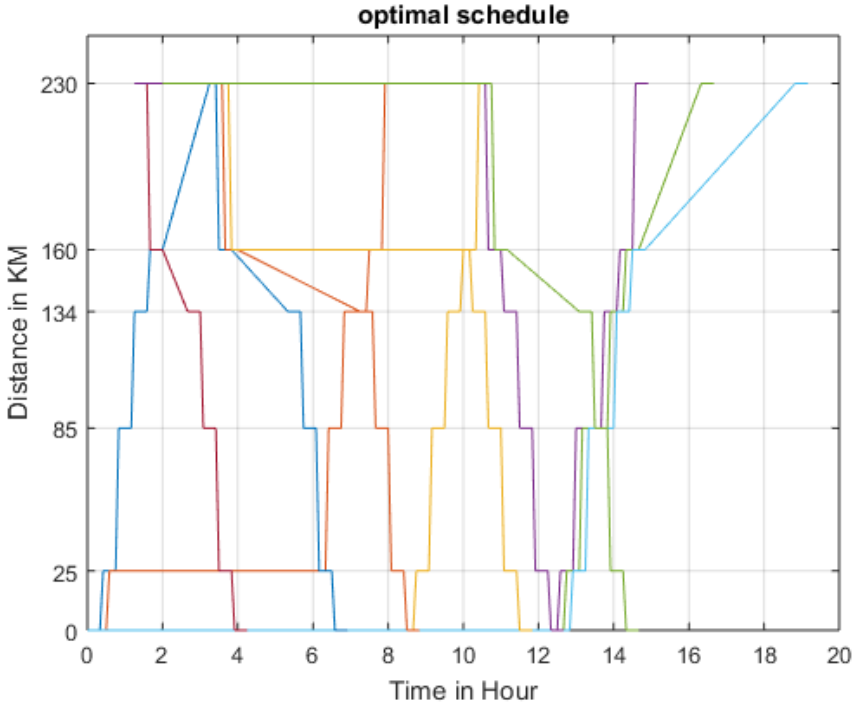

Figure 17: problem 2 optimal solution

Appendix C: Data Collected

(From Ethio-Djibouti Railway Enterprise, Dire Dawa)

17/01/1997

HORAIRES DES TRAINS

Trains No.	*Heures d'arrivée du trains à D. Daoua*	*Trains No.*	*Heures de sortie Locos du Dépot*	*Heures de départ du trains de la gare de D.Daoua*
406	3:27	9	5:10	6:00
408	5:17	11	6:40	7:30
422	7:20	31	7:56	8:46
22	8:45	37	8:40	9:30
424	14:30	319	13:41	14:30
10	13:46	321	15:10	16:00
12	17:24	21	18:40	19:30
		323	20:10	21:00

17	14:00	18	4:45	5:45
105	5:10	14	5:10	6:00
15	6:55	26	6:10	7:00
107	10:35	210	10:50	11:40
13	18:15	212	14:40	15:30
101	19:29	16	18:40	19:30
		218	19:40	20:30

