FREIGHT TRAIN OPTIMIZATION AND SIMULATION

Thai Hoa Le

A thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Computer Science

Concordia University

Montréal, Québec, Canada

January 2013

School of Graduate Studies
CONCORDIA UNIVERSITY

This is to certify that the thesis prepared

By : **Thai Hoa Le**

Entitled : **Freight Train Optimization and Simulation**

and submitted in partial fulfilment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee :

Date January 2013

Abstract

Freight Train Optimization and Simulation

Thai Hoa Le

Train scheduling has already received a lot of attention, whether for passenger or freight trains. While the volume of goods transport has increased over the years, extensions of railway systems are very rare because they represent major investments for railway companies or governments. Accordingly, the railway companies are often operating freight trains in a system that is close to saturation. It follows that a very effective planning and optimization of the rail network is needed.

While passenger train schedules are relatively static and cyclic, and can be planned months ahead, freight train schedules are designed with a much shorter planning time period, sometimes even one day or few hours before train departures. Moreover, passenger train schedules must obey some strict time window constraints as trains must arrive and depart from stations in order for passengers to get off/on the trains according to the posted schedule. On the opposite, the schedule of the freight trains may vary according to the train lengths or loads, i.e., freight trains have a much greater variability in their average speed. Lastly, the track configuration of the freight trains does not have a dedicated direction as it is often the case for passenger trains. For all those reasons, the scheduling of freight trains is more complex than for passenger trains.

In this thesis, we propose a new dynamic row/column management algorithm for the schedule of freight trains in a single/double track railway mesh network system. While many works have already been devoted to train scheduling, previously published optimization models all suffer from scalability issues. Moreover, very few of them take into account the number of alternate tracks in the railway stations or in the sidings for train meets, as well as the delay incurred by trains that take sidings. We propose a non time-indexed model, which takes into account such constraints, and we design an original solution scheme with iterative additions/removals of constraints/variables in order to remain with a manageable sized mixed integer linear program, while still ensuring convergence to an optimal solution. Numerical results are presented on several data instances of CPR (Canada Pacific Railway) on the Vancouver-Calgary corridor, one of the busiest corridors in their railway system.

In addition, we developed a simulation tool within the Arena framework, for the scheduling of freight trains. Comparisons of the simulation and optimization tools are made, together with a review of the pros and cons of simulation against optimization tools.

Acknowledgments

Writing a thesis is a very difficult task and to recognize how much help I received during my research is even a more challenging one.

First, I would like to thank my first co-supervisor, Doctor Brigitte Jaumard for her patient and enlightening guidance. Most of my knowledge in the field of mathematical modelling and optimization are gained thanks to her. Her instructions are invaluable, her insights so crucial to the model building and solving process. I would like to thank also my second co-supervisor, Doctor Ali Akgunduz, who have spent a lot of time teaching me how to make a simulation and who have given me many excellent comments and suggestions while I was struggling to build the simulation. I could not forget the many hours we spent in his office discussing on how to solve the deadlock issue.

This work could not have been fulfilled without the highly effective cooperation from the Canadian Pacific Railway (CPR) people. Space limited, I would like to mention here two of them. Peter Finnie is our contact person in CPR from the very beginning of our research project and he has always been very efficient to answer our numerous questions in order to build the most appropriate optimization and simulation models. He spent hours explaining us the context of train management and the key parameters for building a train scheduling tool with the most meaningful output parameters. Harold Beaty has also provided us with insightful comments. I would also like to mention the week visit I made at the headquarters of CPR in Calgary in March 2012, during which I was able to meet and exchange with several groups of CPR, allowing me to fully understand the many challenges of freight train scheduling.

Last but not least, I would like to express deep gratitude to my parents who have raised me up and who always encourage me. My thanks also go to my wife, Thanh Huong Bui for her understanding, her sharing and her patience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Due to its nature, railway is among the most efficient transportation means. Freight trains carry goods, commodities, ... in massive amount and at low rates. As the cost of energy is at record highs, the role of railway transportation is becoming more and more important. This means an ever increasing railway traffic.

However, the existing infrastructure (the railway networks) cannot grow proportionally to the growth of railway traffic. This fact is due to several reasons; the most evident one is that expanding a railway network (for example, building more sidings or more double tracks) is very expensive. Second, in many cases, expanding a network is simply impossible due to space limitations (as in case of tracks in a city or in a mountainous area).

In order to handle the increasing traffic, railway companies have no choice but to better exploit the available railway networks. The main goal of this study is to model the railway traffic with the objective of finding an optimized (ideally optimal) train scheduling under several realistic conditions such as infrastructure limitations, speed, expected arrival departure times, cost (delays, earliness), etc. The newly designed scheduling model and algorithm would serve as an analyzing tool as well as a planning tool for the train operators.

1.2 Original Contributions

In this study, we first propose a new optimization model that addresses freight train scheduling constraints of CPR in the context of a mesh railway network. We come up with a new way of modeling the freight train scheduling problem that can handle all the required constraints in a railway network, including several critical constraints that were not covered by any previous work in the literature, i.e., capacity of the sidings, selection of the train which takes sidings, modulation of the speeds of the trains taking sidings and isolated double tracks.

Furthermore, we propose a simple but highly effective methodology, namely dynamic row/column management algorithm to solve the problem. The proposed solution methodology enables us to solve real-life problems that are taken from CPR's operations within a desirable time-frame.

Secondly, a simulation model has been developed both for verification of the optimization model and for further investigating the railway dynamics under stochastic environment. One of the challenging problems in railway simulation, the deadlock problem was addressed by a simple control logic. This simulation, being an independent meaningful work by itself, also produces results as a benchmark for the optimization model.

1.3 Plan of the Thesis

The remainder of this thesis is organized as follows. The relevant background on freight train scheduling as well as on mathematical modeling and solution methodologies is discussed in Chapter 2. Next, a brief overview of the current literature of train scheduling is provided in Chapter 3. The core part of the thesis, the optimization model is described in Chapter 4 with details on modeling, solution procedure and in Chapter 5 with numerous numerical results. Chapter 6 is devoted to the simulation model. Therein, modeling details, the two deadlock avoidance conditions are presented. We also discuss comparative numerical results of simulation vs. optimization model. Finally, conclusions and the future directions are drawn in Chapter 7.

Chapter 2

Train Scheduling Optimization and Simulation Background

2.1 Generalities

We first introduce the terminology related to freight train scheduling.

Station: Location where trains are departing/arriving and/or delayed for various operations such as crew changes, refueling, goods loading/unloading.

Siding: Location where trains are passing or crossing each other.

Location point (or *point* for short): A station or a siding. From the modeling point of view, a station and a siding are often the same, so this term is often used to facilitate the description of the model when there is no need to distinguish a station from a siding.

Segment: The railway connecting a station/siding with an adjacent station/siding. The segment can be two-way single track or double track.

Single Track Segment: Segment with only one two-way track connecting two segment ends. A single track does not allow two trains on opposite directions to travel on it at the same time.

Double Track Segment: Segment with two two-way single tracks. Two trains in opposite directions can travel on the segment at the same time assuming they are on different tracks.

2.2 Train Scheduling Problem

The input to the train scheduling problem is the topology of the railway network and the list of trains that need to be scheduled with their earliest departure time at the origins (and possibly their latest arrival time at the destinations). In our mathematical model, the type of the rail-segment (single track or double track) is specified for each segment. Accordingly, all safety rules are incorporated in the mathematical model. For single track segment, two trains in opposite directions are not allowed to be on the same track segment at the same time. Trains in opposite directions can only meet each other at sidings or stations. For single track segments, two trains in the same direction can be on the segment at the same time, but they must maintain a safety distance, and they can pass each other only at sidings or stations. In order to model these safety conditions, we propose to investigate a non-time indexed modeling approach. The non-time indexed modeling approach enabled us to determine arrival and departure times at stations and/or sidings and average speeds on segments for all trains precisely. Consequently, the model guarantees the safety of all trains at all times according to the given safety rules. These rules will be described in more details in the next section.

The ultimate goal is to determine an optimal schedule for all trains by specifying the arrival and departure times at each location point, i.e., at each station or siding.

2.2.1 Objective

There are many possible objectives for train scheduling problems. One may choose to minimize the total traveling time of all trains in a given time window. Another objective may be to minimize the operation costs associated with all the trains in the network. If on-time arrival is concerned, the objective may be to minimize the total penalties of the late trains. In our study, we choose to optimize the average travel time of the trains, subject to some possible bounds on the travel times of some or all trains.

2.2.2 Constraints

Train operations are subject to several constraints. In this study, we focus on major constraints which are significantly affecting the performance of railway operations. We give special attention to the single track networks with a small number of double tracks. Indeed, CPR operates trains on such a railway network.

The first group of constraints is deadlock constraints. They forbid two trains passing the same segment in opposite directions at the same time.

The second group of constraints are safety constraints. They express that two trains going on the same segment in the same direction must always maintain a safety distance (headway).

The third group of constraints is related to the capacity of each station/siding. It requires that the number of trains dwelling at one station/siding must not exceed the number of alternate tracks of that station/siding, and must not remain on the main track.

The fourth group of constraints have to do with the traveling time of each train on a given segment. Since there is a speed limit in each segment, the traveling time on each segment cannot be smaller than some given minimal time. The other set of constraints in this group is related to the earliest departure times. They only allow a train to depart after its earliest departure time.

2.2.3 Optimization and Simulation

Optimization model

In the optimization model, we need to determine the arrival and departure times of each train at each location point. These arrival and departure times are defined by the corresponding decision variables and they are related by a set of linear inequalities which represents the constraints of the train scheduling problem. The solution is chosen from the set of feasible solutions (values of decision variables that satisfy the constraints), according to a certain criterion (or objective), e.g., minimal average travel time of all trains in the system. This criterion is called objective function and is represented by an expression linking the decision variables.

Solving the optimization model

Once the model is defined by a set of linear constraints, a linear objective function, and a mixed set of integer and continuous variables, it can be solved to optimality or heuristically.

As will be seen in Chapter 4, we propose a mathematical model, which is a MILP-Mixed Integer Linear Program. In order to solve it, we design an exact algorithm, which, if the computing times are too long, can be modified into a heuristic with some indications on the accuracy (an upper bound on how far is the heuristic solution from an optimal one) of the output solution. As will be developed in Section 4.4 of Chapter 4, we design a row/column management strategy in order to cope with the very large number of variables and columns, while preserving the possibility reach to an optimal solution for small to medium sized instances.

Simulation

A simulation model enables decision makers to test their ideas under stochastic conditions. The real power of simulation models is the capability of integrating many influential factors into the model under realistic operating conditions. The proposed mathematical model determines the arrival and departure times of each train at each location point in such a way that there is no conflict with any train pair at all times. On the contrary, the simulation model makes decisions locally which means, the arrival and departure times of each train at each location point are not computed at once so no conflict would occur later in the simulation. For example, the time of train t arriving at a location p is only known exactly when t reaches p and the departure time t from p is also only known when it departs. The departure time and arrival time of t at p will depend on the current condition (which is not known in advance due to stochastic nature) such as the traffic currently running in the segment in the opposite direction or the available tracks on the following stations. Another crucial difference is that the simulation is a best effort system. While it tries to achieve certain objectives, e.g., minimize the average train traveling time, there is no guarantee about the optimality of the schedules. Therefore, the simulation model is considered as a testing environment rather than a schedule planning tool. The simulation model we developed as part of this thesis enables railway companies to test the feasibility of given schedules under stochastic conditions. Furthermore, the effect of several other factors, e.g., failures or weather conditions can be precisely integrated into the model so that the decision makers can see the capabilities of the railway network and determine the best control mechanism to improve the traffic flows. The following section presents the main pros and cons of optimization and simulation approaches.

Comparison between optimization model and simulation

Optimization model

Pros:

- Able to give an optimal solution (corresponding to a certain criterion).
- Has a global view on all the decision variables at the same time. This helps to avoid problems that only arise if we look locally, e.g., deadlock problem.
- Development time is relatively short thanks to package solvers such as CPLEX.

Cons:

- Although development time is quite short, computing time for each real life dataset can be very long.

- In some cases, the model cannot even output any results due to, e.g., memory issue.
- If the system is too complex, it might be the case that it cannot be modelled by a mathematical model or it can be modelled by mathematical inequalities but the model is too hard to be solved (e.g., model with non linear constraints).

Simulation

Pros:

- User friendly, if animation is available then users will have a feel on how the system works in a real life environment.
- If the system under study is too complex, simulation may be the only choice.
- Often scalable if well designed.
- Shorter running times for each data instance and therefore easy to test many scenarios.
- Easy to incorporate some randomnesses, therefore easy to test different scenarios, in particular for train rescheduling problem.

Cons:

- Longer development time.

- Since simulation considers the network locally, it might encounter issues that are not seen in optimization problem. One notable issue is train deadlock, (a NP-hard problem).

Chapter 3

Literature Review

3.1 Introduction

Train scheduling is well studied and there are many works published in both passenger train scheduling and freight train scheduling domains. Indeed, passenger train scheduling is very different from freight train scheduling. For example, passenger train schedules must obey strict time windows constraints at each station along the trains' routes whereas for freight train schedules, strict time windows constraints are only enforced at the origins and/or the destinations. Another difference is that passenger train schedules are relatively static and cyclic while freight train schedules are not. In this chapter, we will only review the literature of both optimization and simulation models for freight train scheduling. However, the main focus will be the freight train scheduling optimization. For the simulation, we will review only the literature with respect to the main issue of the freight train scheduling simulation, namely the deadlock avoidance. Readers interested in train scheduling simulation may refer to [Mar99, DM04] for more details.

3.2 Train Scheduling Optimization

Among the various mathematical programming models which have been proposed for train scheduling, we can distinguish two classes of models. The first class of models corresponds to those relying on a classical MILP (Mixed Integer Linear Program) formulation, and they very often have a very large number of variables, and hence suffer from scalability issues and would ultimately resort to heuristic algorithms. This is the most popular approach and we are going to review the most recent ones in the next section. The second class of models relies on time-indexed column generation formulations, and assumes a time discretization. Such models offer satisfactory solutions for either periodic (day) scheduling or short coverage systems (e.g., a country in Europe), but lack scalability for timetable planning over a week or a month period, as well as for railway systems spanning a whole continent with several time zones. We are going to review one paper in this class.

3.2.1 Train Scheduling: ILP Models

Heuristics

Kraay and Harker [KH95] proposed a Mixed Integer Linear Program (MILP) with only a subset of the constraints (dwell times, train meets and overtakes, time windows on the departure/arrival times) which does not include any capacity constraint, i.e., limit on the number of available tracks at a given station/siding. Moreover, they use heuristics in order to solve their model as their solution process is not able to scale with the large number of constraints and variables. Experiments are very limited (less than 11 stations/sidings along a single line track). A very similar MILP model was developed by Higgins, Kozan and Ferreira [HKF96] and tested against a Tabu Search heuristic on data instances with up to 30 trains and 12 sidings. As for [KH95], the MILP model could not scale properly. Consequently, the authors only solve the linear relaxation of their MILP model, and use the lower bound it provides in order to evaluate the quality of their heuristic solutions. Indeed, they managed to solve only datasets consisting of 11 stations and an unspecified number of trains. Depending on the papers, the objective varies from minimizing the tardiness of the trains [HKF96] to minimizing the total arrival times [ZZ07, MD11] or to maximize the total profit of trains [CCT10].

