Improvement of Decision Support System for Rail Depot Planning

Master of Science Thesis in the Programme Computer Science:Algorithms, Languages and Logic

BERKAY BEYGO

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, September 2011

Improvement of Decision Support System for Rail Depot Planning

BERKAY BEYGO

Examiner: PETER DAMASCHKE

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover: A picture showing that *Jeppesen Systems AB* is producing solutions for customers serving in different modes of transportation sector. © *Jeppesen Systems AB*

Department of Computer Science and Engineering
Göteborg, Sweden April 2012

Preface

First of all, I would like to thank my family and my friends for supporting me throughout the thesis and my master's education. Since the thesis was accomplished in collaboration with *Jeppesen, a Boeing Company,* I would like to thank Tomas Gustafsson who gave me the opportunity to do my thesis and show my skills in such a great company, to my supervisor Mattias Grönkvist for directing and supervising me with his precious skills and for always being helpful throughout the project.

Furthermore, I owe many thanks to Jesper Hansen for helping me understand the insights of the problem, sharing his valuable ideas with me and answering all my questions with great patience. Moreover, I would like to thank my examiner Peter Damaschke at Chalmers University of Technology for his contributions.

Finally, I thank all the employees working at the Göteborg office of *Jeppesen* for being so friendly and being a part of the great atmosphere at the office.

To my Family/Ailem'e

Hem eğitim yaşamım boyunca hem de hayatın her anında her zaman yanımda olan ve bana destek olan aileme sonsuz teşekkür ederim…

Abstract

The thesis is implementing an improvement for the depot planning decision support system at *DSB S-tog* (Danish State Railways S-train). In the literature, for the sake of tractability, rail depot planning (shunt planning) is divided into different sub-problems and each is solved independently. In the *Jeppesen* system we have divided the problem into following sub-problems: Composition and rotation steps, parking problem and driver scheduling.

The parking problem which was proved to be NP-Hard was seen as the hardest step of the overall plan in some literature, and the focus of the thesis is on that problem. Given a timetable and the specifications of trains in the timetable, the problem focuses on parking the trains at shunt tracks which will be idle for a while and will operate later by considering the topology of the shunt yard and the demands of shunt planners.

In the current system used at *Jeppesen*, the composition and rotation steps are solved quite efficiently for real life instances, and the parking problem is modeled as a Set Partitioning Problem in which all the feasible assignments for each track are generated explicitly. The current model then chooses at most one assignment for each shunt track. Since the number of feasible assignments grows exponentially in the number of trains, it is hard to solve the problem for some real life cases.

In the thesis, by taking into consideration the requirements from *DSB S-tog*, a MIP model is developed that has polynomial number of variables in the number of trains and shunt tracks. All the shunt tracks at *DSB* are of type LIFO that allows the trains get in/out using a single end. Although the new model satisfies all the requirements of the customer, some extensions existing in the literature are not included in the system, and some extensions differ from the ones in literature. Compared to the current model being used, instead of generating all the feasible assignments for each track which leads to an exponential number of decision variables, all the requirements are modeled as constraints. The model is tested and applied to 6 depots at *DSB S-tog*. The results show that the new MIP model is superior for most of the small and medium instances. For the larger instances, which were hard to solve by the current system and sometimes could not be solved at all, the MIP model finds the optimal solutions in a reasonable amount of time.

Table of Contents

1. Introduction

1.1. Structure of the Thesis

Chapter 1 is an introductory chapter discussing the usage of decision support systems in railway companies, followed by some brief information regarding the system currently being used at *DSB S-tog* developed by *Jeppesen*. The weakness of the current system for some real life instances and the purpose and scope of the thesis are explained afterwards. The companies *DSB S-tog* and *Jeppesen* are described briefly at the end.

Chapter 2 is a review of the literature about the usage of operations research techniques in the railway industry and since the thesis is focused on the parking problem, most literature is related to that problem.

In chapter 3, the whole rail depot planning problem is explained. The sub-problems, i.e., the matching problem, the parking problem, and driver scheduling are introduced separately and the interaction between them is depicted. The parking problem is described in more detail and the current method used in the *Jeppesen* system is explained.

The MIP model developed is discussed in chapter 4. Results and comparisons with the current model are presented in chapter 5. Further research and answers to some research questions are included in chapter 6.

List of terminology, some results from the instances and references are given in the appendix section.

1.2. Railway Planning Processes

Rail transport is one of the most common types of conveyance of passengers, especially in Europe. The popularity of railway transportation, the number of passengers using it and the complex structure of the operational plans make railway companies create systematical and detailed plans. These plans depending on the level of detail and the planning horizon are mainly divided into strategic plans, tactical plans, operational plans and short term plans.

Strategic plans are created for several years in advance in which the goals and the quality aspects of the companies are set, capacity planning to reach these goals is analyzed and demand is forecasted. Using the forecasted demand and having in mind the goals of the companies, the railway lines, number of train units and number of personnel for the activities needed are decided. After making sure that there are enough resources, in the tactical plans –created for at most one year in advance-, the resources and the capacity are allocated, the configuration of trains and their arrival / departure times are decided. Operational plans are more detailed plans and it is the phase where several adjustments resulting from sudden changes in demand during specific events e.g., concerts, sports matches, are made. Short term plans are for at most three days in advance and they handle the possible break downs, delays and unexpected situations. After having these central plans, local plans which concern single stations are created by the shunt planners at railway companies.

The rail depot planning (shunt planning) is a problem in which plans at shunt yards are created. A shunt yard is the whole area where the shunt tracks are located, and a depot is the whole station which includes a shunt yard. These operational plans cover the decisions made by shunt planners in order to run the activities in a depot efficiently. Some of these decisions are: assignments of trains to perform specific arrival/departure trips, parking of rolling stock units at the depot when they are idle and will not be in service for a while, creating the schedules for each crew member and deciding which driver will perform which shunting operation and when it will take place. Depending on the shunt yard, the planners may also be responsible for creating the routes for each arriving train from the arrival platforms to assigned shunt tracks for parking and for each departing train from the shunt yard to the departure platforms. In some shunt yards there are special places or tracks used for cleaning the rolling stock units, the decisions of when and where to clean these units are also under the control of shunt planners in such shunt yards.

Although these activities seem to be independent of each other, they are related considering the big picture. For instance, the plans of assigning trains to specific legs affect the parking step, because as a result of these assignments the trains which will be idle for a while are identified. Using these idle trains the parking problem is solved and for each train a shunt track is assigned. The assignments of trains to shunt tracks for parking constitute the input of the depot driver scheduling part. As clearly seen, the problems are related with each other and any change made in the previous steps can alter the whole problem.

These activities have been done manually for years by most railway companies. Since the problems are highly connected with each other, the planners prefer to create the plans as late as possible. Moreover, the problems of interest are difficult to solve. The difficulty and the structure of the problem, also taking into account the increasing demand in railway transportation increased the use of automated decision support systems in recent years. By using these systems the planning phases are accelerated and nowadays such systems are being used to a greater extent by the companies. Especially in Netherlands, huge investments are being made in the field.

1.3. Purpose and Scope of the Thesis

DSB S-tog uses an automated decision support system for their railway planning developed by *Jeppesen*. The system greatly helps the planners and is able to produce good and reliable solutions. It gets the station topology, the timetable of planned arrivals and departures and other relevant information as input. Using the planned arrivals / departures and their times, firstly a matching between these arriving train units and departing train units are made so that it is known which train will perform which trip. This problem is known as Train Matching Problem (TMP) in Freling et al. [1] and is called composition problem in the system developed at the company. In the rotation phase, given the composition of trains the routing of them between different depots or between the locations where local activities, e.g. cleaning, maintenance, take place are decided.