CHEMIN DE FER DJIBOUTO-ETHIOPIEN

Figure 18: Trains schedule

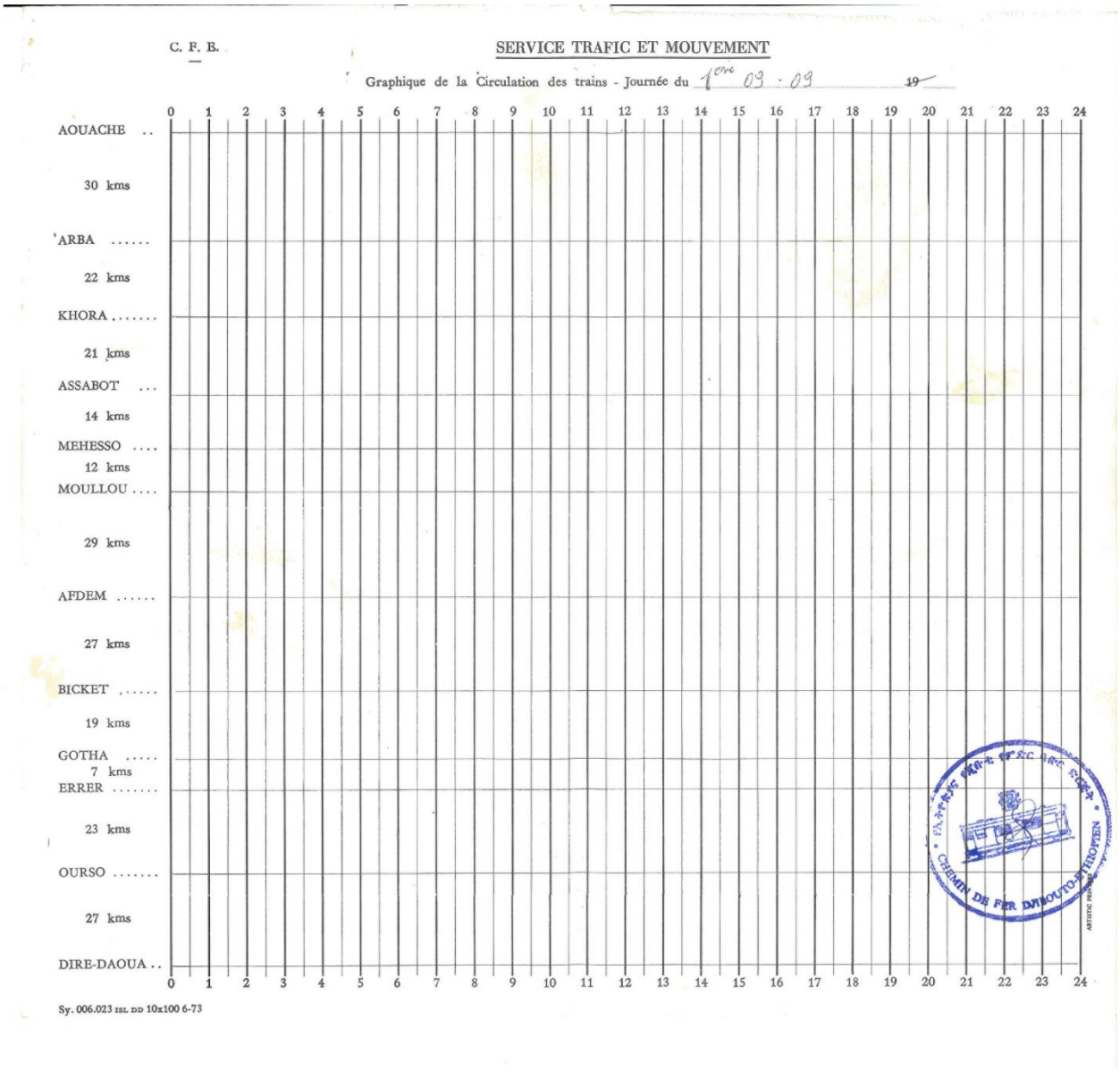

Figure 19: Time Distance diagram

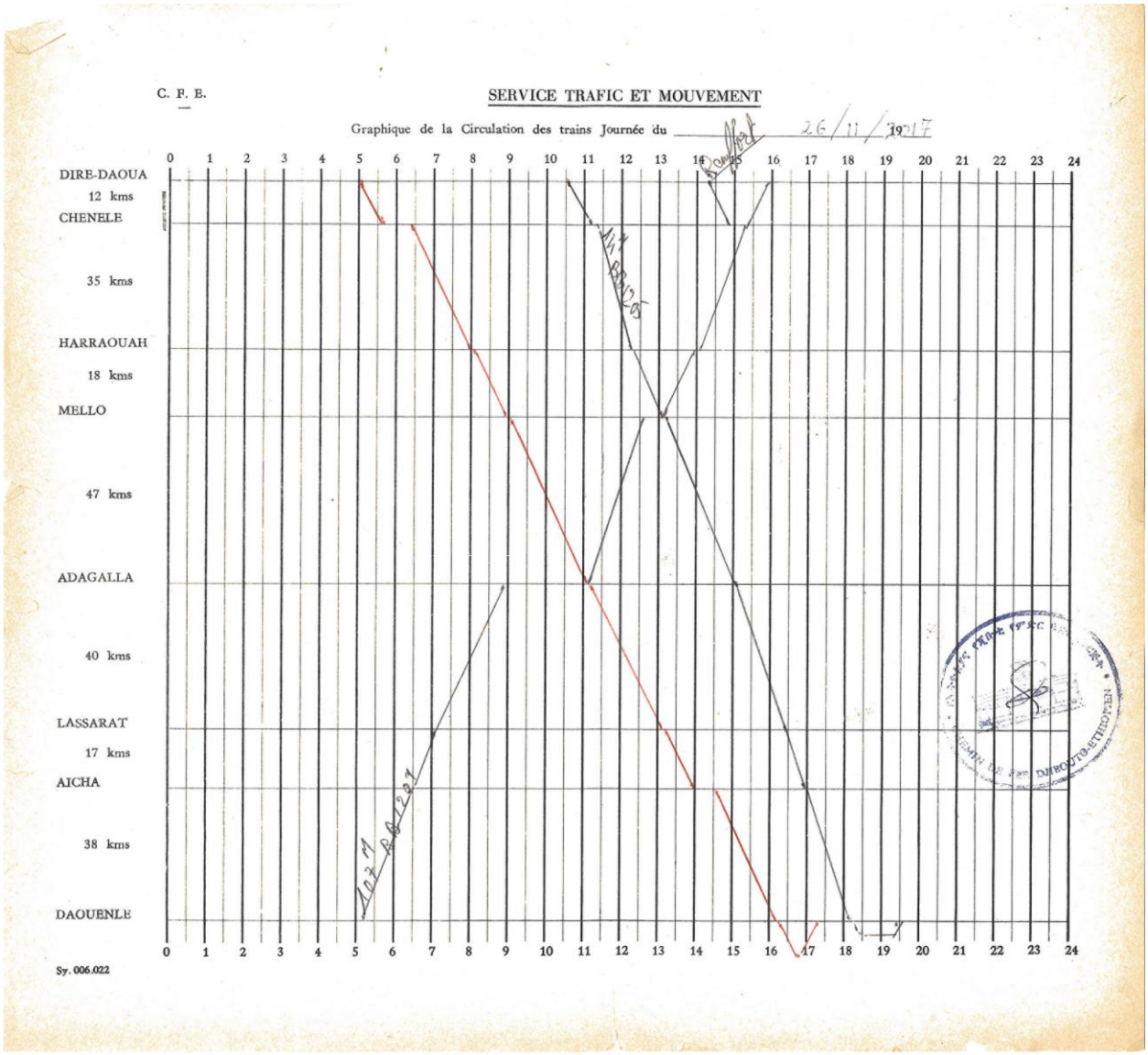

Figure 20: Train diagram of three trains working on 26-11-2017

Appendix D: MATLAB Source code

A. Conflict Detection Block

```
function [meet, pass, OtherConflicts] =ConflictDetection (TimeTable)
%====================
% detection starts here
% first scan stars here
%====================
[r,c]=size(TimeTable);
meetpoints = c/2;
trains=r;
tr=r/2;
segments = meetpoints-1;
outbound_trains = tr;
%segment start time
SegStart=zeros(r,segments);
for i=1:r
    for j=1:segments
     if(i<=outbound_trains)
        SegStart(i,j)= TimeTable(i,2*j);
     else
        SegStart(i,j)= TimeTable(i,2*j+2);
     end
    end
end
%sort in intering order to segment
[SortedSeg,TrainIndeces]=sort(SegStart,1);
SegFinish=zeros(r,segments);
for i=1:r
    for j=1:segments
     if(i<=outbound_trains)
       SegFinish(i,j)= TimeTable(i,2*j+1);
     else
        SegFinish(i,j)= TimeTable(i,2*j-1);
     end
    end
end

finish1=zeros(r,segments);
for i=1:r
    for j=1:segments
finish1(i,j)=SegFinish(TrainIndeces(i,j),j);
    end
end

% make a data with train indeces, start, and finish time
data=zeros(r,3,segments);
```

```
for seg=1:segments
data(:,:,seg)=[TrainIndeces(:,seg) SortedSeg(:,seg) finish1(:,seg)];
end
C=cell(20,6); %preallocate conflict
meet_counter=0;
pass_counter=0;
conflict_counter=0;
meet =zeros(20,3);
pass =zeros(20,3);
k=1;
k1=1;
for seg=1:segments % for all segments
for j=1:trains-1
    % check if the two train are toward the same direction