A similar MILP model has been reused in [DLZL06, ZZ07, MD11, CL95]. In [ZZ07], Zhou and Zhong design a branch-and-bound based heuristic and a Lagrangian relaxation lower bound in order to solve data instances with up to 30 trains and 18 stations. In [MD11, CL95], the authors propose a vertical decomposition algorithm in order to overcome the scalability issues, i.e., dispatching the trains one by one, or one train cluster at a time in the MILP model. However, the size of successive MILP models to be solved is constantly increasing, and therefore the size of solved data instances is not much larger than those of previous studies. Note also that, in the first algorithm (*FixedPath*) of [MD11] and in [CL95], the definition of the route of a train includes whether to travel or not to travel a siding, and routes are defined at the outset (i.e, no optimization is made on which trains should travel which sidings). Track capacity constraints are enforced with flow conservation constraints which require the introduction of additional variables. In the second algorithm (*FlexiblePath*) of [MD11], routes are no more defined at the outset, however, several restrictions apply, in particular, two trains travelling in the same direction cannot run at the same time on an identical segment, one train behind the other one (under some headway constraints). The authors solved data instances with 4 trains using their exact models, and then larger data instances with up to 10 sidings and 24 trains with the help of heuristics and a parallel algorithm.

Model Decomposition

Another approach for solving the train scheduling problem is with the use of decomposition technique. However, it turns out to be quite challenging to come up with a decomposition model that could cover the required constraints. In [CCT10], for example, Cacchiani *et al.* propose a column generation model for the train scheduling in Italy. In this model, they discretize the arrival and departure times of each train at each station into minutes and they consider a time window of 1 day or 1440 minutes. A ”path”, i.e., a column, corresponds to a potential schedule of a train within one day and each train may choose one schedule only. The objective is to maximize the total profit of all trains. This model is quite limited. The constraints taken into account are safety constraints and trains can take over (pass) each other in station. The speed of a train between two stations is assumed to be constant and known in advance. Moreover, they consider only one direction of the network and they do not consider any capacity constraint, which is usually an important but hard one. In addition to considering only a limited number of constraints, the scope of the model is limited with respect to the number of trains it can solve. In addition, only one-day time window has been considered in the experiments.

3.2.2 Train Scheduling Simulation: Deadlock Avoidance

The crucial and often most difficult problem of train simulation is the deadlock avoidance. Deadlock is the situation in which some trains are blocking each other (due to operation constraints) and therefore no train can be moved.

Naturally, we would like to have an algorithm which is safe, i.e., that guarantees no deadlock will occur and at the same time maximizes the resource utilization or minimizes the total travel time of all trains. However, determining whether there is an imminent deadlock (with respect to the network capacity as well as current locations and directions of the trains) is a NP hard problem [LDL04]. Therefore, most of the algorithms attacking this problem are heuristics that guarantee to have no deadlock but may not give an (sub) optimal solution with respect to a certain objective, i.e., minimizing the average travelling time. Another issue is that applying an algorithm that guarantees deadlock avoidance in all cases often leads to resource underutilization. Also, according to [Pac11] all proposed algorithms in the literature are not yet applicable to real life train dispatching systems or simulations.

One approach to the problem is to reuse some deadlock avoidance algorithm developed in the context of operating systems due to similar constraints and conditions that lead to a deadlock. An example of this is the Banker's algorithm, which is discussed in [Pac11] and [Cui10] for example. However, applying such an algorithm requires simplifying or modifying the train network constraint. For example, considering single two-way track segments as computer resources, and trains as computer processes that requires such resources. Banker's algorithm requires that one single two-way track segment can be used by at most one train. However, indeed, two trains travelling in the same direction can use that segment. In addition, Banker's algorithm was applied with little success [Pac11] and [Cui10]. Some authors try to modify the original Banker's algorithm but the modified one has only been tested with a small example [Cui10].

Another approach is the Dynamic Route Reservation [Cui10]. In this approach, a set of rules are defined and these rules are considered each time a train is about to move to the next node. Based on these rules, some resources such as segment track or siding track are reserved in advance for the train, i.e., before the train uses these resources. Early reservation may help to avoid deadlocks with some efficiency but so far it has been successfully applied to very simple railway network and few operation constraints. Similarly, Pachl [Pac11] proposes a set of rules for reserving routes, i.e., sets of sections (a section is a part of the segment tracks or siding tracks). A train must reserve sections according to these rules before it can move. However, the author does not specify how to use them in an algorithmic way. For example, the authors does not describe how a single track segment should be divided into sections and how many sections ahead a train should start reserving resources. Another issue is that these rules are quite conservative as they do not allow two trains to travel in the same direction in the same section at the same time. In fact, these rules are still only applied manually in some unspecified test cases of unknown complexities and the author does not include any experimental results in his paper.

Chapter 4

Optimization - Mixed Integer Linear Programming Model

4.1 Introduction

In this chapter, we are going to describe in detail our Mixed Integer Linear Programming (MILP) model, called SDT_TS model, that we built to solve the train scheduling problem, as described in Chapter 2. This chapter is organized as follows. In Section 4.2, we present the problem statement of the train scheduling in a single/double track freight train railway system, as that of CPR - Canadian Pacific Railway. The newly proposed optimization model is detailed in Section 4.3, with the inclusion of the siding management constraints, i.e., which trains take the sidings and the number of alternate tracks at a given siding or station (we are not aware of any optimization model which included such constraints). Solution to the optimization model, an original dynamic row and column generation/removal exact algorithm is next described in Section 4.4. Numerical results are presented in Section 5 on several data set instances in order to evaluate the performance of the optimization model, as well as an estimation of the network capacity of a railway system. Results are obtained with several CPR data set instances, with up to 78 sidings and up to 28 trains. Therein, we evaluate the performance of the optimization SDT_TS model, as well as the sensitivity of the train schedules to several parameters, i.e., length of the operation hours, length of the trains, flexibility around the planned departure times, priority of some trains over others.

4.2 Problem Statement

Our study considers a mesh rail network with single two-way tracks between stations or sidings as well as few double tracks. Each track is divided into segments which are separated by sidings or stations. Single tracks can be used by trains running in both directions, and trains can meet and pass at stations or sidings. Sidings are typically added to a railway line in order to allow two trains to pass one another and are the most common method used to expand capacity. Sidings are typically built long enough to permit all regular trains to come to a full stop inside the siding while remaining clear of the switches at either end. In this study, we will consider two types of trains, the so-called regular ones that can fit any siding, and the long ones (double the regular ones) that can fit only 8 specific sidings (details to be found in dataset description in Chapter 5).

The proposed optimization SDT_TS model, which will be detailed in the next section, builds a freight train schedule with all meaningful constraints. The input is the topology of the network as described by its set of segments and the list of trains that need to be scheduled, with their characteristics: origin/destination stations, expected departure times, length, average and maximum train speed. Length of the trains may restrict the number of sidings a train can take. Moreover, each train has a specific priority which depends on the train series, i.e., the types of goods. It may also depend on the customer contract agreements and the train loads.

We assume that the given railway network is a single track mesh network, with few double tracks. Two trains in opposite directions are not allowed to be on the same single track segment and they can meet each other only at a siding or a station or on double tracks. Two trains in the same direction can run on a segment at the same time but they must maintain a safety distance, and they can pass each other only at a siding or a station.

The output of the model is a schedule for each train that specifies the departure and arrival times at each siding/station, and consequently the earliness/tardiness on the expected departure time.

In this study, we focus on the objective of minimizing the average travel times between departure/destination stations, while investigating the impacts of the length of the trains, the flexibility around the planned scheduled departure times, as well as the number of operating hours in departing stations.

4.3 Optimization Model

4.3.1 Notations

Railway network parameters

P	$= P^{\text{STATIONS}} \cup P^{\text{SIDINGS}}$, indexed by p, where P^{STATIONS} (resp. P^{SIDINGS}) is the set of station (resp. siding) locations.
S	Set of segments in the railway network, indexed by s. A segment is a single track between two successive locations (either a station or a siding) of P. Written under the form $s = [p, p']$, it means we consider an ordered pair of locations, with p traversed before p'.
D	Set of double track segments in the railway network, indexed by (s, s'). A double track segment is a segment with two-way single tracks between two successive locations (either a station or a siding) of P. Written $d = (s, s')$, it means we consider an ordered pair of two single tracks.

Train parameters

T	Set of trains, indexed by t
T_p (resp. T_s)	Set of trains that go through location p (resp. segment s)
$T_s^{\rightarrow}$ (resp. $T_s^{\leftarrow}$)	Set of trains that travel segment $s = [p, p'] \in S$ going through p (resp. p') first
$T_s^{\rightrightarrows}$ (resp. $T_s^{\rightleftarrows}$)	Set of (t, t') pairs of trains that travel segment $s = [p, p'] \in S$ in the same (resp. in opposite) direction(s)
$\textsc{src}(t)$	Departure station of train t
$\textsc{dst}(t)$	Destination/arrival station of train t
S^t	List of segments defining the route of train t from $\textsc{src}(t)$ to $\textsc{dst}(t)$
$\overline{d}^t_{\textsc{src}(t)}$	Expected (planned) departure time of train t at its origin location
π^t	Priority (e.g., series number) of train t in the network

Location and train parameters

dw_p^t	Minimum dwell time of train t at location point p. If p is only a location that train t is passing through, then $\mathrm{dw}_p^t = 0$. We assume that $\mathrm{dw}_p^t = 0$ for $p \in P^{\textsc{sidings}}$, and $\mathrm{dw}_p^t \geq 0$ for $p \in P^{\textsc{stations}}$
$\textsc{cap}_p$	The capacity, in terms of the number of tracks, of the siding located at p, i.e., the number of parallel tracks, excluding the main track. For the time being, we assume that each siding can host any regular train, one at a time, and that only four of them are long enough in order to accommodate the long trains. We will assume that $\textsc{cap}_p = 1$ for $p \in P^{\textsc{sidings}}$.

We assume that all times are expressed in minutes. In order to simplify the expression of the constraints, we assume that all constraints are expressed in terms of times, meaning that the average/maximum speeds are translated into times it takes for a train to travel a given segment:

r^t_s	Average time for train t to travel segment $s = [p, p']$ with $p, p' \in P$, i.e., $r^t_s = \text{Distance}(p, p')/(\text{Average speed of } t \text{ on } s)$.
$\underline{r}^t_s$	Minimum time for train t to travel segment $s = [p, p']$ with $p, p' \in P$, i.e., $\underline{r}^t_s = \text{Distance}(p, p')/(\text{Maximum speed Limit of } t \text{ on } s)$.
$\overline{r}^t_s$	Maximum time for train t to travel segment $s = [p, p']$ with $p, p' \in P$, i.e., $\overline{r}^t_s = \text{Distance}(p, p')/(\text{Minimum speed Limit of } t \text{ on } s)$.
τ^t_s	Time required for train t to travel the safety distance on segment $s = [p, p']$.

4.3.2 Variables

The first set of variables are related to the arrival and departure times of the trains.

d^t_p	Departure time of train t from location p
EARLY^t_d	Earliness of train t at departure (source) station
LATE^t_d	Lateness of train t at departure (source) station
a^t_p	Arrival time of train t at location p
DELAY^t_d	$= \max\{\text{EARLY}^t_d, \text{LATE}^t_d\}$

All the above variables are real valued variables.

Both arrival and departure time values will be rounded to the closest minute in practice. We use real valued variables to model them to simplify the solution of the model.

A train schedule, $\textsc{schedule}(t)$, is characterized by its arrival/departure time at every station/siding along its way from origin to destination.

$\textsc{schedule}(t) = [(a^t_{\textsc{src}(t)}, d^t_{\textsc{src}(t)}), \ldots, (a^t_p, d^t_p), \ldots, (a^t_{\textsc{dst}(t)}, d^t_{\textsc{dst}(t)})]$.

The next set of variables corresponds to decision variables, which takes their values in $\{0, 1\}$.

For any $t, t' \in T_p^{\rightrightarrows} : t < t'; s = [p, p'] \in S; p, p' \in P$:

$$\theta_p^{tt'} = \begin{cases} 1 \text{ if } t \text{ leaves station/siding } p \text{ before } t' \text{ , i.e., } a^t_p \leq a^{t'}_p; \\ 0 \text{ otherwise.} \end{cases}$$

For any $t, t' \in T_p : t < t'; p \in P$:

$$\alpha_p^{tt'} = \begin{cases} 1 \text{ if intervals } [a^t_p, d^t_p] \text{ and } [a^{t'}_p, d^{t'}_p] \text{ overlap,} \\ 0 \text{ otherwise.} \end{cases}$$

For any $t, t' \in T_p; p \in P$:

$$\beta_p^{tt'} = \begin{cases} 1 \text{ if } a^t_p \leq d^{t'}_p, \text{ i.e., if train } t' \text{ departs after the arrival of train } t \text{ at point } p, \\ 0 \text{ otherwise, i.e., if } a^t_p > d^{t'}_p. \end{cases}$$

Note that $a^t_p > d^{t'}_p$ implies $a^{t'}_p \leq d^t_p$ (train t departs after the arrival of train t' at point p), but the reverse is not true $a^{t'}_p \leq d^t_p \not\Rightarrow a^t_p > d^{t'}_p$.

Based on the definitions of $\alpha_p^{tt'}$ and $\beta_p^{tt'}$, we have:

$$\alpha_p^{tt'} = \beta_p^{tt'} \beta_p^{t't} = \beta_p^{tt'} + \beta_p^{t't} - 1, \qquad p \in P, t, t' \in T : t < t'.$$

For any $t \in T$, and a double track segment $d = (s, s')$ between p and p':

$$x_s^t = \begin{cases} \text{1 if train t travels on s,} \\ \\ \text{0 otherwise if it travels on s'.} \end{cases}$$

Note that we do not need to define the variable x'^t_s. In other words, we need to define only one variable per double track segment. Also, we let the optimization solution decides on which track is used by a given train. Therefore if $s \in S^t$, we also have $s' \in S^t$. In order to go around the symmetrical solution (which lengthen the solution of the optimization model), we will assume that: $x_s^t \leq x_s^{t'}$ for all $t, t' \in T^s \cup T^{s'}$.

For any $t \in T$, and a siding/station location $p \in P$:

$$y_p^t = \begin{cases} \text{1 if train t remains on the main track,} \\ \\ \text{0 otherwise.} \end{cases}$$

4.3.3 Minimize the Train Travel Times

Minimize the Train Travel Times.

We look at the objective of minimizing the train travel times in order to estimate the network capacity, i.e., the maximum number of trains which can be running on the tracks without deteriorating too much the average travel times between source/destination stations. Indeed, when a railway network is overloaded, waiting times, in addition to the dwell time and necessary times for meeting or bypassing at sidings, are increasing. We also allow some light changes in the planned departure/arrival times.