Having the output from the composition and rotation phases, the second step starts. This second problem is called the parking problem. In this step, for each idle train at the station, which will perform its operation later, a shunt track is assigned so that the train can be parked there. The solution of the parking problem is given as input to the depot driver scheduling phase where all the shunt movements are assigned to the drivers.

Prior to solving the whole problem some feasibility analysis is made in order to check whether the problem data is consistent and whether there are any infeasibilities, so they can be taken care of by the shunt planners earlier.

Although the system developed helps the shunt planners in a great way, for some real life instances the model is unable to produce a feasible solution. The reason is that it is sometimes impossible to solve the parking problem. It is modeled as a set partitioning problem where for each shunt track, all feasible assignments to that track are generated explicitly and at most one assignment is chosen. Since the number of feasible assignments for each track –decision variables- increases exponentially in the number of trains, it is hard to solve the problem for some real life cases. Especially planning at Hillerød depot –the most complex station topology- during weekends is so hard that the system runs out of memory.

The focus of the thesis is on the parking problem. Considering the requirements, limitations, the data and the station topologies from *DSB S-tog* a new model was developed and as seen from the experiments it is able to solve all the cases optimally.

1.4. *Jeppesen Systems AB* Company Profile

Jeppesen Systems AB is a subsidiary of *The Boeing Company* since 2006. *Jeppesen* specializes in providing navigational information and developing electronic navigational charts for the transportation sector. The company creates products which integrate navigation, operations and logistic information and produces today's most advanced flight information solutions, planning software, pilot training and operations management systems for the aviation companies.

Cartography services used by the marine companies are also offered. Moreover *Jeppesen* develops automated decision support and operations management systems for clients mainly in the transportation sector. Resource optimization solutions e.g., rail crew, fleet and logistics solutions are implemented so that the clients can use their equipment more efficiently.

Jeppesen has several offices around the world. The Göteborg office where the thesis was done develops and implements resource optimization solutions for airlines and railways [15].

1.5. ***DSB S-tog*** Company Profile

DSB S-tog a/s (S-tog) is a railway company located in Denmark. *S-tog* is owned by the Danish State Railways (*DSB*) which is an independent public company owned by the Danish Ministry of Transport and Energy. *DSB* is the largest railway company in Scandinavia.

DSB S-tog operates the S-trains running in the Copenhagen area, Denmark. The S-trains network today can be defined as the main public transportation in the city and connects the suburbs and other areas in the greater Copenhagen area. Today S-tog is serving more than 357,000 passengers per day. The network has 170 kilometers of dual tracks, 84 train stations and 10 different lines during rush hours in the morning. There are almost 1100 departures each day from the terminals, [16]. The network is shown in Figure 1.

The company has different types of trains with different features which are all powered by overhead lines. For instance the types SA and SD can be coupled whenever necessary, especially during rush hours, and the types SE and SH are used primarily when the traffic is low.

There are 9 depots in total. A depot is the whole station which includes a shunt yard. Two of the depots, Copenhagen central station and Høje Taastrup, do not use decision support systems and are planned and operated manually. The external cleaning of the trains is accomplished at the Hundige depot. The remaining depots Farum, Ballerup, Hillerød, Frederikssund, Køge and Klampenborg are the depots of interest in the thesis.

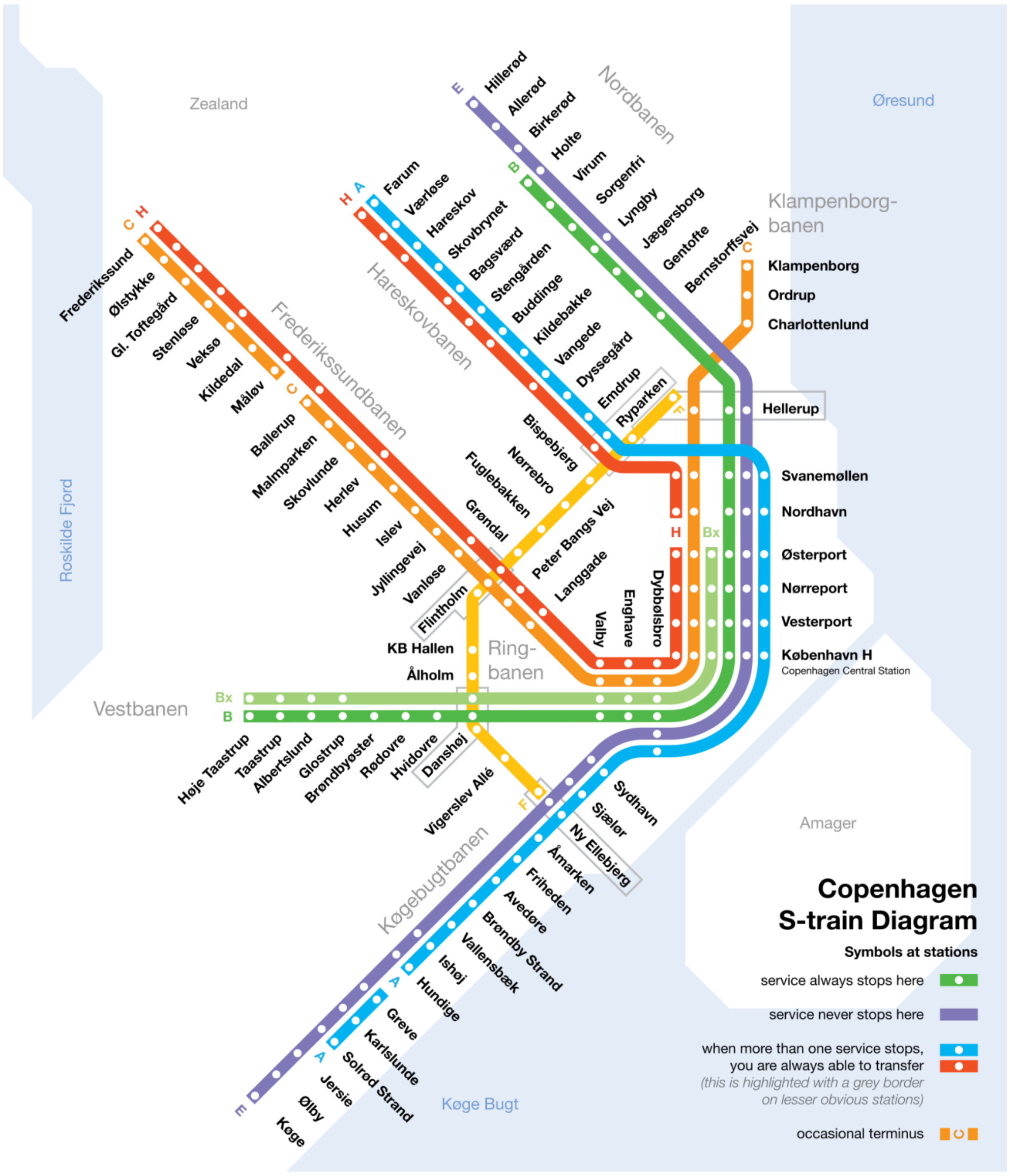


Figure 1- The S-train network.
(Wikipedia. 2009 [retrieved 2011-08-31]. Accessible at: http://upload.wikimedia.org/wikipedia/commons/thumb/6/6b/S-train_diagram.svg/2000px-S-train_diagram.svg.png)

2. Literature Review

In this section a review of the literature regarding the use of operations research techniques in the railway industry focusing on the parking problem is given. Many of the applications and articles are from the Netherlands and there is some joint work between the companies and the universities as well.