    if ((data(j,1,seg)<=tr && data(j+1,1,seg)<=tr)|| (data(j,1,seg)>tr &&
data(j+1,1,seg)>tr))
        % check for over take conflict
      if((data(j,2,seg)<data(j+1,2,seg))&&(data(j,3,seg)<data(j+1,3,seg))||...
            (data(j,2,seg)>data(j+1,2,seg))&&(data(j,3,seg)>data(j+1,3,seg)))
         % guarantee train overtake happen only at meet point, without problems
         % do nothing
      else
         track_seg=seg;
         pass_counter=pass_counter+1;
         conflict_counter=conflict_counter+1;
         Apass=data(j,1,seg);
         Bpass=data(j+1,1,seg);
         k=pass_counter;
         pass(k,:)=[Apass,Bpass,track_seg];
         type_1 = 'pass conflict b/n trains ';
        location=' at Track segment';
        C(conflict_counter,:) = {'pass conflict b/n trains ',data(j,1,seg),...
            ' and ',data(j+1,1,seg),' at Track segment ',track_seg};
        X = [type_1,num2str(data(j,1,seg)),' and
',num2str(data(j+1,1,seg)),location,num2str(track_seg)];
        disp(X)
     end
    else % the trains are toward diffrent direction.
        % check for crossing conflict
      if (data(j,3,seg)<data(j+1,2,seg)||data(j,2,seg)>data(j+1,3,seg))
          % guarantee train crossing happen only at meet point, without problems
          % so do nothing
      else % there is crossing conflict
         track_seg=seg;
         meet_counter=meet_counter+1;
         conflict_counter=conflict_counter+1;
         Ameet=data(j,1,seg);
         Bmeet=data(j+1,1,seg);
         k1=meet_counter;
         meet(k1,:)=[Ameet,Bmeet,track_seg];
         type_1 = 'meet conflict b/n trains ';
```

```
         location=' at Track segment';