The objective can then be expressed as follows:

$$\min \quad \frac{1}{|T|} \sum_{t \in T} \pi^t \, (a^t_{\text{DST}(t)} - d^t_{\text{SRC}(t)}) \tag{1}$$

subject to the following first set of constraints in order to minimize the earliness/tardiness of the trains around the departure times. For each train $t \in T$, we have:

$$d^t_{\text{SRC}(t)} = \overline{d}^t_{\text{SRC}(t)} + \text{LATE}^t_d - \text{EARLY}^t_d; \quad \text{DELAY}^t_d \geq \text{EARLY}^t_d \quad ; \quad \text{DELAY}^t_d \geq \text{LATE}^t_d \tag{2}$$

Constraints (2) allow the departure time of any train t to be delayed or advanced, i.e., to lie in $[\overline{d}^t_{\text{SRC}(t)} - \text{EARLY}^t_d, \overline{d}^t_{\text{SRC}(t)} + \text{LATE}^t_d]$, if it allows a reduction of the travel time of train t due to better train meets.

4.3.4 Train Scheduling Constraints

The remaining set of constraints is divided into several subsets of constraints which are next described.

Dwell constraints

For all $t \in T, p \in P$

$$d^t_p - a^t_p \geq \text{dw}^t_p. \tag{3}$$

Constraints (3) enforce that differences between departure and arrival times should be large enough in order to allow the planned train operations in each station.

Travel time constraints

For all $t \in T_s^{\rightarrow}; s = [p, p'] \in S^t; p, p' \in P$

$$a_{p'}^t - d_p^t + 3\tau_s^t(y_p^t + y_p^{t'}) \geq \underline{r}_s^t + 3\tau_s^t \tag{4}$$

$$a_{p'}^t - d_p^t \leq \overline{r}_s^t. \tag{5}$$

Constraints (4) take care of the speed decrease/increase of a train taking a siding, while constraints (5) enforce speed limitations on each segment.

Safety distance constraints

For all $s = [p, p'] \in S^t \cap S^{t'}; t, t' \in T_s^{\rightrightarrows} : t < t'; p, p' \in P$,

Single track:
$$d_p^{t'} - d_p^t \geq \tau_p^{t'} - M(1 - \theta_p^{tt'}) \tag{6}$$

$$a_{p'}^{t'} - a_{p'}^t \geq \tau_p^{t'} - M(1 - \theta_p^{tt'}) \tag{7}$$

$$d_p^t - d_p^{t'} \geq \tau_p^t - M\theta_p^{tt'} \tag{8}$$

$$a_{p'}^t - a_{p'}^{t'} \geq \tau_p^t - M\theta_p^{tt'}. \tag{9}$$

Double track
$$x_s^t \geq x_s^{t'} \tag{10}$$

$$d_p^{t'} - d_p^t \geq \tau_s^{t'} - M(1 - \theta_p^{tt'}) - M(x_s^t - x_s^{t'}) \tag{11}$$

$$a_{p'}^{t'} - a_{p'}^t \geq \tau_s^{t'} - M(1 - \theta_p^{tt'}) - M(x_s^t - x_s^{t'}) \tag{12}$$

$$d_p^t - d_p^{t'} \geq \tau_s^t - M\theta_p^{tt'} - M(x_s^t - x_s^{t'}) \tag{13}$$

$$a_{p'}^t - a_{p'}^{t'} \geq \tau_s^t - M\theta_p^{tt'} - M(x_s^t - x_s^{t'}). \tag{14}$$

Constraints (6) to (14) enforce the safety constraints, forcing two consecutive trains to maintain a headway between them.

Capacity constraints

$$a_p^t \leq d_p^{t'} + M(1 - \beta_p^{tt'}) \qquad p \in P, t, t' \in T \tag{15}$$

$$d_p^{t'} < a_p^t + M\beta_p^{tt'}. \qquad p \in P, t, t' \in T \tag{16}$$

$$\alpha_p^{tt'} + \alpha_p^{tt''} + \alpha_p^{t't''} \leq 2 \qquad p \in P; \quad t, t', t'' \in T_p : t < t' < t'' \tag{17}$$

$$\sum_{t \in T'} \sum_{t' \in T' : t < t'} \alpha_p^{tt'} \leq \binom{\text{CAP}_p + 2}{2} - 1 = \frac{\text{CAP}_p(\text{CAP}_p + 3)}{2} \qquad p \in P; \; T' \in T_p : |T'| = \text{CAP}_p + 2. \tag{18}$$

Constraints (15) to (18) take care of the siding or station capacity constraints, i.e., the limitations imposed by the number of available alternate tracks in a given location.

Deadlock constraints

For all $s = [p, p'] \in S; t \in T_s^{\rightarrow}, t' \in T_s^{\leftarrow}; p, p' \in P$ in two-way single-track:

$$a_p^{t'} \leq d_p^t + M(1 - \beta_p^{t't}) \tag{19}$$

$$a_{p'}^t \leq d_{p'}^{t'} + M(1 - \beta_{p'}^{tt'}) \tag{20}$$

$$\beta_p^{t't} + \beta_{p'}^{tt'} \geq 1. \tag{21}$$

For all $s, s' \in (S^t \cup S^{t'}) \times (S^t \cup S^{t'}), s \neq s'$, s and s' have the same endpoints:

$p, p'; t, t' \in T_s^{\rightleftarrows}; p, p' \in P$ in two-way double-track:

$$a_p^{t'} \leq d_p^t + M(1 - \beta_p^{t't}) + M(x_s^t - x_s^{t'}) \tag{22}$$

$$a_{p'}^t \leq d_{p'}^{t'} + M(1 - \beta_{p'}^{tt'}) + M(x_s^t - x_s^{t'}) \tag{23}$$

$$x_s^t \geq x_s^{t'} \tag{24}$$

$$\beta_p^{t't} + \beta_{p'}^{tt'} \geq 1 - (x_s^t - x_s^{t'}). \tag{25}$$

Constraints (19) to (25) prevent deadlocks resulting from the situation of two trains running in opposite directions on the same track segment; first in a single track railway system (Constraints (19) - (21)) and then in a double track railway system (Constraints (22) - (25)).

Siding constraints

For all $p \in P^{\text{SIDINGS}};\ t, t' \in T_p$

$$a_p^t \geq d_p^{t'} + M(1 - \beta_p^{tt'}) \tag{26}$$

$$a_p^{t'} \geq d_p^t + M(1 - \beta_p^{t't}) \tag{27}$$

$$y_p^t + y_p^{t'} \leq 3 - \beta_p^{tt'} - \beta_p^{t't}. \tag{28}$$

Constraints (26) - (28) select the trains that take the sidings.

Train meeting constraints

For all $t \in T, p \in P$

$$d_p^t - a_p^t \leq M(1 - y_p^t) \tag{29}$$

$$\mathrm{dw}_p^t \, y_p^t = 0 \tag{30}$$

$$\sum_{t':t>t'} \alpha_p^{t't} + \sum_{t':t<t'} \alpha_p^{tt'} + \mathrm{dw}_p^t \geq 1 - y_p^t. \tag{31}$$

Constraints (29) enforce that a train which remains on the main track must keep moving at any location. If there is some dwell times for some trains, then Constraints (30) force the trains to free the main track. Last, if a train does not meet any train at a siding/station, it must remain on the main track, see Constraints (31).

Please note that in Cplex, strict inequalities are not allowed. Consequently, in practice, we enforce the strict inequalities by using an ε value. Indeed, a strict equality of the type $ax < b$ is rewritten as $ax \leq b - \varepsilon$ with ε is set to a hundredth of a minute in our program.

4.4 Solving the SDT_TS (Single Double Track - Train Scheduling) Model

4.4.1 Introduction

As has been seen in the Section 4.3, we take into account many types of constraint of the train scheduling problems. On top of that, the numbers of some types of constraints, e.g., deadlock constraints, capacity constraints, siding constraints are often in the order of $|P| \times |T|^2$, which is very high if we consider real life data instances. Similarly, the number of variables is also very high and is in the order of $|P| \times |T|^2$. For example, in the data instance of 20 trains and 1 subdivision (out of 5 subdivisions of the network we study), there are 22,240 constraints and 11,200 binary variables. Moreover, most of the variables are binary, which makes solving the problem even harder. Therefore, if we put all constraints and variables into the model, we will not able to solve the optimization model with data instances of reasonable sizes (indeed, we could not go beyond the instance of 10 trains and 1 subdivision). In other words, we do need a good algorithm to manage the number of constraints and variables to solve our problem. In the next section, we will describe such an algorithm.

4.4.2 General Framework of the SDT_TS Algorithm

In order to overcome the large number of constraints and variables, we propose a row and column generation algorithm, called SDT_TS algorithm, in which we iteratively add/remove some rows and columns until we reach an ε-optimal train schedule. Indeed, the idea is to start with a rather small optimization model made of constraints (2) - (5) only, i.e., of the constraints involving only continuous variables: the earliness and tardiness constraints around the departure times (2), the dwell constraints (3), the travel time constraints (4), and the maximum speed constraints (5).

The resulting MILP model is then solved, and then we check the feasibility of the solution, examining the constraints taking care of the interaction between two (or more) train schedules. Note that those last constraints, namely, constraints (6) up to (31), each involves one or two binary variables (with some constraints sharing the same binary variable(s)), so that their addition to the incumbent mathematical program will often entail the addition of one or two new 0-1 variables. A compromise has to be found for the number of added constraints and variables at each iteration between the following two extreme strategies: adding one violated constraint at a time or adding all violated constraints. With the first strategy, the convergence might be too slow, while with the second strategy, we might end up very quickly with an unnecessary large set of constraints and variables. Once we have added some or all violated constraints, the optimization model becomes a MILP model, which is solved again, and we keep adding violated constraints until all constraints are satisfied. Note that, in practice, it does not require solving a MILP with all possible constraints, but with a quite small fraction of the overall set of variables and constraints, as will be seen in the next chapter.

For the addition of the violated constraints, we consider the following strategy. Trains are ordered according to a given criterion. In this study, we order the trains according to their departure time, alternating between westbound trains and eastbound trains (as the rail network we consider is an East $\leftrightarrow$ West one). Remaining ties, if any, are arbitrarily broken. At iteration $\textsc{iter} \geq 2$, after solving the current MILP, we revisit the constraints for all the train interaction constraints, namely, (6) up to (31) with respect to the first ITER trains, identify the ones which are violated and add them to the current MILP. Once we reach iteration $\textsc{iter} = |T|$, we may need several iterations before reaching a feasible schedule, i.e., train schedules which satisfy all constraints.

Note that in the course of the iterations, we may have too many constraints and variables, so that the scalability of the current MILP is impaired. In such a case, except for constraints (2) - (5), we remove all the other constraints which are not binding constraints in the last computed MILP solution.

The flowchart of Figure 1 summarizes the algorithm for solving the SDT_TS model.

In the next section, we will give some details about how we identify the violated interactive constraints and add them to the model. Also, we will describe how we

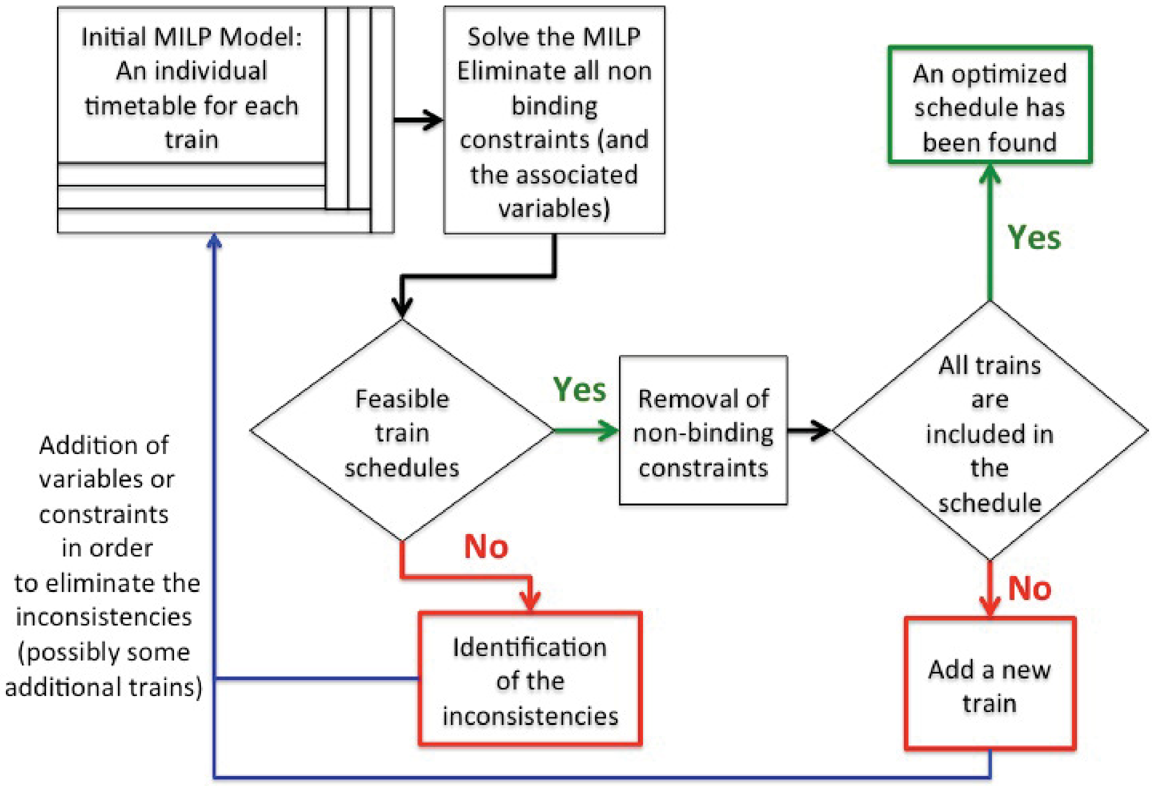

Initial MILP Model: An individual timetable for each train
Solve the MILP Eliminate all non binding constraints (and the associated variables)
An optimized schedule has been found
Yes
Feasible train schedules
Yes
Removal of non-binding constraints
All trains are included in the schedule
Addition of variables or constraints in order to eliminate the inconsistencies (possibly some additional trains)
No
No
Identification of the inconsistencies
Add a new train

Figure 1: Flowchart of the solution process

identify and remove the non binding constraints in order to maintain the manageable size of the MILP.

4.4.3 Solving the Separation Problem

As mentioned in the previous section, we do not embed all constraints when we solve the model therefore we may encounter some infeasibility, i.e., the solution obtained might not satisfy some constraints that are currently not considered in the model. Therefore, given a solution with respect to the current set of constraints and variables, we need an algorithm to identify which (not necessarily all) constraints that are violated. After the violated constraints are identified, we add them to the current model.