Blasum et al. [6] and Winter and Zimmermann [7], are among the first who studied the shunting problem. Blasum et al. [6] discussed dispatching trams in a depot where all the tracks were of type LIFO. The arriving trains in their research were assigned to a shunt track at the depot and later in the morning the parked trams at the depot were assigned to the trips of that day. Their focus was on assignment of parked tram units to the trips of the next day and minimizing the number of shunt moves at the depot, and they used dynamic programming. Winter and Zimmermann [7], studied dispatching of trams in a depot in real time, especially for daily schedules. They have developed optimization models and heuristic algorithms which give the decisions in a short time. These articles are the basis for some of the works done in the Netherlands.

Freling et al. [1], introduced the train unit shunting problem (TUSP) and showed that TUSP is NP-Hard. They divided the problem into two steps:

- Matching of arriving and departing train units (TMP).
- Parking these train units (TAP).

In the first step, the train matching problem (TMP), arriving train units are matched with departing train units in such a way that the shunting operations are minimized and the train units are kept together as much as possible. The mathematical model for this step is solved by a MIP solver. During the second step, the parking problem called the track assignment problem (TAP), the train units are assigned to shunt tracks for parking. It was modeled as a set partitioning problem and for each track they generate all the feasible assignments. The model picks one assignment per track. Although they have the advantage and flexibility of extending and modifying the feasibility conditions for different requirements implicitly, the total number of assignments may be exponential in the number of train units. Thus, they introduced a column generation heuristic. The method was tested and it was seen that the parking step was the bottleneck. In all the test cases except two, the instances were solved in a reasonable amount of time and all the train units were parked. The disadvantage of the approach is that dividing the problem into two sub-problems may prevent finding the global optimum or even a feasible solution.

Lentink [2], in his PhD thesis divided the whole problem into five sub-problems:

- Matching of arriving and departing train units.
- Parking of train units.
- Routing of the train units.
- Cleaning the train units.
- Crew planning.

Crew planning was considered outside the scope of the thesis and is not the focus. Firstly the solution procedures for each sub-problem are given in details; later an integrated approach for matching and parking problems is introduced. In the train matching problem (TMP), the quality of the solution was measured by the number of train units which will need parking in the next stage. In the mathematical model for TMP, shortest path problems and a matching problem was integrated. Moreover, the computational complexity of TMP was discussed. The model was tested and the results showed that the instances were solved within one second. The parking problem was modeled as a set partitioning problem where all assignments are explicitly enumerated. The advantage of the model is that the feasibility conditions and cost calculations are taken into account implicitly for each assignment. The disadvantage is that the number of track assignments may be exponential in the number of train units leading to a huge number of decision variables. They have also shown that the parking problem (TAP) is NP-Complete. A column generation heuristic was developed in which a dynamic programming algorithm was used to solve the pricing problem. According to the test cases, it was seen that the heuristic approach produced good solutions within 10 seconds except for some instances at station Zwolle which took at most 10 minutes. In the routing problem Occupied Network A* algorithm was proposed to find the possible conflicts between routes. Since the algorithm searches the routes sequentially the order of the routes found has an impact on the problem. In order to reduce this dependency, a 2-OPT (Croes et al. [18]) procedure was applied. Test results showed that ONA* search is fast, but requires interaction from the shunt planners in some instances. The 2-OPT approach resulted in improvements for some cases but it was noticed that it may get stuck at local optima.

The integrated problem of matching and parking the train units studied by Lentink [2] was developed by Schrijver [8], [9]. They developed a basic model and extended it by adding some features. The first model was restricted to LIFO tracks and had more than 300.000 constraints for an instance at Zwolle station. In order to decrease the number of constraints they aggregated the crossing and other constraints which resulted in a reduction of the number of constraints, but an increase in the number of decision variables. Next, a new model which restricts the number of mixed tracks was introduced. Train units of same type are aimed to be parked together at the same track. Since these train units can be used interchangeably the crossing situations (constraints) for such tracks are eliminated. Finally, free tracks were included in the model as an extension. This extension increased the number of decision variables and the number of additional crossings greatly. The test results showed that the solutions found by the model and the computation times depend on the restricted number of mixed tracks used and a good and quick solution could only be found if the model contains such tracks. This integrated approach improved the solution quality compared to solving the sub-problems TMP and TAP sequentially, but the computation time was longer than the sequential approach. Some similarities between their approach and this thesis were noticed, although the requirements from the railway companies seem to be different.

A CSP (Constraint Satisfaction Programming) (Rossi et al. [19]) approach for TUSP was studied in Abbink [10]. The criteria he thought as important are: finding a solution near to optimal, being able to handle large cases, involving a heuristic approach, finding robust solutions, meeting the industrial standards and solving the problem integrally rather than splitting up the problem into sub-problems. In his work an initial model was proposed which was able to find feasible solutions within a minute. Since the objective is finding a feasible solution the preferences of the shunt planners such as minimizing the number of shunting operations were not taken into account in this initial work.

Haijema et al. [11] introduced a heuristic approach that is related to dynamic programming. The solution procedure contains two steps. In the first step –blueprint algorithm- the decisions of from which track to collect the train units to form a departing train are given. The blueprint algorithm tries to obey the crossing situations, minimize the number of different tracks used to form a departing train and the shunting operations done. The second step is the matching step where the arriving train units are matched with departing units, this process is done by finding positions for the arriving units on the blueprint generated in the first step. Although splitting up the problem in these two steps cannot guarantee to find a solution, they have not encountered such a case and instances were solved less than a second.

The very first version of the current system used at *Jeppesen* was developed as a thesis project by Peter Føns [4]. The thesis uses the general model described by Freling et al. [1] in which the problem was modeled as a set partitioning problem. For each track the set of all feasible assignments to that track are generated, and the model selects at most one assignment per track. Since the number of feasible assignments per track grows exponentially in the number of train units, the tracks which are identical or very similar and have the same routing costs for each assignment were grouped together and this way the number of decision variables was aimed to be decreased. Later the model was extended to include platform parkings which will be mentioned later in detail. Prior to solving the overall rolling stock problem some feasibility checks are done and if there are any problems in the input data, they are detected earlier. These feasibility checks include checking whether the capacity of the shunt yard is enough or not. If it is not enough, another check to see whether allowing parking at platforms can make it feasible or not is performed. Moreover, minimum number of drivers required for the scheduling phase is also calculated. Intermediate shunting moves are the movement operations performed from one shunt track to another shunt track. Although these type of movements are not allowed by *DSB,* it is checked whether introduction of such movements overcomes the capacity problems and helps finding a solution. Later on a column generation approach was added to the system which slightly improved the overall solution.

D. Jekkers [12], in his master's thesis aimed to create feasible and good plans using genetic algorithms (Rayward-Smith et al. [17]). Flexible shunt times were included in the approach which extends the search space. Two different approaches were developed and the one in which GA was used together with a heuristic seemed to be superior. The test results showed that for smaller instances GA was comparable with the MIP model; whereas for larger instances GA outperformed the MIP approach and it was seen that the introduction of flexible shunt times affected the quality of solutions in a good way. In the resulting plans it was noticed that some of the shunting activities were combined which decreased the total number of shunting operations. The disadvantage of the approach is that like almost all meta-heuristics some adjustments to the parameter settings are required.

Akker et al. [13], proposed two approaches for the shunting problem: a greedy algorithm and an exact solution method based on dynamic programming. The greedy algorithm goes through the events (arrivals and departures) and for each event gives a decision immediately considering the current event of interest and the previous events which were already decided. They have indicated that the shunt planners prefer solutions in which the departing trains are ready and waiting for departure just before they depart. That's why, they scan the event list backwards, and if a departure event is encountered a shunt track is assigned to that event, otherwise if it is an arrival event then a matching between the arriving train and train units that has already parked at a shunt track is done. In the dynamic programming approach just before an event takes place a set of states for each track is computed. Starting with an initial state and the first event, a set of states are created and using these states and the second event new sets of states are created, and the process continues. Finally, a network of states is generated where each arc represents an event. Each transition in the network has a cost which reflects the shunting efforts. Thus, the route from the initial state to the final state which has the lowest cost gives the solution. In theory the approach works well but in practice for larger instances it may be impossible to obtain a solution in a reasonable amount of time.