         C(conflict_counter,:) = {'meet conflict b/n trains ',data(j,1,seg),' and
',...
             data(j+1,1,seg),' at Track segment ',track_seg};
         X = [type_1,num2str(data(j,1,seg)),' and
',num2str(data(j+1,1,seg)),location,num2str(track_seg)];
         disp(X)
      end
    end
end
end
%end of first Detection Block
```

```
pass conflict b/n trains 6 and 5 at Track segment1

meet conflict b/n trains 3 and 5 at Track segment2

meet conflict b/n trains 5 and 2 at Track segment2
```

```
[sorted_station,index1]=sort(TimeTable,1);

for i=1:trains
   %     conflict at Station
   for me=1:meetpoints
       switch(i)
          case {2*me}
            for t=1:trains
               if(sorted_station(t,i)-sorted_station(t,i-1)<=5)
  type_3 = 'arrival and Departure interval not respected b/n train ';
  location4=' at station ';
  conflict_counter=conflict_counter+1;
  C(conflict_counter,:) = {'arrival and Departure interval not respected b/n
train',index1(t,i),...
      ' and ',index1(t,i-1),' at station ',i/2};
  y = [type_3,num2str(index1(t,i)),' and ',num2str(index1(t,i-
1)),location4,num2str(i/2)];
  disp(y)
                end
             end

       end
    end
%
% minimum headway
    for j=1:trains-1
         if (((sorted_station(j,i)~= 0 &&
sorted_station(j+1,i)~=0)&&(sorted_station(j,i)~=1000 &&...
                sorted_station(j+1,i)~=1000)))
             if(sorted_station(j+1,i)-sorted_station(j,i)<5)
             type_1 = 'minimum headway not respected b/n trains ';
```

```
             location3=' at station ';
        for mm=1:meetpoints
            switch(i)
             case {(2*mm)-1 2*mm}