However, if we add all these violated constraints, many of them might become redundant. Consequently, adding all violated constraints might increase significantly and unnecessarily the running time after each iteration. Therefore after each iteration, we will add only a limited number of violated constraints. This number can be parametrized and is currently set to 100 in our program.

The next sections describes the details on identifying and adding a set of violated constraints. At the end of each description of each set of constraints, we give some rough estimation of the complexity.

Deadlock constraints. We have to check whether any deadlock constraint, i.e., no two trains on opposite direction can travel on the same track at the same time, is violated.

The algorithm will check, for each segment $s = [p, p']$ and for each pair (t, t') such that t goes from p to p' and t' from p' to p whether t and t' have a conflict in

(p, p'). From the explanation of the constraint (19)-(21), we can deduce that the deadlock constraints are violated on a segment $[p, p']$ if $d^t_p \leq a^{t'}_p$ AND $d^{t'}_{p'} \leq a^t_{p'}$ in case s is single track. If s is a double track, then on top of this condition, the deadlock constraints are violated if t and t' are on the same track (of the double track). When a deadlock constraints are violated with respect to s and (t, t'), we add the 3 corresponding constraints (19) - (21) if s is single track, or (22) - (25) if s is double track.

```
1. For each segment $s = [p,\ p']$ in the network
2.     For each pair $(t, t')$ such that $t$ goes from $p$ to $p'$ and $t'$ from $p'$ to $p$
3.         If $d^t_p \leq a^{t'}_p$ AND $d^{t'}_{p'} \leq a^t_{p'}$ AND
           ($s$ is single track OR ($s$ is double track and $t$ and $t'$ are on the same track)
           //If train $t$ and $t'$ conflict on segment $s = [p, p']$
4.             Add $\beta^{t't}_p$ and $\beta^{tt'}_{p'}$ to the set of $\beta$ variables
5.             Add $(t,\ t')$ to the set of train pairs for which we need to enforce
               the deadlock constraints in $s$
6.             Add the corresponding deadlock constraint (19) - (21) if $s$ is single
               track, or (22) - (25) if $s$ is double track.
```

The "For" loop in line 1 checks for all segments $s = [p,\ p']$ in the set of segments in the network, which are roughly $|P|$ segments. The "For" loop in line 2 checks for all pairs of (t, t') such that t goes from p to p' and t' from p' to p, which are roughly $|T|/2 \times |T|/2 = |T|^2/4$. In all, the complexity of this checking algorithm is roughly

$|P| \times |T|^2/4$ or $O(|P| \times |T|^2)$.

Capacity constraints The idea is to check for each location p, whether there are more trains dwelling at the same time than its capacity (its number of alternate tracks plus the main track). We do so as follows: for each train t passing p in its route, we count the number of trains t' also passing p and such that the arrival time of t at p, a_p^t, is in the interval of $[a_p^{t'}, d_p^{t'}]$. In other words, the dwelling of t' overlaps that of t at p or for short, t' overlaps t at p. If the number of such train t' is bigger than the capacity of p then we add the capacity constraints for t at p, i.e., (15), (16) and (17) (or (15), (16) and (18) if $CAP_p = 2$). However, when we add these capacity constraints, we do not consider all trains that pass the point p but only trains t' that overlaps t. That way we can reduce the number of constraints in the set of capacity constraints for t at p.

```
1. For each point $p$
2.     For each train $t$ that passes $p$
3.         Count = 0
4.         Set of trains overlapping with $t$ at $p$ $\textsc{Overlap}(t, p) = \emptyset$
5.         For each train $t'$ that passes $p$
6.             If $a_p^t \geq a_p^{t'}$ and $d_p^t \leq d_p^{t'}$ // If $t'$ overlaps with $t$
7.                 Count = Count + 1
8.                 Add $t'$ to the set of overlapping trains with $t$:
9.                 $\textsc{Overlap}(t, p) \leftarrow \textsc{Overlap}(t, p) \cup \{t'\}$
```

```
10.         If Count $\geq CAPA_p$ // Capacity constraint violated
            //Add the capacity constraints for $t$ at $p$:
11.             For each train pair $(t_1, t_2)$ in the set $t \cup$ Overlap$(t, p)$
12.                 Add $\beta_p^{t_1 t_2}$ and $\beta_p^{t_2 t_1}$ to the set of $\beta$ variables
13.                 Add the corresponding capacity constraints (15), (16) and (17)
                    (or (15), (16) and (18) if $CAP_p = 2$)
```

The "For" loop in line 1 checks for all points p in the network, which are roughly $|P|$ points. The "For" loop in line 2 checks for all train t such that t goes through p, which are roughly $|T|$. The "For" loop in line 5 (which is inside "For" loop in line 2) also checks for all train t such that t goes through p and has roughly $|T|$ loops. The "For" loop in line 11 (which is also inside "For" loop in line 2) may have at most $|T|^2$ loops. In all, the complexity of this checking algorithm is roughly $|P| \times |T| * (|T| + |T|^2)$ or $O(|P| \times |T|^3)$.

Siding constraints. The siding constraints are checked in a very similar way to capacity constraints. The difference is that at each point p, for a give capacity constraint, we add a new one only if the number of trains that overlaps each other are larger than the capacity. For siding constraints, we add a new one whenever two trains overlap. The addition of siding constraints is as follows:

```
1. For each point $p$
2.     For each train $t$ that passes $p$
```

```
3.      Count = 0
4.      Set of trains overlap with $t$ at $p$ OVERLAP$(t,p) = \emptyset$
5.      **For** each train $t'$ that passes $p$
6.          **If** $a_p^t \geq a_p^{t'}$ and $d_p^t \leq d_p^{t'}$
7.              Count = Count + 1
8.              **Add** $t'$ to the set of train overlaps with $t$, OVERLAP$(t,p)$
9.      **If** Count $\geq 1$
          //Add the siding constraints for $t$ at $p$:
10.       **For** each pair train $t'$ in the set OVERLAP$(t,p)$
11.             **Add** $\beta_p^{tt'}$ and $\beta_p^{t't}$ to the set of $\beta$ variables
12.             **Add** the corresponding siding constraints (26) - (28)
```

This checking algorithm is very similar to that of the capacity constraints and therefore also has complexity $|P| \times |T|^3$ or $O(|P| \times |T|^3)$.

We can see that the complexities of each of the three checking algorithms are $O(|P| \times |T|^2)$ or $O(|P| \times |T|^3)$. So the overall algorithm for the identifying and adding violated constraints has complexity $O(|P| \times |T|^3)$.

4.4.4 Removing Non Binding Constraint and Corresponding Variables

As mentioned above, after one iteration, the number of non binding constraints become very big and we need to remove these now-redundant constraints. In practice without removing the redundant constraints, i.e., only adding the violated constraints in each iteration, we could not solve further than around 20 trains and 1 subdivision. In the next section, we are going to describe how the removal of constraints is done.

Removing the non binding deadlock constraint. The set of deadlock constraints of a segment s is indexed by the set of pair of trains $(t,\, t')$ which need to enforce the deadlock constraints. So removing a non binding constraint is equivalent to removing the corresponding pair $(t,\, t')$ from that set. A deadlock constraint for $(t,\, t')$ and segment s is identified as non binding if the travelling time of train t on s does not strictly overlap with that of t', i.e., if the time finish t for travelling s is **strictly smaller** than the start time t' for travelling s ($a^t_{p'} < d^{t'}_{p'}$) or vice versa. Below is the detail of the algorithm.

```
1. For each segment $s$
2.         For each $(t,\, t')$ in the set P of pairs of trains which are currently
           checked for deadlock constraints in segment $s = [p, p']$
3.                 If the time finish $t$ for travelling $s$ is strictly smaller than the time
                   start $t'$ for travelling $s$
                   or the time finish $t'$ for travelling $s$ is strictly smaller than the time
                   start $t$ for travelling $s$
                   //($a^t_{p'} < d^{t'}_{p'}$ or $a^{t'}_p < a^t_p$)
4.                     Remove $(t,\, t')$ from the set $P$
```

```
5.          Remove also $\beta_p^{t't}$ and $\beta_p^{tt'}$ from the set of $\beta$ variables
6.          Remove the corresponding deadlock constraints (19) - (21) if $s$ is single
            track, or (22) - (25) if $s$ is double track.
```

The "For" loop in line 1 checks for all segments s in the network, which are roughly $|P|$ points. The "For" loop in line 2 checks for all pairs (t, t') which are currently checked for deadlock constraints in segment $s = [p, p']$. This number is at most $|T|^2/4$ but normally much fewer, i.e., in order of $|T|$. In all, the complexity of this removing algorithm is roughly at most $|P| \times |T|^2/4$ but normally only $|P| \times |T|$.

As a remark, in the program, we define, e.g., $a_{p'}^t$ is **strictly smaller** than $d_{p'}^{t'}$ if $a_{p'}^t < d_{p'}^{t'}$ - τ where τ is 60 minutes.

In principle, we could set τ to few minutes or less and therefore can remove more non binding constraints in the removing step. However, by doing so, we may remove constraints which are satisfied so tightly that the chance they are violated again and need to be added in a later iteration is high. Therefore, the solving process might be less incremental, more iterations might be needed and the running time might increase significantly. Setting τ to 60 minutes, a rather large value, ensures that the non binding constraints being removed have little chance to be violated again, which ultimately speeds up the solving process.

Removing the non binding capacity constraint The set of capacity constraints of a station p is indexed by the set S of set of pair of trains sp $\{(t_1,\ t_2),$

$(t_1,\ t_3)$,...$(t,\ t_k)$,$(t_2,\ t_3)$,...$(t_{k-1},\ t_k)$ }, i.e., all possible combination of two trains in the set $t_1,\ t_2$,...,t_k}. So removing a non binding capacity constraint is equivalent to removing this set *sp* from S. We identify the non binding capacity constraint by counting the number of trains in the set $(t_1,\ t_2,..t_k)$ which overlaps against each other. If this number is smaller than the capacity then we remove the set of pairs $\{(t_1,\ t_2),\ (t_1,\ t_3),...(t,\ t_k),(t_2,\ t_3),...(t_{k-1},\ t_k)\}$ mentioned above.

```
1. For each station $p$
2.      For each set of pairs $sp$ $\{(t_1, t_2), (t_1, t_3),...(t, t_k),(t_2, t_3),...(t_{k-1}, t_k)\}$ in the
        set S of set of pairs of trains which are currently checked for capacity constraints
        in station $p$
            //Count the number of train $t_i$ that $t_1$ overlaps with
3.          CountOverlap = 0
4.          For each pair $(t_i, t_j)$ in the set of pairs $\{(t_1, t_2), (t_1, t_3),...(t, t_k),(t_2, t_3),$
            $...(t_{k-1}, t_k)\}$
5.              If $t_i$ and $t_j$ overlap at $p$
                i.e., $a_p^{t_i} \le d_p^{t_j}$ AND $a_p^{t_j} \le d_p^{t_i}$
6.                  Increase CountOverlap by 1
7.          If CountOverlap $\le$ the capacity of $p$ - 1 //Non binding constraint
8.              Remove the set $sp$ $((t_1, t_2), (t_1, t_3),...(t, t_k),(t_2, t_3),...(t_{k-1}, t_k))$
9.              Remove the corresponding capacity constraints(15), (16) and (17)
                (or (15), (16) and (18) if $CAP_p = 2$)
```

The "For" loop in line 1 checks for all points p in the network, which are roughly $|P|$ points. The number of loops of the "For" loop in line 2 is equal to the number N of set of pairs sp $\{(t_1,\ t_2),\ (t_1,\ t_3),...(t,\ t_k),(t_2,\ t_3),...(t_{k-1},\ t_k)\}$ in the set S of set of pairs of trains which are currently checked for capacity constraints. In theory, the cardinality of S might be very big, even exponential to the cardinality of T. However, in practice this number is much fewer, normally less than T (that is one reason why our algorithm is efficient). In all, the complexity of this removing algorithm is normally only $|P| \times |T|$.

Removing the non binding train siding constraints. The idea is to check pairs of trains (t,t') at a point p against which train siding constraints are enforced and to remove those pairs if the corresponding constraints are satisfied but not as strict equality, i.e., the dwelling times of t and t' at p do strictly not overlap. For each point p, those pairs are identified as (t,t') such that variables $\beta_p^{tt'}$ and $\beta_p^{t't}$ are present in the current MILP.

```
1. **For** each point $p$
2.        **For** each pair of trains ($t$, $t'$) in the set S of pairs of trains which are currently
          checked for siding constraints at $p$, i.e., $\beta_p^{tt'}$ and $\beta_p^{t't}$ exists
3.             **If** $a_p^t$ is **strictly greater** than $d_p^{t'}$
               or $a_p^{t'}$ is **strictly greater** than $d_p^t$
4.                  **Remove** $\beta_p^{t't}$ and $\beta_p^{tt'}$ from the set of $\beta$ variables
5.                  **Remove** the corresponding train siding constraints (26) - (28)
```

In the program, we define, e.g., a_p^t is **strictly greater** than $d_p^{t'}$ if $a_p^t \geq d_p^{t'} + \tau$ where τ is 60 minutes, for the same reason as for deadlock constraints.

The "For" loop in line 1 checks for all points p in the network, which are roughly $|P|$ points. The "For" loop in line 2 checks for all pairs (t, t') which are currently checked for siding constraints at p. This number is at most $|T|^2$ but normally much fewer, i.e., in order of $|T|$. In all, the complexity of this removing algorithm is roughly at most $|P| \times |T|^2$ but normally only $|P| \times |T|$.

In summary, we can see that the removing of all non binding constraints requires roughly in the order of $|P| \times |T|$. Therefore, the removing non binding constraint algorithm is dominated by the identifying and adding violated constraint algorithm, which is in the order of $|P| \times |T|^3$.

Chapter 5

Optimization Numerical Results

5.1 Data Instances

In this chapter, we are going to present our optimization results. First we illustrate the efficiency of our algorithm with an example on the evolution of number of constraints and variables when we solve the data instance of 20 trains 1 subdivision. After that, we will investigate some aspects of the train departure times. We then continue by showing the results of differentiating trains by series or by long against regular trains. The chapter concludes with some results showing the effect of adding a double, a siding or both.

We evaluated the performance of the SDT_TS model and algorithm proposed in Chapter 4 on the CPR network between Calgary and Vancouver [ICF$^+$04]. It is essentially a single track railway system, with few double tracks (we consider 15), which is divided into 5 subdivisions. The number of sidings/stations in each subdivision (including the endpoints) is:

Subdivision 1: Calgary - Field - 19 stations or sidings - 2 double tracks

Subdivision 2: Field - Revelstoke - 15 stations or sidings - 3 double tracks

Subdivision 3: Revelstoke - Kamloops - 17 stations or sidings - 5 double tracks

Subdivision 4: Kamloops - Mission - 19 stations or sidings - 3 double tracks

Subdivision 5: Mission - Vancouver - 11 stations or sidings - 2 double tracks

which leads to an overall number of 78 sidings/stations.