Cornelsen et al. [14] studied the track assignment problem in which trains are allowed to stop at the same shunt track at the same time. Both sides of the station may be used for arrivals and departures, but the trains arrive and depart according to a timetable and their movement direction is also given as an input. The problem is reduced to a graph coloring problem of a conflict graph generated from the given input. For some cases the graph can be colored in polynomial time, for other cases they present approximation algorithms.

3. Overall Planning Problem

The rail depot planning problem is a complex problem containing lots of decisions to be made. Depending on the specifications of the shunt yards and the requirements from railway companies, shunt planners try to make good and flexible plans.

As clearly seen, the overall rail depot planning is a difficult problem to be solved in a single step. Thus, for the sake of tractability mostly it is divided into different sub-problems and each is solved independently. Freling et al. [1], introduces two sub-problems which were proved to be NP-hard as discussed in chapter 2:

- Train Matching Problem (TMP).
- Track Assignment Problem (TAP) - known as the parking problem-.

Lentink [2] instead identifies the sub-problems as:

- Matching problem.
- Parking problem.
- Routing problem.
- Cleaning problem.
- Crew planning.

In the current system at *Jeppesen*, the whole problem is divided into the following sub-problems:

- Composition problem.
- Rotation problem.
- The parking problem.
- The depot driver scheduling problem.

In this chapter first the sub-problems will be introduced one by one; afterwards the current solution procedure in the *Jeppesen* system will be explained.

3.1. The Matching Problem

Each arriving and departing train has a number of train units connected in some order which is called configuration. Given the arriving train configuration, departing train configuration, and the timetable of planned arrivals / departures, the train units in the arriving trains should be matched with the train units in departing trains in such a way that the shunting operations are minimized.

To have a feasible matching not only the arriving and departing train units should fit with each other but also the configurations of train units in each matching must be the same. For instance, the configuration BA-SA is not the same as SA-BA. On the other hand, train units of the same type are exchangeable, which increases the possible options for the planners, and provides them some flexibility. Moreover, arriving train units can be matched to departing train units if and only if the time of the arriving train is earlier than the time of the departing train and the time difference between them is enough to perform the shunting operations required. Table 1 below shows an example of a simplified timetable.

Table 1 **-A sample timetable showing the arrival/departure of trains**

Train #	Time of event	Event Type	Configuration
1022	2009-11-10 08:25	Arrival	SA-SA-SD
1023	2009-11-10 09:10	Arrival	SE-SE
2020	2009-11-10 13:18	Departure	SA-SD
1024	2009-11-10 14:12	Arrival	SH-SH
1025	2009-11-10 15:23	Arrival	SH
2021	2009-11-10 17:48	Departure	SH-SH-SE-SE
2022	2009-11-10 19:22	Departure	SA-SH

A feasible matching according to the timetable above could be:

- matching of the first SA unit of arriving train 1022 to the left side of departing train 2022
- matching of the SA-SD units of arriving train 1022 to departing train 2020
- matching of arriving train 1023 to the right side of departing train 2021
- matching of arriving train 1024 to the left side of departing train 2021
- matching of arriving train 1025 to the right side of departing train 2022

3.2. The Parking Problem

In this section, the parking problem which is the focus of the thesis will be introduced in general using some notation from the system developed.

Given the matching of arriving and departing train units from previous step, the station topology and the timetable, the parking problem is the next step of overall rail depot planning.

A matching between an arriving train and a departing train which is a set of connected train units always kept together throughout the problem are called a block. A block can be a single train unit or a set of train units, but it is the smallest unit in the problem which cannot be further detached into pieces. Throughout the thesis the term block will be used.

A block has the following properties:

- Arrival/departure time,
- Arrival/departure leg,
- Arrival/departure platform,
- The type of train units block is composed of,
- The length of the block,
- Position of the block in the arrival leg,
- Position of the block in the departure leg.

A leg is the trip of a train and is defined by two end points. Each block has an arrival leg and a departure leg. Arrival leg is the leg the block performs when it arrives at the depot and the departure leg is the leg the block will perform after leaving the depot. Since there can be multiple blocks belonging to the same leg the positions of the blocks in the arrival/departure legs should be recorded.

A set of blocks, which has some idle time (not operating) during their stay at the shunt yard, and thus need to be parked somewhere are of interest in this step. Blocks which are leaving the shunt yard soon, or the ones which arrive to and depart from the same platform and there are no other activities on that platform during their stay at the depot do not need to be parked and are not taken into consideration.

The inputs to the parking problem are the station topology and a timetable of blocks as stated above. The first input, the topology of the station, has the information regarding the layout, possible connections between tracks, and length of the tracks. Hillerød depot's topology can be seen in Figure 2.

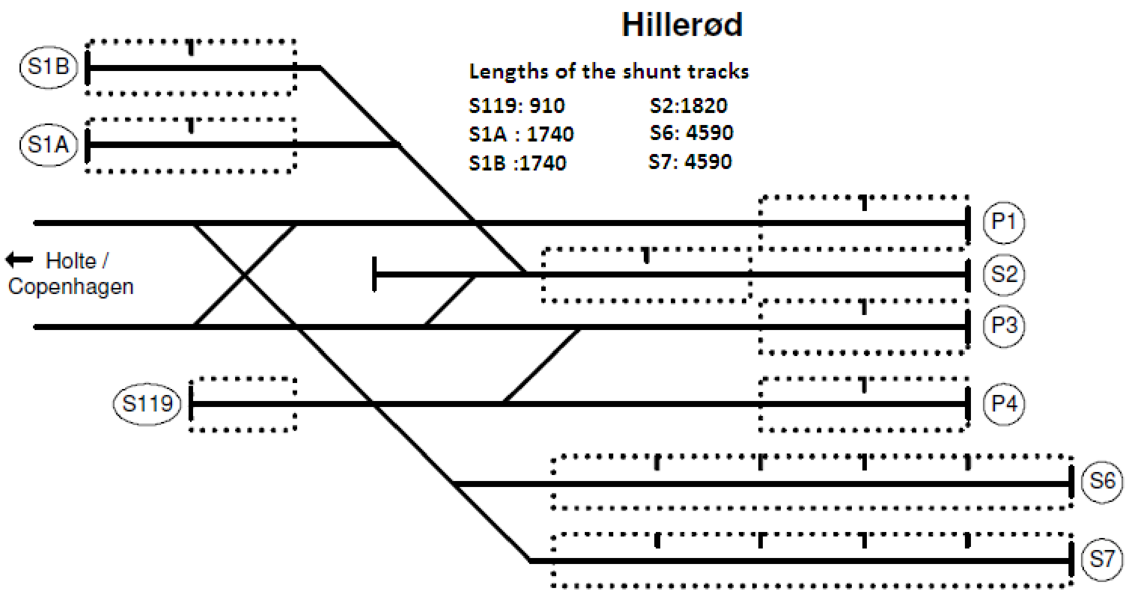


Figure 2 -The station topology of Hillerød depot, Peter Føns [4].

The second input is a chronological timetable of arrivals and departures of blocks. Each arrival and departure is called an event. The timetable shows the type of the events (arrival or departure), the arrival/departure leg, type of the train units in the blocks and arrival/departure platform of the blocks. The first arrival/departure leg column for arrival events indicates the arrival leg, whereas the second arrival/departure leg column indicates the departure leg. For the departure events on the other hand, the first arrival/departure column shows the departure leg, and the other column shows the arrival leg. An example of a timetable is shown in Table 2.