                    s=mm;
            end
        end

               conflict_counter=conflict_counter+1;
               C(conflict_counter,:) = {'minimum headway not respected b/n trains
',index1(j,i),...
                    ' and ',index1(j+1,i),' at station ',s};

               X = [type_1,num2str(index1(j,i)),' and
',num2str(index1(j+1,i)),location3,num2str(s)];
             disp(X)
             end
         end
     end
end

%delete unused space in pre-allocated cell
meet(k1+1:end,:)=[];
pass(k+1:end,:)=[];
C(conflict_counter+1:end,:)=[];
%write conflicts found in a text file
        fileID = fopen('pushit.dat','w');
        formatSpec = '\r\n%s %d %s %d %s %d \r\n';
        [nrows,~] = size(C);
        fprintf(fileID,'Conflict Report:');
        for row = 1:nrows
         fprintf(fileID,formatSpec,C{row,:});
        end
        fclose(fileID);
        type pushit.dat
```

```
minimum headway not respected b/n trains 3 and 4 at station 2

arrival and Departure interval not respected b/n train 1 and 6 at station 2

minimum headway not respected b/n trains 2 and 6 at station 2

arrival and Departure interval not respected b/n train 6 and 4 at station 3
```

```
Conflict Report:

pass conflict b/n trains 6 and 5 at Track segment 1

meet conflict b/n trains 3 and 5 at Track segment 2

meet conflict b/n trains 5 and 2 at Track segment 2

minimum headway not respected b/n trains 3 and 4 at station 2

arrival and Departure interval not respected b/n train 1 and 6 at station 2

minimum headway not respected b/n trains 2 and 6 at station 2

arrival and Departure interval not respected b/n train 6 and 4 at station 3
```