In some instances, trains are divided into so-called regular and long trains. The long trains can only take the "long" sidings, i.e., the sidings corresponding to the endpoints of the subdivisions and some few other specifically built sidings. In all, there are 8 ”long” sidings: Calgary, Kamloops, Revelstoke, Field, North Bend, Coquitlam, Chase, Malakwa.

In terms of capacity (number of alternate tracks), we assume 3 alternate tracks at every location that is the endpoint of a subdivision, and 1 otherwise. For both ends of double tracks, we consider them as a station but without any alternate tracks. In other words, a double track cannot be used as a siding and a train cannot be idle on any of the 2 tracks. As a consequence, if two trains need to meet while using a double track, they need to run independently on each of the tracks.

The algorithm SDT_TS was run on 1 to 5 subdivisions with different numbers of trains in order to evaluate the network capacity of the railway system. Indeed, there is a compromise between the number of trains in the railway system and the overall travel times of the trains: if the number of trains is too large, then the overall travel times of the trains increase with significantly increased waiting times, which is undesirable.

We use a set of a 16 to 28 trains, with 78 sidings/stations, (with the same number of trains from Vancouver towards Calgary as from Calgary towards Vancouver unless otherwise indicated) with departure times uniformly distributed over a time period of 8 or 24 hours. Consequently, when the number of trains increases, their departure times are less spaced out.

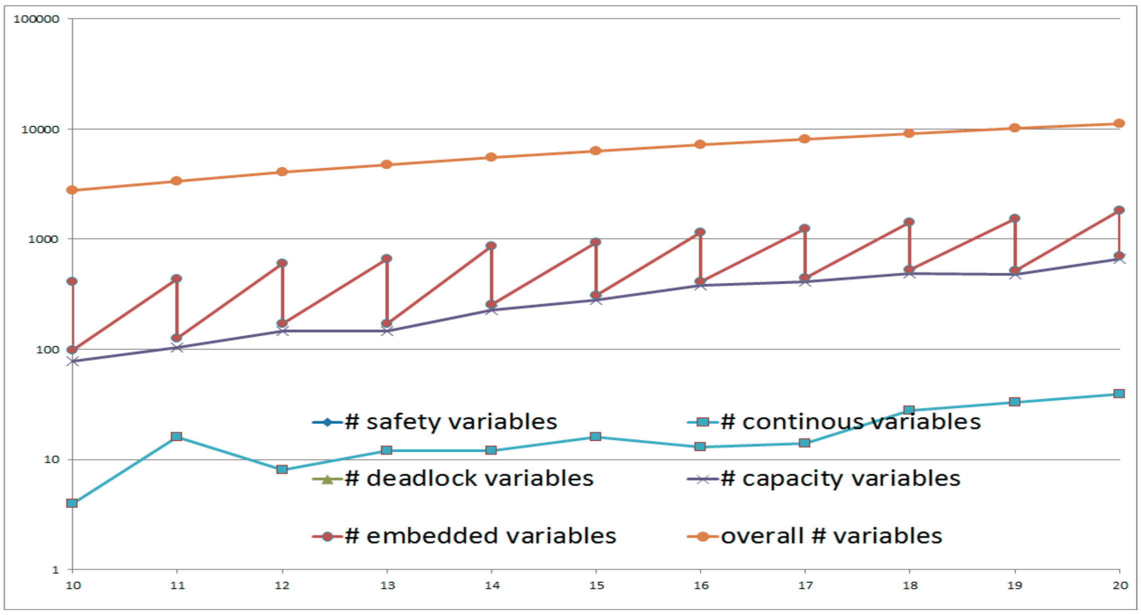
Note that as the set of departure times are not nested in each other when we increase the number of trains, some particular phenomena may occur. For instance, the average travel times may decrease when increasing the number of trains due to more favorable meeting conditions thanks to the train departure times.

5.2 Efficiency of the SDT_TS Algorithm

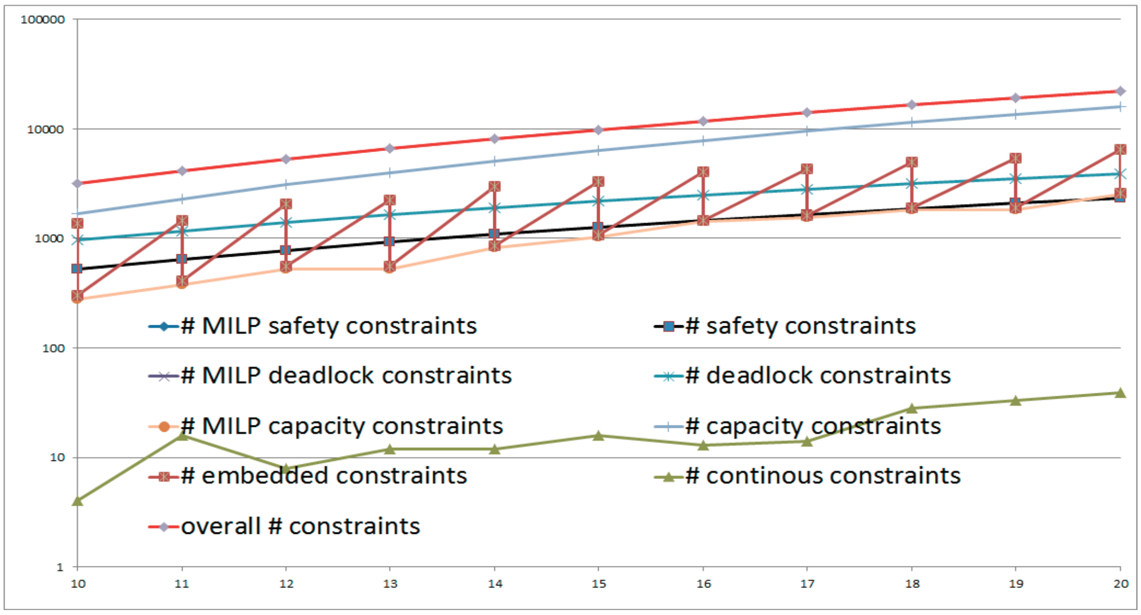
Following the description of the SDT_TS algorithm in Chapter 4, the algorithm iteratively adds trains to be taken into account in the overall train schedule, and alternates between adding violated constraints and removing non binding constraints.

In Figure 2, we plot the number of constraints and variables at each major iteration (i.e., when we add a new train to be taken into account in the schedule) of the SDT_TS algorithm for train scheduling with 20 trains and no flexibility on the departure times. We remove non binding constraints before inserting the constraints (3) to constraints (5) related to an additional train, so we plot the number of variables/constraints before/right after the removal of the non binding constraints for the curves associated with the current total number of embedded constraints. Those plots correspond to the saw-tooth curves in Figure 2. In addition, we add the plots related to the number of constraints/variables for each set of constraints, but plot only the numbers after the removal of the non binding constraints. Both Figures 2(a) and 2(b) are drawn in logarithmic scale. In Figure 2(a), the top curve corresponds to the overall number of variables in the MILP model: we observe that it goes close to ten thousand variables while the number of considered variables barely exceeds 1,000 for 20 trains. The legend indicates the different groups of variables, in correspondence with the constraints in which they appear. In Figure 2(b), the number of constraints follows the same trend as the number of variables.

We observe that the SDT_TS algorithm allows remaining with a highly manageable set of constraints and variables, in spite of the theoretical huge number of variables and constraints of the model, in particular when the number of trains increases. For instance, the complete MILP model contains 11,200 binary variables and 22,240 constraints for 20 trains. As expected, the dominant group of constraints corresponds to the capacity constraints as soon as the number of trains increases, while the safety constraints are much less critical (due to the distribution of train departure times).



(a) Number of Variables



(b) Number of Constraints

Figure 2: Evolution of the number of variables and constraints (1 subdivision, 20 trains)

5.3 Travel Times vs. Number of Trains

We now investigate the network capacity of the Calgary - Vancouver corridor. The goal is to investigate the increase of the travel times from source to destination vs. the number of trains running in the railway network. To do so, we use the following statistics:

- Average travel times (mean - μ, lower bound - LB (the optimal LP solution z^*_{LP}), standard deviation - σ): $\dfrac{\sum\limits_{t \in T} (a^t_{\text{DST}} - d^t_{\text{SRC}})}{|T|}$,
- Average waiting times (mean - μ, standard deviation - σ): $\frac{\sum\limits_{t \in T} \sum\limits_{p \in P} (a^t_p - d^t_p - \text{dw}^t_p)}{|T|}$,
- Number of train meetings/Number of possible meetings,
- Accuracy of the ε-optimal solution (relative value of the difference between the incumbent value and a lower bound):
  - ε^{IN}, the requested accuracy at the outset
  - ε^{OUT}, the obtained accuracy as measured by $\frac{\overline{z}_{\text{ILP}} - z^*_{\text{LP}}}{z^*_{\text{LP}}}$, where $\overline{z}_{\text{ILP}}$ is the optimal ILP solution obtained and z^*_{LP} is the optimal LP solution
- Average earliness/tardiness values on the expected departure times (last two columns in Table 3):
  - $\text{AVG}(d^t_{\text{EARLY}}) = \dfrac{\sum\limits_{t \in T} \text{EARLY}^t_d}{|T|}$
  - $\text{AVG}(d^t_{\text{LATE}}) = \dfrac{\sum\limits_{t \in T} \text{LATE}^t_d}{|T|}$

– $\overline{\text{TRAVEL}}$ = upper bound on the travel time from source to destination stations

All times are in hours	$\lvert T\rvert$	Average travel times			Average waiting times		Number of train meetings	ε^{IN}	ε^{OUT}	$\overline{\text{travel}}$	CPU
		μ	LB	σ	μ	σ	meetings	%	%		h:m
1 subdivision:	16	6:30	6:30	0:35	0:19	0:14	25/64	1	0.0	6:54	0:01
	18	6:33	6:12	0:42	0:21	0:19	35/81	10	5.3	-	0:01
Kamloops	20	6:45	6:10	0:32	0:25	0:20	44/100	15	8.6	-	0:02
$\updownarrow$	22	6:54	6:16	0:44	0:33	0:27	62/121	10	9.3	-	0:11
	24	7:05	6:17	0:53	0:37	0:32	68/144	15	11.3	-	0:17
Revelstoke	26	7:06	6:30	0:33	0:33	0:16	79/144	15	8.4	-	0:30
	28	7:14	6:30	0:40	0:39	0:26	95/196	10	10.0	-	4:31
	30	7:14	6:22	0:49	0:41	0:24	107/225	15	11.9	-	10:35
3 subdivisions:	16	19:26	18:33	1:39	0:38	0:31	58/64	5	4.6	-	0:07
	18	19:38	18:39	1:08	0:42	0:26	74/81	5	5.0	-	0:33
Kamloops	20	20:06	18:36	1:14	1:05	0:41	91/100	15	7.4	-	0:56
$\updownarrow$	22	20:48	18:45	1:00	1:29	0:52	113/121	10	9.8	-	1:33
	24	21:46	18:39	1:42	1:52	1:02	135/144	15	14.3	-	4:44
Calgary	26	22:41	19:27	1:40	2:42	1:41	163/169	15	14.3	-	15:48
	28	22:33	19:28	1:26	2:21	1:28	192/196	15	13.7	-	44:32
5 subdivisions:	16	28:53	27:42	1:39	0:53	0:30	64/64	5	4.1	-	0:13
	18	29:28	27:42	1:50	1:01	0:44	81/81	10	6.0	-	0:28
Vancouver	20	30:05	27:44	1:37	1:22	0:54	100/100	10	7.8	-	1:19
$\updownarrow$	22	30:34	27:43	1:55	1:24	1:07	121/121	10	9.3	-	2:27
	24	30:42	27:43	2:14	1:21	1:01	144/144	15	9.7	-	7:08
Calgary	26	31:31	27:44	1:54	2:14	0:58	169/169	15	12.0	-	33:03
	28	31:35	27:42	1:44	1:52	1:08	196/196	15	12.3	-	69:47

Table 1: Travel times vs. network load - No flexibility on departure times - Time period 24h

Statistics are reported for 1, 3 and 5 subdivisions, i.e., for 14, 50 and 78 sidings/stations respectively. The requested precision ε^{IN} varies between 10% and 15% in order not to spend too much time solving the first MILP models. For small

All times are in hours	$\|T\|$	Average travel times			Average waiting times		Number of train meetings	ε^{IN}	ε^{OUT}	$\overline{\text{travel}}$	CPU h:m
		μ	LB	σ	μ	σ					
1 subdivision:	16	7.52	7:08	1:03	1:14	0:58	62/64	10	9.3	-	0:32
Revelstoke $\leftrightarrow$	18	7:59	7:19	1:07	1:21	1:01	78/81	10	8.4	-	32:27
Kamloops	20										
3 subdivisions:	16	19:07	18:13	1:14	0:29	0:25	64/64	5	3.8	-	0:24
Kamloops $\leftrightarrow$	18	19:25	18:13	1:04	0:39	0:41	81/81	15	6.2	-	1:11
Calgary	20	20:30	18:14	1:27	0:36	1:00	100/100	15	11.1	27:00	2:06
5 subdivisions:	16	29:05	27:29	1:32	0:42	0:35	64/64	15	5.5	-	1:31
Calgary $\leftrightarrow$	18	30:07	27:30	2:11	1:15	1:01	81/81	15	8.7	-	3:36
Vancouver	20	30:52	27:33	2:13	1:43	1:03	100/100	15	10.7	35:00	13:40

Table 2: Travel times vs. network load - No flexibility on departure times - Time period 8h

instances, ε^{IN} can be put as small as 1%. As can be observed in the column entitled ε^{OUT}, the obtained precision is often much better than the requested one. However, the obtained precision varies with the number of trains and we observed that the average times are not always strictly increasing when the number of trains is increasing for a given number of subdivisions. This is due to the side effect of the departure times which are not optimized, but randomly generated as described in Section 5.1. The optimal value is however guaranteed to lie in the interval $\left[\text{LB}, \frac{\sum_{t \in T}(a^t_{\text{DST}} - d^t_{\text{SRC}})}{|T|}\right]$, and there is a clear trend of increasing LB and average travel times values. The increase of the average travel times is consistent with the increase of the average waiting times due to train meetings, as expected. The fact that the lower bound is not strictly increasing can be explained by the distribution of the departure times, which create disparities in the travel times due to more/less favorable departure times with train meets and waiting times at sidings.

In Table 2, we run the same experiments as in Table 1, except that the period of operations is limited to 8 hours. We can see that for 1 subdivision, the average travelling times increase significantly compared to when we distribute over a 24 hour time period. For example, the travelling time for 16 trains increases from 6h30m in the 24 hour period case to 7h52m in the 8 hour period case, i.e., roughly 20 percent. For 18 trains, the increase is from 6h20m to 7h59m, i.e., roughly 25 percent. Notice also the significant increase in number of trains meetings from 25/64 to 62/64 and from 32/81 to 78/81 in case of 16 and 18 trains, respectively. The increase might be due to the fact that the 8 hour period is a shorter period and trains meet more trains and wait longer in general before reaching their destination. In short, Table 1 and Table 2 together give us an idea of the capacity of the network against the density of trains as well as the impact of operating hours.