There are two rows for each block in the input: one for the arrival event of that block and one for the departure event. For instance, the first row (arrival event) and the third row (departure event) give information about the same block saying that the block is arriving to the first platform of the shunt yard at 01:15 on 2005-06-04 in 20050603_S-C-30201_KL/86_ arrival leg, and it is departing from the second platform of the shunt yard at 04:54 on 2005-06-04 in 20050604_S-C-30116_BA/86_ departure leg and the train units in the block are of type LHB.

Table 2 **-Timetable showing the information for each event**

Time	Event Type	Arr./Dep. Leg	Train Units	Arr./Dep. track	Arr./Dep. Leg
2005-06-04T01:05	Arrival	20050603_S-C-30201_KL/86_	LHB	1	20050604_S-C-30116_BA/86_
2005-06-04T01:09	Arrival	20050603_S-H-50904_FS/86_	LHB	2	20050604_S-C-30117_BA/86_
2005-06-04T04:54	Departure	20050604_S-C-30116_BA/86_	LHB	2	20050603_S-C-30201_KL/86_
2005-06-04T05:14	Departure	20050604_S-C-30117_BA/86_	LHB	2	20050603_S-H-50904_FS/86_
2005-06-04T05:25	Arrival	20050604_S-C-30214_KH/86_	RENO	1	20050604_S-C-30120_BA/86_
2005-06-04T06:14	Departure	20050604_S-C-30120_BA/86_	RENO	2	20050604_S-C-30214_KH/86_
2005-06-05T01:05	Arrival	20050604_S-C-30201_KL/86_	LHB	1	20050605_S-C-30119_BA/86_
2005-06-05T05:54	Departure	20050605_S-C-30119_BA/86_	LHB	2	20050604_S-C-30201_KL/86_
2005-06-05T06:25	Arrival	20050605_S-C-30217_KH/86_	RENO	1	20050606_S-C-30117_BA/86_
2005-06-06T01:05	Arrival	20050605_S-C-30201_KL/86_	LHB	1	20050606_S-C-30116_BA/86_
2005-06-06T04:54	Departure	20050606_S-C-30116_BA/86_	LHB	2	20050605_S-C-30201_KL/86_
2005-06-06T05:14	Departure	20050606_S-C-30117_BA/86_	RENO	2	20050605_S-C-30217_KH/86_

Although the arrival and departure times shown in the timetables seem to be fixed, there is some flexibility added in the current system. There is a parameter which sets how tightly in time the blocks can park at the same shunt track. It limits how much an arriving block must be separated from a previously parked block so that the departing block can depart without any obstructions.

The earliest departure time for a block is the earliest time it can leave its parked shunt track and move to the departure platform. If there is a time interval in which there are no arrival/departure events on that platform just before the departure time of the block, the block can leave the shunt track earlier. The latest arrival time of a block at shunt track *t* is the time the block can move to *t* after finishing its operations e.g., waiting for passengers to get off at the arrival platform. The block can stay at its arrival platform until there is a new arrival/departure event on that platform after its arrival. Having these flexibilities, the user can choose one of the following options in the system in order to allow block *b1* be parked at the same shunt track as block *b2*.

- Option 1: Arrival time of *b1* > Departure time of *b2*
- Option 2: Arrival time of *b1* > Earliest departure time of *b2* from *t*
- Option 3: Latest arrival time of block *b1* > Departure time of *b2*
- Option 4: Latest arrival time of block *b1* > Earliest departure time of *b2* from *t*

According to Lentink [2], the parking problem is considered as the one which affects the robustness of the overall plan mostly compared to the other ones.

Using the inputs mentioned, the problem is assigning the blocks to tracks and satisfying the constraints for feasibility and the constraints determined by the customers. The constraints which should definitely be satisfied for the feasibility of the problem are called hard constraints, and the constraints which we would like to satisfy if possible are called soft constraints. Usually, soft constraints have penalties which form the cost function.

Before describing the constraints some notations will be introduced. A crossing is a situation when the departure of a block is obstructed by another block. If block *m* arrives at the station earlier than block *n*, departs before *n*, and the arrival time of *n* is earlier than the departure time of *m*, they cannot be parked at the same track (see Figure 3). Parking them at the same track will obstruct *m*'s departure.

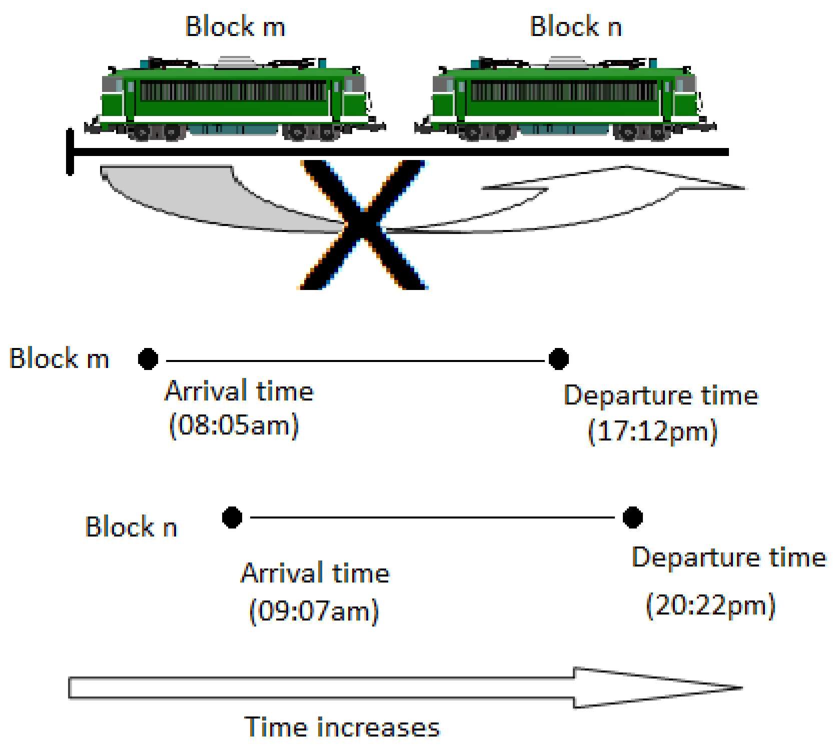


Figure 3 **-Crossing between two blocks**

The hard constraints for the parking problem in general are as follows:

- Two blocks that have a crossing between are not allowed to park at the same shunt track.
- At any time, the total length of the blocks on each shunt track cannot exceed the length of the track.
- A block can only be parked at a shunt track if there is a driving connection between the shunt track to be parked at and the arrival platform of the block, and between the shunt track and the departure platform of the block.

The soft constraints included in the model are:

- Sometimes it is the case that the capacity of the shunt yard is not enough to park all the blocks of interest or the input to the problem data has some inconsistencies resulted from the railway companies or from the previous steps. The number of blocks which are impossible to park at the shunt yard (unparked) should be minimized.
- Platform tracks are basically the same as shunt tracks, but for security reasons and because they are busier than shunt tracks it is not favorable to park the blocks at platforms. However, while some blocks are allowed to be parked at platforms as discussed in detail in section 4.3, plans without platform parkings are preferred.

- A broken arrival is a situation when blocks arriving in the same arrival leg are not parked together in the right order at the same track, thus have to be detached at the platform and be parked separately. Similarly, a broken departure occurs when blocks departing in the same departure leg together are retrieved from different tracks or are not parked in the right order at the same track and have to be attached at the platform. The number of broken arrivals/departures should be minimized so that the attachment/detachment operations done at the platforms will be minimized.
- Parking different types of blocks consecutively at the same shunt track anytime should be minimized. Solutions having the same type of blocks parked at each shunt track are preferred.
- The total cost of movement within the shunt yard should be minimized. The movement cost includes the resources e.g. time, drivers, and energy, used when parking the blocks at the shunt tracks and retrieving them from the tracks to the platforms.