Published with MATLAB® R2015b

B. Conflict Resolution Block

```
function [] = Milp_Algorithm(TimeTable,DISTANCE,priority)
% end of Detection Block
% Start Of Resolution Block
[r,c]=size(TimeTable);
meetpoints = c/2;
trains=r;
segments = meetpoints-1;
outbound_trains = TrainData(2);
inbound_trains = TrainData(1);

rik=zeros(r,segments);
for i=1:r
    for j=1:segments
        if(i<=outbound_trains)
  rik(i,j)=TimeTable(i,j+j+1)-TimeTable(i,j+j);
        else
  rik(i,j)=TimeTable(i,2*j-1)-TimeTable(i,2*j+2);
        end
    end
end

siudwell=zeros(r,meetpoints);
for i=1:meetpoints
 for j=1:r
  siudwell(j,i)=TimeTable(j,i+i)-TimeTable(j,i+i-1);
       if(siudwell(j,i)<=5)
         siudwell(j,i)=10;
       end
  end
end
```

```
%Compute total number of variables
NofVar=trains+2*(trains*meetpoints)+(trains*segments)+(trains*meetpoints);
%define varible parameters
Delays=trains;
Arrivals=trains*meetpoints;
Departures=trains*meetpoints;
RunningTimes=trains*segments;
DwellTimes=trains*meetpoints;

v1=1:trains;
v2=(v1(end)+1):(v1(end)+Arrivals);
arr_table=reshape(v2,trains,meetpoints);
v3=(v2(end)+1):(v2(end)+Departures);
dep_table=reshape(v3,trains,meetpoints);
v4=(v3(end)+1):(v3(end)+RunningTimes);
run_table=reshape(v4,trains,segments);
v5=(v4(end)+1):(v4(end)+DwellTimes);
dwell_table=reshape(v5,trains,meetpoints);

 so=TimeTable(1:outbound_trains,c-1);
 si=TimeTable(outbound_trains+1:end,1);
 sc=[so;si];
 %start to MILP here
 mrik= min(rik(:));
%  mrik=7;
 mdwell=20;
%  mdwell=min(siudwell(:));
lb=zeros(1,NofVar);
lb_in=v2(1)-1;
for i=1:outbound_trains
    lb_in=lb_in+meetpoints;
    lb(lb_in)=TimeTable(i,c-1);
end
lb_out=lb_in+1;
for i=1:inbound_trains
       lb(lb_out)=TimeTable(outbound_trains+i,1);
        lb_out=lb_out+meetpoints;
end
mmrik=reshape(rik,size(v4,2),1);
lb(v4)=mrik;
mmdwell=reshape(siudwell,size(v5,2),1);
lb(v5)=mdwell;
ub=inf(1,NofVar);

% equality constarints
% calculate
EqRow=(2*trains)+trains+(meetpoints*trains)+(segments*trains);
Aeq=zeros(EqRow,NofVar);
beq=zeros(EqRow,1);
beq(1:outbound_trains)=-1*TimeTable(1:outbound_trains,c-1);
beq(outbound_trains+1:r)=-1*TimeTable(outbound_trains+1:r,1);
```

```
% constraint 1 %di=aimi-simi
RF_c1=trains;
c2=v1(end);
for c1=1:RF_c1
    if(c1<=outbound_trains)
     c2=c2+meetpoints;
     Aeq(c1,[c1,c2])=[1,-1];
    else
    c2=c2+meetpoints;
    Aeq(c1,[c1,c2-2])=[1,-1];
    end
end

%constarint #2

RS_c2=r+1;
RF_c2=r+trains;
c2_c2=v1(end);
c2_c3=v4(end)-segments;
c2_c5=v3(end)-1;

sd=segments;
cdwell=zeros(1,sd);
crun=zeros(1,sd);
cc2=-1*ones(1,(2*sd)+1);
cc2(1,1)=1;
count1=0;
for con2=RS_c2:RF_c2
    count1=count1+1;
    c2_c2=c2_c2+meetpoints;
    c2_c3=c2_c3+meetpoints;
    a1=c2_c3;
    c2_c5=c2_c5+segments;
    b1=c2_c5;