In order to illustrate the train scheduling, we represented one of them with the so-called string graph for an instance with 5 subdivisions, i.e., the entire Vancouver - Calgary corridor with 20 trains. String graphs are used to display spatial and temporal information of track occupancy: the vertical axis contains the distances between the Eastern and Western stations (or the location of the intermediate sidings/stations) while the horizontal axis is a time axis. It allows a visualization of the track occupancy and the verification of the capacity constraints (number of alternate tracks at sidings/stations).

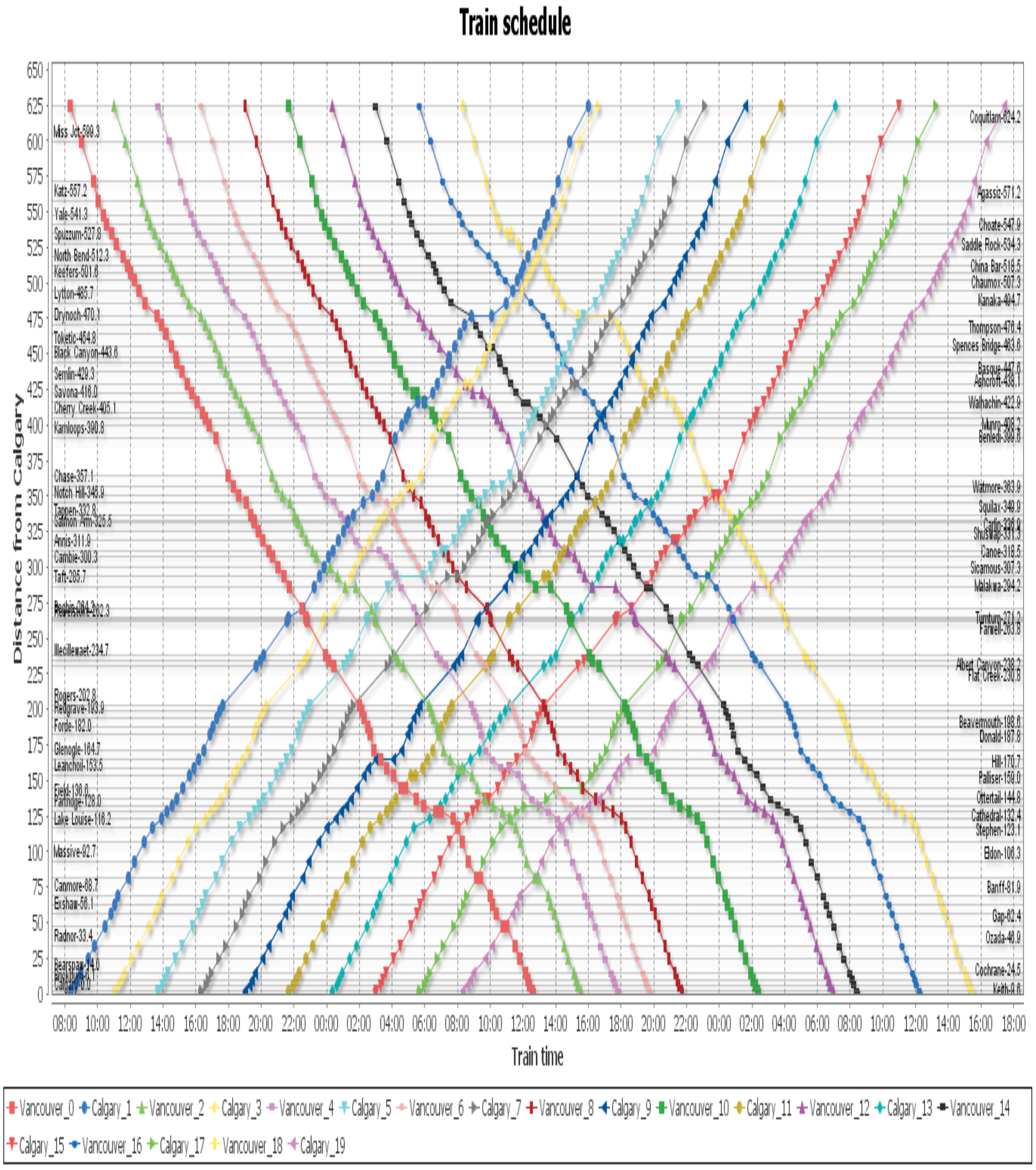


Figure 3: String graph (Vancouver - Calgary - 20 trains - 5 subdivisions)

5.4 Flexible Departure Times

The next set of experiments has been conducted in order to investigate the impact of setting an offset on the departure times for some trains in order to reduce the average travel times. Results are reported in Table 3 on one subdivision. We consider two offsets, the first one where the departure time of train t lies in [d_t−0:30, d_t+0:05], and the second one where the offset interval is larger: [d_t−0:30, d_t+0:30]. In the last two columns, we report the average earliness and tardiness of the trains with respect to the planned departure times, and the numbers between parenthesis correspond to the number of trains which are early and late, respectively. Note that the experiments have been conducted with $\varepsilon^{\text{IN}} = 10$ %, different from the values used in Tables 1 and 2.

We observe that a wise delay (early or late) of 1 to 14 minutes for almost all of the trains can lead to up to a 24 minute difference in the average travel times, see, e.g., the case of 20 trains with and without any offset on the departure times. The improvement is more prominent when we transit from the instances where train must depart on time to the instances where train can depart up to 30 minutes early and 30 minutes late. Notice the numbers of train meetings also decrease and so do the standard deviations, i.e., trains are scheduled somewhat more equal.

Those experiments illustrate the necessity to wisely select the departure times: they have a significant impact on the travel times. Sometimes, leaving 5 mins later or earlier can lead to a huge difference for the travel time. The more so, when the

# Trains	Average travel times		Number of train meetings	ε^{OUT}	$\overline{\text{travel}}$	$\frac{\sum_{t \in T} \text{EARLY}_d^t}{\lvert T \rvert}$	$\frac{\sum_{t \in T} \text{LATE}_d^t}{\lvert T \rvert}$
	μ	σ					
Trains depart on planned departure times							
16	7:07	1:04	26/64	10.0	-	0.	0.
18	7:16	1:17	33/81	8.8	-	0.	0.
20	7:27	0:52	45/100	9.4	-	0.	0.
Trains can be up to 30 mins early and 5 mins late with respect to planned departure times							
16	6:51	0:55	25/64	9.6	7:24	0:15 (12)	0:01 (3)
18	6:57	0:55	32/81	9.2	-	0:12 (11)	0:01 (7)
20	7:02	0:45	40/100	8.4	-	0:14 (16)	0:01 (4)
Trains can be up to 30 mins early and 30 mins late with respect to planned departure times							
16	6:47	0:33	23/64	9.1	7:30	0:10 (7)	0:14 (9)
18	6:51	0:49	32/81	8.4	-	0:05 (7)	0:10 (11)
20	6:53	0:29	38/100	9.1	-	0:10 (12)	0:07 (8)

Table 3: Travel times vs. network load - Some flexibility around the planned departure times

number of trains increases.

5.5 Long vs. Regular Trains

We now investigate the impact of longer trains on the average travel times. Running longer trains comes with the idea of reducing the number of trains. However, not only it reduces the number of locations where two long trains can meet, but it also forces shorter trains to more often take the sidings, in locations where the length of the sidings cannot hold a long train. Indeed, in our set of experiments on 5 subdivisions, out of the 78 sidings, only 8 can be used by long trains. Long trains mean trains of length 2 miles and higher, while regular trains means trains of length 1 mile, or smaller. Experiments are conducted with different percentage of long trains for the set of 20 trains. In the first experiment with 2 long trains, they run in opposite directions, while in the 3 other experiments, all long trains run in the same direction.

		Long trains				Regular trains										
$\|T\|$	# Long Trains	Average travel times		Average waiting times		Average travel times		Average waiting times		Number of train meetings			ε^{OUT}	$\overline{\text{travel}}$		CPU h:m
		μ	σ	μ	σ	μ	σ	μ	σ	SS	SL	LL		S	L	
20	2	29:18	1:07	0:42	0:27	30:03	1:58	1:24	0:44	81	18	1	7.5	-	-	1:16
	4	28:58	0:38	0:03	0:06	30:15	1:24	1:36	0:46	60	40	0	7.2	-	30	1:15
	6	29:32	0:21	0:00	0:00	30:42	1:52	1:43	0:56	40	60	0	8.2	-	30	1:44
	10	29:16	0:12	0:09	0:10	30:21	1:50	2:25	1:15	0	100	0	5.3	-	30	0:25

Table 4: Travel times vs. network load - No flexibility on departure times - Time period 24h

We can see that in all instances the long trains have quite small standard deviation, i.e., a more uniform travelling time compared to regular trains. This is due to the fact that long trains are prioritized over regular and take up less often the sidings. Another interesting point is that the travelling time of the long trains does not vary too much (at most 34 minutes) when we increase the percentage of the long trains.

5.6 Siding Usage

Below we plot a histogram showing the siding usage in three scenarios to investigate how the siding usages differ in 3 scenarios. In the upper pane, each set of 3 bars (one bar for each scenario) corresponds to a siding (in 5 subdivisions) and the height of the bar is proportionate to the average waiting time of all trains at this siding. Similarly, in the lower pane, each set of 3 bars (one bar for each scenario) corresponds to a siding and the height of the bar is proportionate to the number of times trains take this siding. In both panes, if at a siding, there is no trains stop, we do not plot any bar for that siding.

Scenario 1: 20 trains, all regular (20-0L-20S)

Scenario 2: 18 trains, 2 long, 16 regular (18-2L-16S)

Scenario 3: 16 trains, 4 long, 12 regular (16-14L-12S)

We can observe that, on the three considered scenarios, about one third of the sidings are not used. Of course, it may vary with the selection of the departure times. Indeed, the most worthy experiment would be to investigate the siding usage with an optimized selection of the train departure times, which goes beyond the scope of this thesis. While more experiments are needed, it may be the case that a reduction of well selected sidings would not impact significantly the average travelling times of the trains.

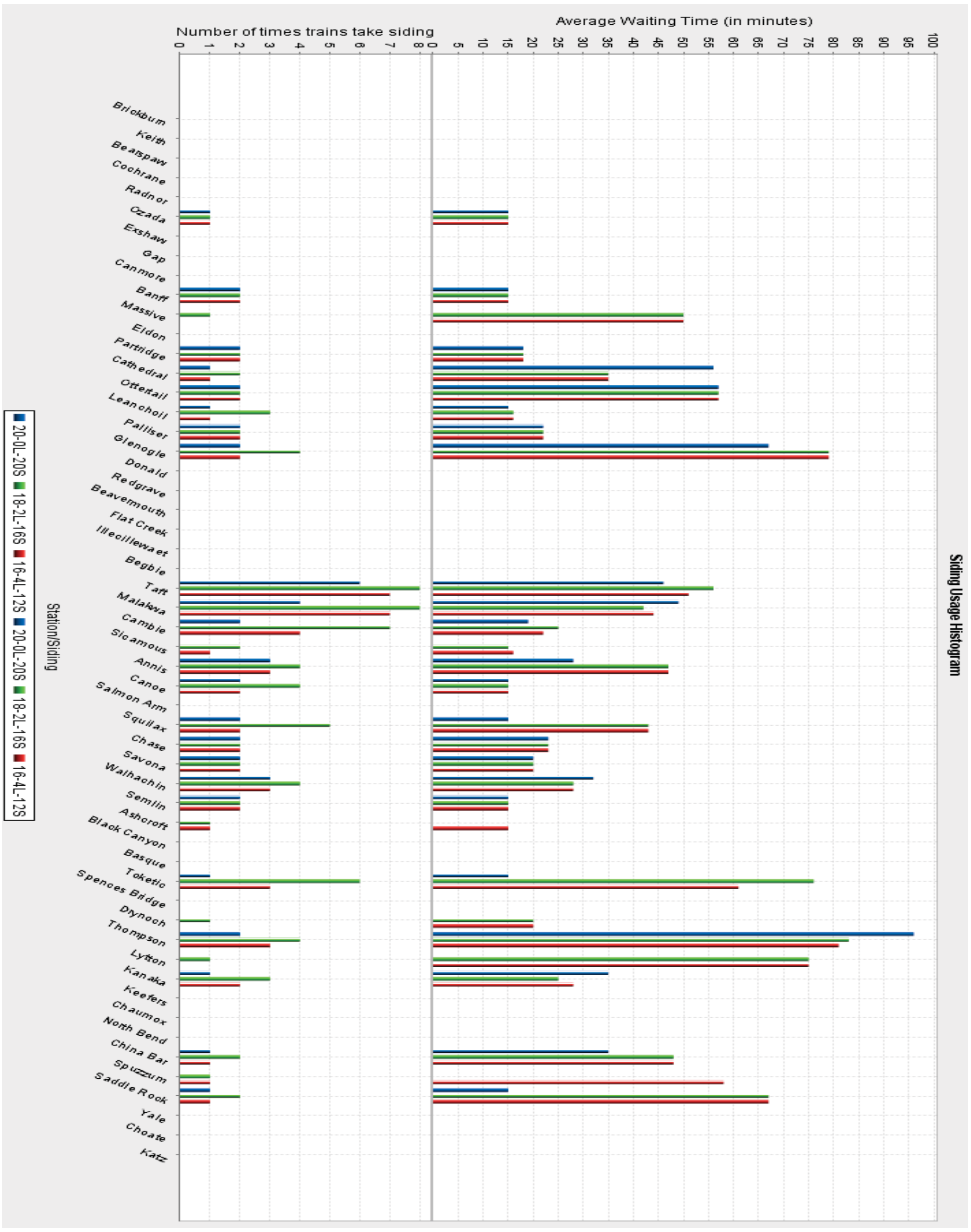


Figure 4: Siding usage histogram

5.7 Prioritizing Trains

Many railway customers are willing to pay more in order to have products arrive at the destination as soon as possible. Therefore, the ability to prioritize train travelling times is one important factor to the profitability of freight train company. In this section, distribution of train priorities will be investigated throughout differentiated limits on the average travel times. The results are for 5 subdivisions.