3.3. Driver Scheduling

After having the matching of arriving and departing trains and identifying the specific shunt tracks where to park them, the next step is to schedule the crew members for the activities in the shunt yard.

Depending on the specifications of the shunt yard and the requirements from the railway companies the scheduling part may vary. Although the core part of this step is assigning drivers to the shunting operations, a planner may need to schedule the activities for cleaning crews and shunting assistants as well. In the depots we are dealing with at *DSB*, the cleaning operations are either performed at some other special places or are planned manually. Thus, scheduling of cleaning personnel is out of the scope of the scheduling part. Shunting assistants are responsible for attachment and detachment operations of blocks at platforms and shunt tracks. Scheduling of shunting assistants is out of scope here as well.

Given a set of shunting movements and the times when to perform them, assignment of drivers to the activities is carried out in the scheduling part. Although the arrival and departure times in the problem may seem to be fixed, some flexibility can be created after analyzing the input. For instance, a block can be moved from the shunt track it was parked at to its departure platform a few minutes earlier if there are no other activities during that time at the platform, or similarly it can be moved from its arrival platform to the assigned shunt track a few minutes later than its arrival time and can wait on its arrival platform until the next event on that platform takes place. These time intervals in which the shunting operations can be performed are called time windows. For instance the fourth event in Table 3 is a departure at 19:28 but the movement starts earlier at 19:17 and finishes at 19:21. Having such time windows increases robustness and provides flexibility to the planners so that simultaneous movements in the depot are minimized, but makes the problem more challenging. In consequence, the decision of when an operation will be started, when it will be finished and who will perform it so that simultaneous moves will also be minimized is given in this step by considering the walking times of drivers within the shunt yard, the driving times between the tracks and obeying the laws and regulations. Table 3 below shows a sample schedule for drivers.

Table 3 **-A Sample Driver Scheduling**

Time	Type	Arr./Dep. Leg	Train Units	Tracks	Arr./Dep. Leg	Movement Time	Driver
2006-12-15T08:32	Arrival	1010	SE	2/ 12	1013	08:32 -> 08:39	E0
2006-12-15T08:52	Arrival	1011	SE	2/ 11	1013	08:52 -> 08:59	E0
2006-12-15T19:12	Arrival	1012	SAR	2/ 10	1019	19:12 -> 19:16	E0
2006-12-15T19:28	Departure	1013	SE	12/ 2	1010	19:17 -> 19:21	E1
2006-12-15T19:28	Departure	1013	SE	11/ 2	1011	19:21 -> 19:28	E0
2006-12-16T00:32	Arrival	1014	SAR	2/ 10	1018	00:32 -> 00:39	E0
2006-12-16T00:52	Arrival	1015	SAR	2/ 12	1020	00:52 -> 00:59	E0
2006-12-16T01:12	Arrival	1016	SAR	2/ 12	1017	01:12 -> 01:19	E0
2006-12-16T04:48	Departure	1017	SAR	12/ 2	1016	04:41 -> 04:48	E0
2006-12-16T05:08	Departure	1018	SAR	10/ 2	1014	05:01 -> 05:08	E0
2006-12-16T05:28	Departure	1019	SAR	10/ 2	1012	05:21 -> 05:28	E0
2006-12-16T05:48	Departure	1020	SAR	12/ 2	1015	05:41 -> 05:48	E0

3.4. Current Method Used In the *Jeppesen* System

As stated earlier in the chapter, the problem is divided into the following steps in *Jeppesen* system: Composition and rotation phases, parking problem and driver scheduling. Firstly the overall solution method will be introduced briefly, and then the current method used for solving the parking problem will be explained in more detail.

The composition problem at *Jeppesen* corresponds to the matching problem. A composition is defined as a set of train units connected together in a certain order. In the composition phase given a timetable, topology of the railroad and forecast of the number of passengers, the compositions for each train are decided satisfying a set of constraints. Additional constraints to the ones in TMP can be assuring that some train units remain the same throughout the trip so that passengers will not need to change their seats at mid-stops and some constraints concerning the technical limitations about attaching and detaching the train units. The composition problem is modeled as a MIP.

Given the composition solution and the connections between depots, the rotation problem focuses on routing of these units. This routing is between different stations and also maintenance locations. In the system these two steps, composition and rotation, are related with each other. The maintenance periods and operations required for each train unit varies. Thus, when a composition solution is found, it may not be feasible for maintenance. To overcome this problem after finding a solution to the composition problem the rotation step is called and some constraints are added if necessary. Depending on the nature of the problem instance, these two steps can be solved in sequence or in a feedback loop. In consequence, the shunting operations and costs are minimized and the compositions and routings are determined, J. Hansen [3].

The overall solution procedure used in the *Jeppesen* system can be seen in Figure 4. The timetable of planned arrivals / departures, topology of the railroad, forecast of the number of passengers are given as inputs to the composition and rotation phases and matching of arriving and departing train units with their configurations is obtained as output. By using the matching of arriving and departing trains (blocks) together with the station topology and a timetable containing information for each block, the parking problem generates the assignments of blocks to tracks. In the final step, drivers are assigned to the shunting operations in the plan.

The parking problem was formulated as a set partitioning problem by Hillier and Lieberman [5]. Having the same strategy and extending the model so that it satisfies the requirements of *DSB*, the basic modeling was done in 2006 by Peter Føns [4] as a thesis work using the general model described in Freling et al. [1]**.** All feasible assignments for each track are generated prior to solving the problem, and the model chooses at most one assignment for each shunt track and a block is covered by at most one assignment.

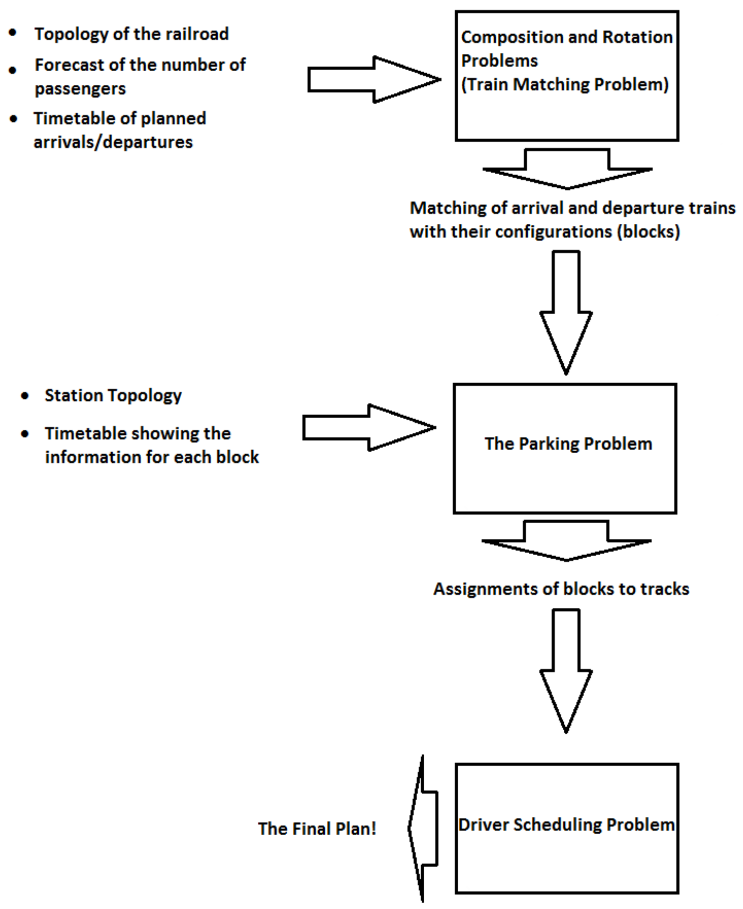

Topology of the railroad
Forecast of the number of passengers
Timetable of planned arrivals/departures
Composition and Rotation Problems (Train Matching Problem)
Matching of arrival and departure trains with their configurations (blocks)
Station Topology
Timetable showing the information for each block
The Parking Problem
Assignments of blocks to tracks
The Final Plan!
Driver Scheduling Problem

Figure 4 -Overall solution procedure in the *Jeppesen* system

In order to show what is meant by the set of feasible assignments for shunt track *s,* suppose that we have the timetable shown in Table 4 below, in which the blocks are ordered by their arrival times to the depot.