         for seg=1:sd
        cdwell(1,seg)=a1;
        crun(1,seg)=b1;
        a1=a1+1;
        b1=b1+1;
         end
        if(count1<=outbound_trains)
    Aeq(con2,[c2_c2,cdwell,crun])=cc2;
        else
            Aeq(con2,[c2_c2-2,cdwell+1,crun])=cc2;
        end
end

%constraint #3

RS_c3=RF_c2+inbound_trains+1;
```

```
RF_c3=RS_c3+(r*meetpoints)-1;
c3_c2=v4(end);
c3_c3=v2(end);
c3_c4=v1(end);
for con3=RS_c3:RF_c3
    c3_c2=c3_c2+1;
    c3_c3=c3_c3+1;
    c3_c4=c3_c4+1;
    Aeq(con3,[c3_c2,c3_c3,c3_c4])=[1,-1,1];
end

%constraint #4

RS_c4=RF_c3+1;
RF_c4=RF_c3+(segments*trains);
c4_c2=v3(end);
c4_c3=v1(end)+1;
c4_c4=v2(end);
c4_c1=0;
c4_c=0;
for con4=RS_c4:RF_c4
c4_c1=c4_c1+1;
c4_c2=c4_c2+1;
c4_c=c4_c+1;
Aeq(con4,c4_c2)=1;
c4_c3=c4_c3+1;
c4_c4=c4_c4+1;
  if(c4_c==(RunningTimes/2)+1)
      c4_c1=0;
       c4_c4= c4_c4+2;
      Aeq(con4,[c4_c3,c4_c4])=[-1,1];
      c4_c1=c4_c1+1;
  else
    if(c4_c1<=segments)
     Aeq(con4,[c4_c3,c4_c4])=[-1,1];
    else
        c4_c1=0;
        c4_c4= c4_c4+1;
        c4_c3=c4_c3+1;
        Aeq(con4,[c4_c3,c4_c4])=[-1,1];
        c4_c1=c4_c1+1;
    end
   end
 end

%inequality constraints

dr=(trains)/2-1;
sd=(trains)/2-1;
for vv=1:dr
    sd=sd-1;
```

```
    dr=dr+sd;
end
%==============
if(pass(1,1)==0)
    rpass=0;
end
if(meet(1,1)==0)
    rmeet=0;
end

InEqRow=(r*meetpoints)+rmeet+(rpass*2)+(2*meetpoints*dr);
pp=(r*meetpoints);
pq=(r*meetpoints)+rmeet;
pw=(r*meetpoints)+rmeet+(rpass*2);

A=zeros(InEqRow,NofVar);
b=zeros(InEqRow,1);

if(pass(1,1)~=0)
    b(pq+1:pw)=10;
end
if(meet(1,1)~=0)
    b(pp+1:pq)=-10;
end
b(1:r*meetpoints)=-20;
b(pq+1:InEqRow)=-10;

%constraint #4

RF_c7=(r*meetpoints);
c7_c3=v2(end);
c7_c4=v1(end);
for con7=1:RF_c7
    c7_c3=c7_c3+1;
    c7_c4=c7_c4+1;
    A(con7,[c7_c3,c7_c4])=[-1,1];
end

arrival=v1(end);
departure=v2(end);
arrival_table=zeros(trains,meetpoints);
departure_table=zeros(trains,meetpoints);
for table_counter=1:trains
    for column=1:meetpoints
        arrival=arrival+1;
        departure=departure+1;
      arrival_table(table_counter,column)=arrival;
      departure_table(table_counter,column)=departure;
    end
end
 %meet constarint
```

```
if(meet(1,1)~=0)
 for mr=1:rmeet
    se=meet(mr,3);
    if(meet(mr,1)>meet(mr,2))%first train  is inbound
            A(pp+mr,[arrival_table((meet(mr,2)),se+1),...
                departure_table((meet(mr,1)),se+1)])=[1,-1];
    else % first train is outbound
            A(pp+mr,[arrival_table((meet(mr,2)),se),...
            departure_table((meet(mr,1)),se)])=[1,-1];
    end
 end
end
% % %pass constarint
rp=pp+rmeet;
rp2=pp+rmeet;
if(pass(1,1)~=0)
for pass_resoltion=1:rpass
     seg=pass(pass_resoltion,3);
    if(pass(pass_resoltion,1)>3&&pass(pass_resoltion,2)>3) %both inbound