$\vert T\vert$	Train series				$\overline{\text{travel}}$				Average traveling times			
	100	200	400	800	100	200	400	800	100	200	400	800
20	20	0	0	0	-	-	-	-	30:05	-	-	-
	16	4	0	0	30	-	-	-	29:01	29:40	-	-
	10	5	5	0	30	31	-	-	29:10	29:16	29:53	-
	6	4	5	5	29	31	32	-	28:38	29:31	29:04	29:36
	2	4	7	7	28:30	31	31	-	28:24	28:56	29:15	30:31

$\vert T\vert$	Train series				Average waiting times				ε^{OUT}	CPU
	100	200	400	800	100	200	400	800		
20	20	0	0	0	1:22	-	-	-	7.8	1:19
	16	4	0	0	0:51	1:19	-	-	4.4	1:20
	10	5	5	0	0:55	0:51	1:11	-	5.3	1:32
	6	4	5	5	0:40	1:08	0:54	0:53	4.0	1:41
	2	4	7	7	0:47	0:57	1:05	1:19	4.2	1:12

Table 5: Train priorities

We consider 4 different series of trains, called 100, 200, 400 and 800 following the terminology of CPR. Table 5 shows that we can prioritize the trains according to its series by putting different upper bounds on each one. As long as the bounds are reasonable, we can shorten the travelling times of some trains while keeping the travelling times of other trains not too long. For example, with different bounds, we can reduce the travelling time of 100 series trains (the most prioritized ones) from 30h05m to 28h24m, i.e., by roughly 2 hours. Meanwhile, the longest travelling time, that of the least prioritized one is increased only slightly, from 29h36m to 30h31m, i.e., less than one hour. Indeed, we can reduce further the travelling time of the 100 series if the speed limit allows, and again we can possibly further optimize the selection of the departure times.

5.8 Impact of Additional Sidings/Double Tracks

In this set of experiments, we study the effects of adding one more double track, adding one more siding and adding simultaneously one double track and one siding.

Scenario 1: only single tracks

Scenario 2: current 78 CPR single sidings and 15 double tracks

Scenario 3: Scenario 2 + 1 double track (Three Valley - Taft in subdivision Revelstoke-Kamloops)

Scenario 4: Scenario 2 + 1 additional siding (Three Valley in Revelstoke - Kamloops)

Scenario 5: Scenario 2 + 1 additional double track (Three Valley - Taft in Revelstoke - Kamloops) + 1 additional siding (Cisco in Kamloops - North Bend)

Scenarios	$\|T\|$	Average travel times μ	Average waiting times μ	Number of train meetings	ε^{IN}	ε^{OUT}	$\overline{\text{travel}}$	CPU h:m
1	16	31:18	2:32	64/64	10.0	9.0	-	4:46
	20							
2	16	28:53	0:53	64/64	5.0	4.1	-	0:13
	20	30:05	1:22	100/100	10.0	7.8	-	1:19
3	16	28:02	0:27	64/64	1.0	0.0	-	3:29
	20	28:56	0:49	100/100	10.0	4.4	-	0:58
4	16	28:46	0:50	64/64	5.0	3.8	-	0:13
	20	29:47	1:05	100/100	10.0	6.9	-	1:42
5	16	28:07	0:27	64/64	1.0	0.0	-	3.43
	20	28:54	0:50	100/100	10.0	4.1	-	1:40

Table 6: Additional double track/siding (5 subdivisions)

Note that, within the limit of 72 hours, we could not solve the instance with

20 trains in the first scenario, i.e., no double tracks and only single track. With 16 trains, we do get the results but need much more computing times. It is clear that the average travelling time in scenario 1 is much longer than in the other scenarios, with the difference ranging from 2h15m to almost 3h. Therefore, double tracks seems to play a big role in reducing the travelling time (and also arguably the computing time). However, their location need to be carefully planned.

For the current network and for the given train departure times, it turns out that the effect of adding a new double tracks is better than adding a new siding at the selected location. For example, the average running time of 20 trains in the current network, i.e., 30h05m is reduced to 28h56m when we add a new double track (2h09m difference) and to 29h47m when we add a new siding (only 18m difference). Also for 20 trains, when we add one more siding on top of adding one double track, the decrease in average travelling time is insignificant, only 2m (from 28h56m to 28h54m). Again, the selection of the location of those additions may play a crucial role, but this goes beyond the scope of this thesis

Chapter 6

Simulation Results

6.1 Generalities

The goal of the simulation is to test and verify various scheduling scenarios to better understand the strengths and weaknesses of each scenario under realistic operating conditions. The animation capability of most discrete system simulation packages (Arena by Rockwell [RS02, KSS07] in our case) further assist decision makers to observe system dynamics in different settings. Consequently, capacity limitations in certain delays and the reasons for delays in specific corridors can be identified. Simulation also serves as a benchmark against optimization results.

In order to reach the above mentioned goals, we developed a train simulation model based on the data received from CPR. The developed simulation model includes most factors that influence the train operations. Although it is possible to incorporate them into the model, we did not include service disruptions due to unexpected events (flood, rain etc) since we did not have sufficient data. Rather, we focused on modeling track usage, safety distances, station capacities, siding capacity, impact of the slowing down and speeding up times, needed for full stops of trains in stations and sidings, dwelling operations and finally deadlock avoidance. Consequently, a train simulation model of CPR's Vancouver-Calgary corridor was built within the Arena simulation software.

The remainder of this chapter is organized as follows. First, the details of the developed train scheduling simulator are given in Section 6.2. Therein, we describe train scheduling parameters in Section 6.2.1 and then scheduling constraints in Section 6.2.2. Next, we describe two conditions to handle the train deadlock situation in Section 6.2.3 and then the simulation workflow in Section 6.2.4. Finally the numerical results obtained from the simulation are presented in Section 6.3.

Arena Simulation Software

Our freight train scheduling simulation is implemented in Arena, a very popular simulation software in the industry and currently used at CPR. This software has some built-in features that serve quite well railway network simulation. One drawback of Arena is that incorporating an algorithm into the simulation model takes quite a lot of effort. A more flexible simulation tool is OMNET++, which allows users to easily customize available features by adding new features to the tool as needed. However, this software is primarily designed for communication network simulation which bears little resemblance with train simulation. Therefore in this thesis, we used Arena as our simulation tool.

To produce train schedules, we use the same data instances as for the optimization model so that we could compare in a meaningful way the results of optimization and simulation data. In other words, we use the same network data (in terms of the number of stations and sidings, capacity of stations and sidings, and the speed limits on each segment) and the same train data (in terms of number of trains, routes of trains and departure times).

6.2 Simulating Train Scheduling

6.2.1 Train Scheduling Parameters

Railway Network Parameters

The following parameters are used in the simulation:

Network parameters

- Maximal speed on each segment
- Segment length
- Safety distance
- Number of alternate tracks at each station/sidings
- Track length.

Train Parameters

For each train, they includes:

- Train series (for identification and for priority decision)
- Origin
- Destination
- Expected departure time at origin

- Stations where train operations (crew change/fuel) are performed

- Dwell time at these stations.

Since we focus on the Vancouver-Calgary corridor, we will consider only trains that operate exclusively or partly in this corridor. If a train operates partly in the corridor, we only consider the part of the train's route that is inside the corridor, therefore we need to know not only the whole route of the train but also the entry and exit stations of the train in the corridor.

- Train series

- Origin

- Destination

- Expected departure time at origin

- Expected arrival time at destination

- Entry station in the Vancouver-Calgary corridor

- Expected departure time at entry station

- Exit station in Vancouver-Calgary corridor

- Expected arrival time at exit station.

6.2.2 Scheduling Constraints

Conflict Constraints. For two trains t and t' traveling on the same segment s in opposite directions, at most one train can be on s at any given time. This is the same as the deadlock constraint in the optimization model. However, in this chapter we use the term deadlock in a different context (more details to follow) so we apply the term conflict constraint here.

Safety Constraints. Two trains t and t' traveling on the same segment s in the same direction must maintain a sufficient safety distance (headway) while traveling on s. These constraints are basically the same as the safety constraints of the optimization model.

Capacity Constraints. For each station/sidings s, there is only a limited number of tracks. At any time, the number of trains dwelling at each station should not exceed the number of tracks. Again, they are similar to the capacity constraints of the optimization model.

6.2.3 Deadlock Avoidance Constraints

In the train scheduling context, deadlock is the situation in which some trains are blocking each other (due to constraints described above) and therefore no train can be moved. Although algorithms have been proposed for deadlock avoidance, it is quite difficult to use them in a simulator environment in order to guarantee a 100%

deadlock avoidance. First, as deadlock avoidance is a NP-hard problem [LDL04], there is no algorithm that can give optimal solution in polynomial time. So the algorithms which are proposed in the literature are approximation algorithms at best. Second, in principle, in order to guarantee 100 % deadlock free, whenever there is an event such as a train departure from a station or arrival at a station,... we have to look into the whole network or at least the whole part of the network that is affected by this event to detect the potential deadlocks. When a simulation runs, there are many events and detecting all potential deadlocks after each event requires a quite big amount of time. That is clearly very costly in terms of computing times even for a moderate network size. So, in practice, we would like an algorithm that can avoid the deadlock in most of the cases but not too costly in terms of computing times. In the next paragraph, we propose two simple conditions. The first one is a sufficient condition in order to identify a deadlock situation. The second one is a sufficient condition that guarantees a deadlock free environment.

Deadlock avoidance

Sufficient condition for deadlock detection Let us consider the situation depict in Figure 5. We consider a train t waiting at p' to go to p. If there is another train currently going from p to p', the train t cannot depart. Let us then now assume that there is no train from p to p', but there are trains going from p' to p and trains going from p'' to p (trains which are currently on segment $[p, p'']$ do not affect the capacity of p so we do not consider them here).

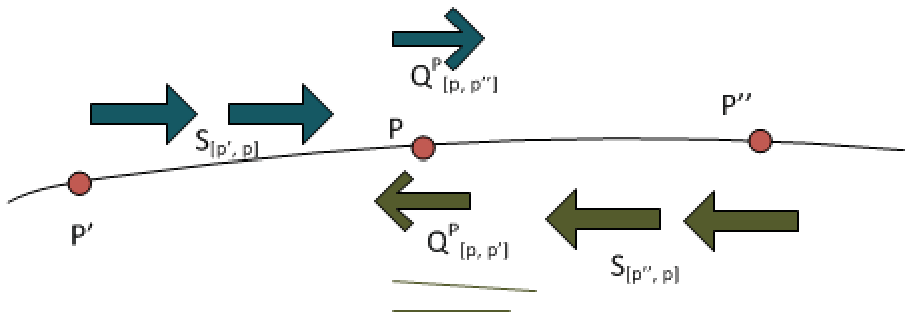


Figure 5: Illustration of the deadlock condition

Let:

C_p: Number of tracks at p

$Q^p_{[p,p']}$: Number of trains currently at p waiting to go to p'

$Q^p_{[p,p'']}$: Number of trains currently at p waiting to go to p''

$S_{[p',p]}, S_{[p'',p]}$: Number of trains currently on segments $[p', p]$ and $[p'', p]$, respectively.

Then we have the following sufficient condition for a deadlock to arise on segment $[p, p]$:

$$\text{if } (Q^p_{[p,p']} + Q^p_{[p,p'']} + \min\left\{(1 - \min\{1, Q^p_{[p,p']}\} + S_{[p',p]}, (1 - \min\{1, Q^p_{[p,p'']}\}) + S_{[p'',p]}\right\} > C_p) \tag{32}$$

then there will be deadlock on $[p, p']$.

The idea behind this condition is as follows. Clearly the terms $Q^p_{[p,p']} + Q^p_{[p,p'']}$ account for the trains currently occupying the tracks at the station. In order for all trains to pass through p, either $(S_{[p',p]})$ trains from the West or $(S_{[p'',p]})$ trains from the East have to reach p first. So, we have at least the minimum of them must be at p before other trains can pass p. Assuming the minimum of the two numbers is $S_{[p',p]}$, i.e., the number of trains which comes from the West, we consider two cases:

- if $Q^p_{[p,p']} \geq 1$ then after all ongoing trains from the West have arrived at p, even there is no unoccupied track, we do not have a blocking case as one train among ($Q^p_{[p,p']}$) can depart and then the other waiting trains or the ongoing trains from the East can go through p one at a time.

- if $Q^p_{[p,p']} = 0$, then after all ongoing trains from the West have reached at p, if there is no unoccupied track, we have a blocking case.

So we have to account for this fact by introducing the term $1 - \min\{1, Q^p_{[p,p']}\}$ which is equal to 0 if $Q^p_{[p,p']} \geq 1$ and to 1 if $Q^p_{[p,p']} = 0$.

Notice that if condition (32) satisfied, a deadlock will occur. Consequently, a train should not leave a station if condition (32) is satisfied. Therefore, when we consider letting a train depart, we will check for condition (32) (which is just a few computations) and release the train only if the condition (32) does not hold.

Condition (32) works well to detect the deadlocks when the number of trains in the network is small. Indeed, its advantage is that it utilizes quite well the network. The condition (32) though, is clearly not sufficient in practice to efficiently detect all deadlocks, in particular when the network traffic increases. Next, we are going to present another condition that if satisfied, guarantees a deadlock free environment.

Sufficient condition for deadlock avoidance. We would like to benefit from the fact that the network of CP is mostly single track and with very few mesh stations. For the Vancouver-Calgary corridor, we reserved half of the capacity of the stations/sidings for each direction (eastbound from Vancouver to Calgary and westbound from Calgary to Vancouver) under the assumption (which is true in practice) that the number of trains in both directions is very similar. This solution prevents traffic from one direction from affecting too much on the traffic from the other direction.

Consider a westbound train t that is dwelling at station p_1 and heading to p_2 and then p_3, we have to decide whether we let the train t to move from p_1 to p_2. Our condition states that: we only let t depart from p_1 if the number of westbound trains at p_2 is less than half of the capacity of p_2. In other words, the sum of the number of trains currently on segment $[p_1, p_2]$ and the number of trains currently dwelling at p_2 and heading to p_3, is less than half of the number of tracks (main track and alternate tracks) at p_2. Below is the condition:

$$\begin{aligned} &\text{Number of trains on the segment } [p_1, p_2] + \\ &\text{number of train at } p_2 \text{ that heading to } p_3 < \frac{1}{2}(\text{Capacity of } p_2) \end{aligned} \qquad (33)$$

Condition (33) has the advantage of being quite simple to implement and in practice does eliminate deadlock situations. However, it comes at the cost of being too conservative, i.e., trains normally have to wait for longer times than expected before they can move. The reason is due to the fact that condition (33) does not allow two trains in the same direction to be on the same track at the same time (even if they do maintain a safety distance). On the contrary, condition (32) does allow this situation. In practice, applying condition (33) may lead to a quite strong underutilization of the network in particular when the network traffic is high.

6.2.4 Scheduling Simulation Workflow

Figure 6 describes the workflow of the simulation. A train is created as an entity at the origin station. The time and the station at which it is created are read from an Excel file. Before leaving any station, it is put into a queue corresponding to the segment connecting the current station and the next station. The queue is a "wait for condition" queue which ensures that no train in the opposite direction is currently travelling on the segment and that the safety distance is maintained. The condition also ensures that a blocking case cannot occur for at least the next station. Only if that condition is satisfied then the train, i.e., the entity is released from the queue.

Whenever a train leaves or enters a station, the network state variables are updated accordingly. The variables include the number of trains waiting at each station, the number of trains currently running on each segment, the segments that are "cleared" (i.e., trains can run on this segment) or not.

When a train is first created at its origin or when a train reaches a station, after performing all the necessary dwelling operations (whose durations are generated randomly), it is eligible to continue its route. Then, we will decide whether the train should move based on some criteria (further details on these criteria are given in the next section).

In parallel to coordinate the trains that currently exist in the network, the simulator also creates new trains into the system according to their planned departure times.