Table 4 -A simplified timetable of blocks

Block #	Length	Arrival Time	Departure Time	Arr Leg	Train Units	Arr/dep Platform	Dep. Leg
1	200	Mon. 01:05	Mon. 04:54	20050603_S-C-30201_KL/86_	SA	1/2	20050604_S-C-30116_BA/86_
2	100	Mon. 01:09	Mon. 05:14	20050603_S-H-50904_FS/86_	LHB	2/2	20050604_S-C-30117_BA/86_
3	300	Mon. 05:25	Wed. 06:14	20050604_S-C-30214_KH/86_	RENO	1/2	20050604_S-C-30120_BA/86_
4	300	Tue. 06:25	Wed. 05:14	20050605_S-C-30217_KH/86_	RENO	1/2	20050606_S-C-30117_BA/86_
5	100	Wed. 01:05	Wed. 04:54	20050605_S-C-30201_KL/86_	LHB	1/2	20050606_S-C-30116_BA/86_

The timetable shows information regarding each block. For instance, block 1 is of type SA and has a length of 200 meters. It arrives at the first platform of the shunt yard on Monday at 01:05 in 20050603_S-C-30201_KL/86_ arrival leg, and departs from the second platform on Monday at 04:54 in 20050604_S-C-30116_BA/86_ departure leg.

Suppose that we have a LIFO shunt track, open from a single end, that is 500 meters long and has connections to each platform. The set of all feasible assignments for this track are then shown in Table 5 below.

Table 5 **-Set of all feasible assignments for a LIFO track using data in Table 4**

Block1	Block1, Block4	Block4, Block5
Block2	Block1, Block5	Block1, Block3, Block5
Block3	Block2, Block3	Block1, Block4, Block5
Block4	Block2, Block4	Block2, Block3, Block5
Block5	Block2, Block5	Block2, Block4, Block5
Block1, Block3	Block3, Block5	

Note that block 1 and block 2 have a crossing between and can never be parked at the same shunt track together, thus none of the feasible assignments contains both of them. On the other hand although block 3 and block 4 do not have a crossing, there is a time interval in which both of them are parked at the depot at the same time, and the total length of the blocks (block 3: 300 meters, block 4: 300 meters) exceeds the capacity of the shunt track which results in an infeasibility, so they are not allowed to park together on that track. The total length of blocks 1, 3 and 5 also exceeds the capacity of the shunt track, but they can be parked at the same track, because when block 3 and block 5 arrive at the depot, block 1 will already have departed and the total length of block 3 and block 5 does not violate the length restrictions.

The formulation introduces the sets as follows: let B be the set of blocks, S the set of shunt tracks, F^s the set of feasible assignments on track $s \in S$,and F_b^s the subset of F^s which includes block b in all the feasible assignments.

$$x_f^s = \begin{cases} 1, if\ assignment\ f \in F^s\ is\ covered\ by\ shunt\ track\ s \in S \\ 0, otherwise \end{cases}$$

$$y_b = \begin{cases} 1, if\ block\ b \in B\ is\ unparked \\ 0, otherwise \end{cases}$$

are the decision variables and the objective function is formed as;

$$Minimize \sum_{s \in S} \sum_{f \in F^s} x_f^s c_f^s + p \sum_{b \in B} y_b \qquad (1.1)$$

s.t. the main constraints in the model

$$\sum_{s \in S} \sum_{f \in F_b^s} x_f^s + y_b = 1 \ \forall b \in B \qquad (1.2)$$

$$\sum_{f \in F^s} x_f^s \leq 1 \ \forall s \in S \qquad (1.3)$$

$$x_f^s \in \{0,1\} \ \forall s \in S, \forall f \in F^s \qquad (1.4)$$

$$y_b \in \{0,1\} \ \forall b \in B \qquad (1.5)$$

c_f^s is a cost value associated with each feasible assignment $f \in F^s$ on shunt track s. The cost value depends on the preferences of the shunt planners. It includes the transportation costs of moving the blocks in the assignments, penalties for not satisfying some of the soft constraints mentioned above e.g., broken arrivals/ departures, mixing different types of train units at the same track. p is the penalty cost of not parking a block anywhere.

Constraints 1.2 state that block b is either covered by exactly one assignment on a single shunt track or it is not parked at all. In constraints 1.3, it is stated that each shunt track $s \in S$ can have at most one assignment, and finally all the decision variables are binary.

The model above is the basic model; some extensions were made in order to satisfy the requirements from *DSB*. For instance, *DSB* allows blocks departing first in the morning be parked at the platform tracks if it will result in a better cost or increase the total number of blocks parked at the shunt yard.

Furthermore, the extended model reduces the number of possible assignments by considering symmetry if there are some identical (symmetrical) tracks -having the same assignments and the same costs for each assignment-. For such tracks it does not matter which one to use. Thus, similar tracks are grouped together and this new extended model leads to a decrease in the number of decision variables. Some more extensions have been introduced to the model, and a column generation approach has been applied.

In result, although the overall solution procedure and the parking problem strategy are able to solve most of the instances, there are some cases in which it is not even possible to find a feasible solution since the number of feasible assignments per track is exponential in the number of blocks and results in a huge number of decision variables in the parking step.

4. MIP Approach

4.1. Introduction

The focus of this chapter is the MIP approach developed for the parking problem. The model will be explained in detail with the help of some figures.

Given a timetable, the topology of the station, all the information regarding each block in the timetable as stated in section 3.2, the model parks as many blocks as possible to the depot satisfying a set of constraints and minimizing the costs to the railway company.

The given timetable, showing the arrival and departure of blocks, can be of any length. Usually the timetables are one day long during weekdays and two days long on the weekends. Prior to solving the problem, the timetable is analyzed and whenever a moment all the shunt tracks are empty is detected, the timetable is split into pieces. This splitting process is done for each such moment detected in the timetable. Since these pieces are independent of each other -there is no event overlapping between two- they can be solved independently. This way, the problem instance is divided into smaller problems which are easier to solve.

As mentioned before, a block is either a single train unit or a set of train units connected to each other that remain together throughout the problem. One can say that a block is the smallest unit which cannot be further detached into smaller pieces. A shunt track is a track connected to some platforms and is used for parking of blocks when they are not operating. The platforms where the passengers get on/off the trains are also tracks. Even though it is not desired to park the blocks at the platforms, some blocks are allowed to park at their departure platforms in order to increase the total capacity of the depot and handle possible problems existing in the input data or data from previous steps.

The topology of a shunt yard gives us all the relevant information regarding the depot. Number of platforms, number of shunt tracks, the connections and routes between the tracks, the lengths of the tracks are all obtained via the topology. All the tracks in *DSB S-tog* are of type LIFO (last in first out). In some stations there are situations where the arrival/departure platform of a block is not connected to some of the shunt tracks or departure platform of a block and its arrival platform may not have common connections to some shunt tracks. Whenever we have such situations it becomes impossible to park the blocks at these shunt tracks. Moreover, there are cases where the connected blocks in an arrival/departure leg may need to turn and change the driving direction on their route at the depot which may result in crossing situations. All the situations will be analyzed in this chapter, and starting from a basic model the final model will be described step by step.