%      if(pass(pass_resoltion,3))==1; %at segment 1

     A(rp+pass_resoltion,[arrival_table((pass(pass_resoltion,1)),seg),...
                arrival_table((pass(pass_resoltion,2)),seg)])=[1,-1];
     A(rp2+pass_resoltion,[departure_table((pass(pass_resoltion,1)),seg+1),...
                departure_table((pass(pass_resoltion,2)),seg+1)])=[1,-1];

    else %both outbound
     A(rp+pass_resoltion,[arrival_table((pass(pass_resoltion,1)),seg+1),...
                arrival_table((pass(pass_resoltion,2)),seg+1)])=[1,-1];
     A(rp2+pass_resoltion,[departure_table((pass(pass_resoltion,seg)),1),...
                departure_table((pass(pass_resoltion,2)),seg)])=[1,-1];
%       end
    end
end
end

c4_c5=v2(end);
c4_c6=c4_c5+1;
c4_c7=c4_c6;
con5=RF_c6+1+rmeet;
i=meetpoints;
j=(trains/2)-1;
coun1=0;
for third=1:6
    if(j>=1)
for col=1:i
    coun=coun+1;
for rot=1:j
coun1=coun1+1;
    c4_c7=c4_c7+meetpoints;
```

```
    A(con5,[c4_c6,c4_c7])=[1,-1];
    con5=con5+1;
end
c4_c6=c4_c6+1;
c4_c7=c4_c6;
end
    end
j=j-1;
end

c4_c8=(v2(end)+v3(end))/2;
c4_c9=c4_c8+1;
c4_c10=c4_c9;
con6=con5;
i=meetpoints;
j=(trains/2)-1;
for third=1:6
    if(j>=1)
for col=1:i
    coun=coun+1;
for rot=1:j
coun1=coun1+1;
    c4_c10=c4_c10+meetpoints;
    A(con6,[c4_c9,c4_c10])=[1,-1];
    con6=con6+1;
end
c4_c9=c4_c9+1;
c4_c10=c4_c9;
end
    end
j=j-1;
end

f=zeros(1,NofVar);
f(1:trains)= priority;
intcon=trains+1:v3(end);
options =optimoptions(@intlinprog,'OutputFcn',@savemilpsolutions);
% options.BranchingRule='mostfractional';
% options.CutGeneration='none';
% options.Heuristics='round';
% options.NodeSelection='mininfeas';
% options.NodeSelection='minobj';
% options.RootLPAlgorithm='dual-simplex';
[x,FVAL,EXITFLAG,OUTPUT] = intlinprog(f,intcon,A,b,Aeq,beq,lb,ub,options);
ToDraw=zeros(trains,meetpoints);
v2c=0;
for TrCount=1:trains
 for draw=1:meetpoints
     v2c=v2c+1;
 ToDraw(TrCount,[2*draw,2*draw-1])=[x(v3(v2c)),x(v2(v2c))];
 end
```

```
end
ToDraw_out=ToDraw(1:outbound_trains,:);
ToDraw_in=ToDraw(1+inbound_trains:end,:);
shape_in=fliplr(ToDraw_in);

for TrCount=1:inbound_trains
 for draw=1:meetpoints
     v2c=v2c+1;
 shape_in(TrCount,[2*draw-1,2*draw])=shape_in(TrCount,[2*draw,2*draw-1]);
 end
end
sketch=(1/60)*[ToDraw_out;shape_in];
yy1=flipud(DISTANCE);
y1=zeros(c,1);
y2=zeros(c,1);
for ax=1:inbound_trains
    y1([2*ax,2*ax-1],1)=DISTANCE(ax);
   y2([2*ax,2*ax-1],1)=yy1(ax);
end
sr=1+outbound_trains;
xlswrite('Autobetter.xls',ToDraw);

figure
plot(sketch(1:outbound_trains,:),y1,sketch(sr:trains,:),y2);
ax = gca;
ax.YTick = DISTANCE;
grid;
title('optimal schedule');
xlabel('Time in Hour'); % x-axis label
ylabel('Distance in KM'); % y-axis label
```

Published with MATLAB® R2015b