The simulator stops when one of the following two conditions is satisfied:

- All trains are created and all trains have reached their destination,
- A deadlock is detected.

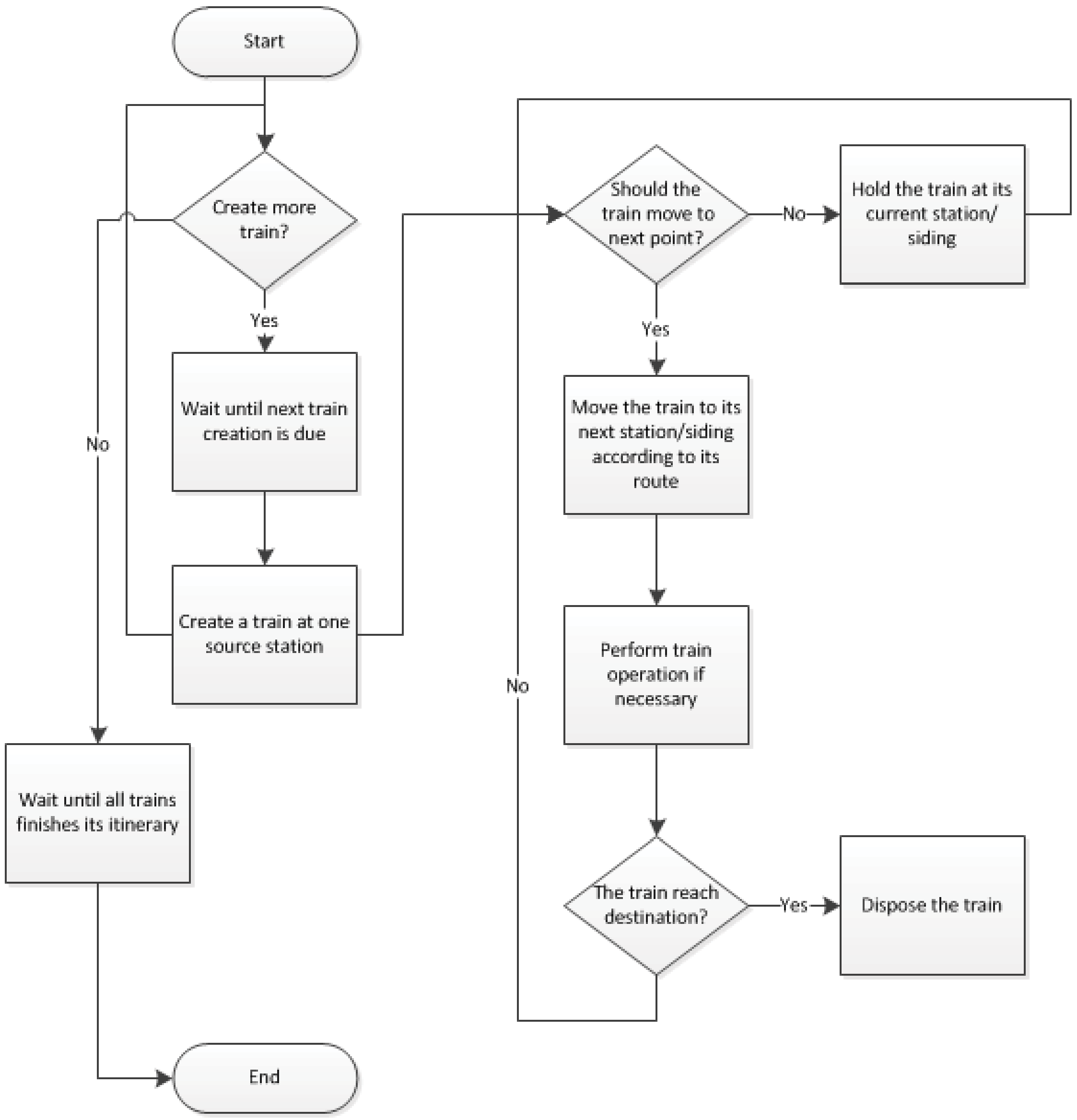

Start
Create more train?
Yes
No
Wait until next train creation is due
Create a train at one source station
Wait until all trains finishes its itinerary
End
Should the train move to next point?
No
Hold the train at its current station/ siding
Yes
Move the train to its next station/siding according to its route
Perform train operation if necessary
No
The train reach destination?
Yes
Dispose the train

Figure 6: Train scheduling simulation workflow

Criteria for moving decision

After a train completes all its required operations (load/unload, crew change,...), it now becomes eligible to move to the next location on its route. However, it is allowed to move only if it complies to some criteria. These criteria are needed so that all constraints such as no-conflict constraints, safety constraints,... are satisfied and so that deadlock occurrences are reduced as much as possible. It is also needed to prioritize trains when more than one train are competing for an available track. The criteria are as follows (listed by the order in which they are considered):

- No train currently running in opposite direction on the segment (no conflict constraint)
- Safety distance with the previous trains is maintained (safety constraint)
- Moving this train will not cause deadlock (in our simulation, we use the condition 33 described in Section 6.2.3
- If two trains are competing for an available track at the next point, only the train with highest priority will be moved, the other will be held in its current location.

6.3 Comparative Results

In this section, we show some results of the simulation and compare them against those of the optimization SDT_TS algorithm developed in Chapter 4. Shown results are for the whole corridor Vancouver-Calgary so that we can have an idea on how much different they might be on a large scale.

We deliberately do not include the running times of the simulation because they are often very short (less than 1 minute) for all instances. Whereas, for the optimization algorithm, as can be observed in Chapter 5, the running times vary from 1 min for small instances (16 or 18 trains) up to 68 hrs for large instances (e.g., 28 trains). That is indeed an advantage of the simulation over the optimization approach. We also do not show the ε accuracy information because it is not applicable to simulation.

We can see that, for each instance, the differences in the average travelling times are quite huge, ranging from about 3 hours to almost 4 hours. This can probably be explained by the fact that we use a very conservative approach, mentioned in Section 6.2.3 above, in order to avoid deadlocks.

The standard deviation are quite different for the simulation and optimization algorithm, from 1 to 2 hours. We can see that the optimization SDT_TS algorithm produces more uniform travelling times than the simulation does, which is better for train operations.

The conservative approach for deadlock avoidance has also an effect on reducing the number of train meetings. Due to the conservative condition, there are trains which have to wait longer until they can depart, e.g., from Calgary. And before their departure, other trains in the opposite direction, e.g., Vancouver may arrive.

All times are in hours	$\lvert T\rvert$	Average travel times		Average waiting times		Number of train meets
		μ	σ	μ	σ	
	OPTIMIZATION					
	16	28:53	1:39	0:53	0:30	64/64
	18	29:28	1:50	1:01	0:44	81/81
	20	30:05	1:37	1:22	0:54	100/100
	22	30:34	1:55	1:24	1:07	225/225
5 subdivisions:	24	30:42	2:14	1:21	1:01	144/144
	26	31:31	1:54	1:21	1:01	169/169
	28	31:35	1:44	1:52	1:08	196/196
	30	N/A	N/A	N/A	N/A	N/A
	SIMULATION					
Vancouver	16	32:41	3:40	1:40	1:31	45/64
	18	32:22	3:13	1:54	1:13	56/81
	20	33:04	2:33	1:54	1:13	68/100
↕	22	32:50	2:00	1:08	3:09	80/121
	24	33:16	3:33	2:22	1:14	93/144
	26	34:02	3:50	1:12	2:14	118/169
Calgary	28	34:52	4:06	3:49	1:26	134/196
	30	34:51	4:07	4:22	1:27	159/225

Table 7: Simulation vs Optimization - Travel times vs. network load - Time period 24h

Therefore, these trains do not meet inside the network and then the number of train meets decreases. On the contrary, the optimization SDT_TS algorithm manages to arrange all train meets at the sidings between Vancouver and Calgary while not forcing trains to wait longer. Therefore, the number of train meets in the optimization are always higher than that of the simulation.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Following the scarce resources of freight train companies, efficient scheduling tools are required in order to optimize the track usage and minimize the train travel times. In this study, we propose an enhanced optimization model which includes the double tracks, mesh topology, siding/station capacities and management (deciding on which trains take sidings). We also propose an exact algorithm which allows a proper management of the constraints and variables in order to remain scalable even for large data instances. Indeed, the newly proposed SDT_TS algorithm is able to solve accurately instances for up to 78 siding/stations and 28 trains.

Using the newly proposed SDT_TS model, we studied various aspects of the freight train scheduling, namely the effect of increasing the number of trains over a certain time period, the potential of prioritized train series or the effect of adding new siding and double tracks. We also show that allowing some flexibility on the train departures does improve the overall quality of the schedules with less train meets, and shorter and more uniform travelling times. Our investigations on long trains show that, in order to take advantage of reducing the number of trains, the addition of longer trains should be done jointly with selecting wisely the departure times. In all, the SDT_TS model is shown to be quite flexible and could well be served to analyze different aspects of the current railway system as well as how to make better use of it.

In terms of solution techniques, our dynamic row/column generation algorithm is shown to be quite efficient. Given the comprehensiveness of the model and the size of data instances, except for very large instances, the program is still able to produce results in reasonable amounts of running times (often in minutes, which is quite enough for an analyzing or planning tool). We have shown that with a good model and a proper management of the constraints and variables, we can use an off-the-shelf solver to tackle a hard problem, for data instances of practical sizes, up to a desired optimality.

An interesting observation is that the running times seem to be quite proportionate to the difficulty of solving a given instance. For example, given the same network of 5 subdivisions and 78 sidings and the same set of 16 trains, the computing times range from 13 minutes (when there are some double tracks in the network) to nearly 5 hours (when there are only single tracks with less possibilities for trains to cross each other).

Another evidence of the dominant role of the solution difficulty with respect to the size of the data instances in computing times can be found in Table 2 in Chapter 5. Therein, trains are distributed over a 8h period, the running time of the instance of 18 trains, 1 subdivision is 32h27m whereas that of the instance with 18 trains 3 subdivisions (almost triple the size of the network) is only 1h11m. Intuitively, it might be explained by the fact that scheduling the same number of trains in the same short time period (in which several conflicts might happen) is much more difficult if we have a much smaller network, i.e., much less space to arrange for the train meetings.

7.2 Future Work

Further analysis using SDT_TS model

In this study, we have shown the efficiency of our SDT_TS model. Based on that model, we can carry out further analysis of the current CPR train network. For example, we could identify the best siding locations and lengths in order to reduce further the train travel times. In terms of departure times, for the time being, the train departure times are uniformly distributed over a certain time period. In the future, we will study different departure time patterns to find out which one is the best. We will also investigate further the trains' efficiency, i.e., maximizing the hauling with the minimum number of trains, under the assumption of a judicious selection of the train departure times.

Further testing

Current, although the SDT_TS and algorithm are valid for mesh networks we only tested them on a single line railway network from Vancouver to Calgary in this thesis. In the future, we will conduct experiments on a mesh network. We will then analyze the numerical results to verify the efficiency of our SDT_TS with a mesh network.

Enhancing the current model

The current model can solve up to 28 trains on 5 subdivisions, which is far more than what previous models solved in the literature. However, the SDT_TS algorithm can still be further improved so that it can handle larger instances of trains and networks. For instance, we could investigate a decomposition model, for example, with a solution scheme involving column generation techniques. However, such an approach requires the reformulation of the model, which is quite challenging considering the comprehensiveness and complexity of the current model.

Another direction is to investigate different strategies for adding/removing constraints and variables. For now, we add one train at each iteration and solve the model up to $\varepsilon-$ optimality. Then, we remove the non binding constraints, add one new train and solve again. Indeed, different alternate strategies could be investigated as to which trains should be added. We might also cluster the trains (according to their origins, their costs) and add all trains belonging to the same cluster at the same time. We could also design a heuristic in order to obtain quickly a first feasible schedule. These approaches might improve significantly the running times without much modifications into the SDT_TS mathematical model.

Enhancing the simulation

For the simulation, we could investigate further logic to better handle the deadlock issue. In the current simulation, the deadlock issue is solved at the expense of increasing significantly the travelling times of the trains. One possible improvement is that we might come up with an approximate algorithm that can avoid deadlocks which only slightly increases the travelling times of the trains.

Bibliography

[CCT10] V. Cacchiani, A. Caprara, and R. C. P. Toth. A column generation approach to train timetabling on a corridor. *4OR, Quarterly Journal of Operations Research*, 6(2):125–142, 2010.

[CL95] M. Carey and D. Lookwood. A model, algorithms and strategy for train pathing. *Journal of Operation Research Society*, 46(8):988–1005, 1995.

[Cui10] Y. Cui. *Simulation-Based Hybrid Model for a Partially- Automatic Dispatching of Railway Operation*. PhD thesis, University of Stuttgart, Germany, 2010.

[DLZL06] M. Dessouky, Q. Lu, J. Zhao, and R.C. Leachman. An exact solution procedure for determining the optimal dispatching times for complex rail networks. *IIE Transactions*, 38:141–152, 2006.

[DM04] M.J. Dorfman and J. Medanic. Scheduling trains on a railway network using a discrete event model of railway traffic. *Transportation Research Part B*, 38:81–98, January 2004.

[HKF96] A. Higgins, E. Kozan, and L. Ferreira. Optimal scheduling of trains on a single line track. *Transportation Research B*, 30(2):147–161, 1996.

[ICF$^+$04] P. Ireland, R. Case, J. Fallis, C. Van Dyke, J. Kuehn, and M. Meketon. The Canadian Pacific Railway transforms operations by using models to develop its operating plans. *Interfaces*, 34(1):5–14, 2004.

[KH95] D. Kraay and P.T. Harker. Real-time scheduling of freight railroads. *Transportation Research*, 29B(3):213–229, 1995.

[KSS07] W.D. Kelton, R.P. Sadowski, and D.T. Sturrock. *Simulation with Arena*. McGraw-Hill, New York, fourth edition edition, 2007.

[LDL04] Q. Lu, M. Dessouky, and R. C. Leachman. Modeling train movements through complex rail networks. *ACM Transactions on Modeling and Computer Simulation*, 14(1):48–75, 2004.

[Mar99] P. Martin. Train performance and simulation. In *Winter Simulation Conference 1999*, pages 1287–1294, 1999.

[MD11] S. Mu and M. Dessouky. Scheduling freight trains traveling on complex networks. *Transportation Research Part B: Methodological*, 45:1103–1123, 2011.

[Pac11] J. Pachl. Deadlock avoidance in railroad operations simulations. In *90th Annual Meeting of the Transportation Research Board in Washington DC*, 2011.

[RS02] J. Rathmell and David T. Sturrock. Arena: the arena product family: enterprise modeling solutions. In *Proceedings of the 34th conference on Winter simulation: exploring new frontiers*, pages 165–172, 2002.

[ZZ07] X. Zhou and M. Zhong. Single-track train timetabling with guaranteed optimality: Branch-and-bound algorithms with enhanced lower bounds. *Transportation Research Part B*, 41:320–341, 2007.