4.2. The Basic Model

In this section the basic model is introduced. The basic model excludes parking at platforms, the cases when an arrival/departure leg consists of more than two blocks and does not incur costs for parking different types of train units next to each other at the same shunt track.

The requirements from *DSB* which are included in the basic model are:

- There are some blocks that should be parked at certain shunt tracks; otherwise they should be left unparked. Those kinds of blocks are said to be locked to specific shunt tracks.
- Broken arrivals/departures should be minimized in order to decrease the attachment/detachment activities at the platforms. Recall that broken arrival is the term used for describing the situation when two blocks arriving in the same arrival leg cannot be parked together at the shunt tracks as connected units thus detached at the platform. Broken departure on the other hand describes the situation when two blocks departing in the same departure leg are retrieved from different tracks and are attached at the platforms.

In the basic model, the blocks can either be parked at a shunt track, or are not parked at all.

SETS IN THE BASIC MODEL:

B: Set of blocks in the problem.
L: A subset of *B* which contains the blocks locked to some shunt tracks.
S: Set of shunt tracks.

THE NOTATIONS USED IN THE BASIC MODEL:

$locks_l$: maps block $l \in L$ to shunt track $s \in S$.
$lengthB_b$: Length of block $b \in B$.
$lengthS_s$: Length of shunt track $s \in S$.
$arrBlockPairs$: Set of pairs of consecutive blocks arriving in the same leg. For instance suppose that blocks $b1$, $b2$, $b3$, $b4$ arrive in the same leg in that order, then the set will contain pairs ($b1$, $b2$), ($b2$, $b3$), ($b3$, $b4$). Note that in the basic model we have at most two blocks belonging to the same leg.
$depBlockPairs$: Set of pairs of consecutive blocks departing in the same leg. Similarly, if blocks $b5$, $b6$ and $b7$ depart in the same leg in that order then the set will contain pairs ($b5$, $b6$), ($b6$, $b7$).

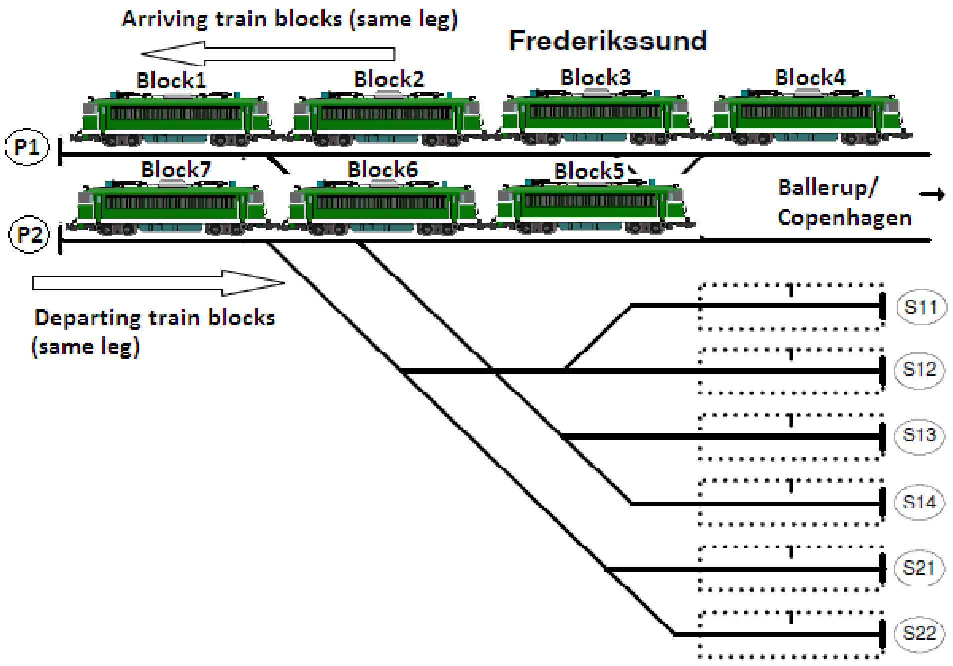


Figure 5-Arriving and departing blocks in the same leg (Frederikssund depot)

$crossingGeneral_s$: Set of pairs of blocks which are not in the same leg, and have crossing between when both are parked at $s \in S$. A crossing is a situation when the departure of a block is obstructed by another block (See Figure 3), and they are not allowed to be parked at the same track. Depending on the parameter mentioned in section 3.2 (how tightly in time the blocks can be parked at the same shunt track) the set of pairs (block *b1*, block *b2*) for each shunt track can be formed in one of the following ways:

Block *b1* and *b2* have a crossing between at shunt track $s \in S$ if

- Option 1: (Arrival time of *b1* < Arrival time of *b2*) AND (Departure time of *b1* < Departure time of *b2*) AND (Departure time of *b1* > Arrival time of *b2*)
- Option 2: (Arrival time of *b1* < Arrival time of *b2*) AND (Departure time of *b1* < Departure time of *b2*) AND (Earliest departure time of *b1* from $s \in S$ > Arrival time of *b2*)
- Option 3: (Arrival time of *b1* < Arrival time of *b2*) AND (Departure time of *b1* < Departure time of *b2*) AND (Departure time of *b1* > Latest arrival time of *b2* at its arrival platform)
- Option 4: (Arrival time of *b1* < Arrival time of *b2*) AND (Departure time of *b1* < Departure time of *b2*) AND (Earliest departure time of *b1* from $s \in S$ > Latest arrival time of *b2* at its arrival platform)

$sameArrLegTcrossing_s$: Set of pairs of consecutive blocks belonging to the same arrival leg and will have a crossing between when parked together as connected units at shunt track $s \in S$. In order to check this crossing situation we need to check the following two conditions first:

1. Are the blocks turning (changing direction) on the platforms before the movement begins?
2. Are the blocks turning during their movement to the shunt tracks?

Each shunt track and platform has two ends (sides) which are denoted as ‘A’ and ‘B’ or ‘N’ and ‘S’. The notations ‘A’ and ‘B’ are used in the thesis (see Figures 6 and 7). All the shunt tracks at *DSB* are of type LIFO and open on a single end. The arrivals to the tracks and departures from the tracks are accomplished using this open side. The blocks in the same arrival leg need to turn on the platform if the side from which they arrive at the platform and the side of the platform which connects it to the shunt track *s* is the same side. In other words if the blocks of interest are arriving at the platform and leaving the platform for parking by using the same side, then they need to change the driving direction (turn) before the movement starts.

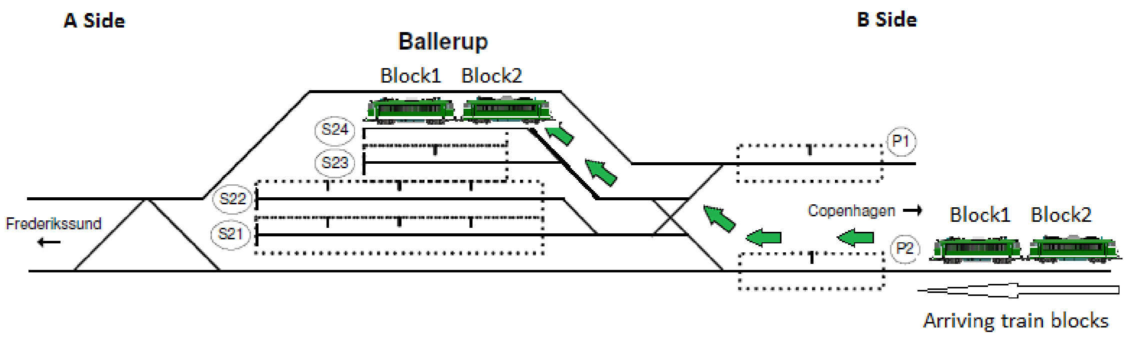


Figure 6 -Arriving train blocks do not need to turn at the platform

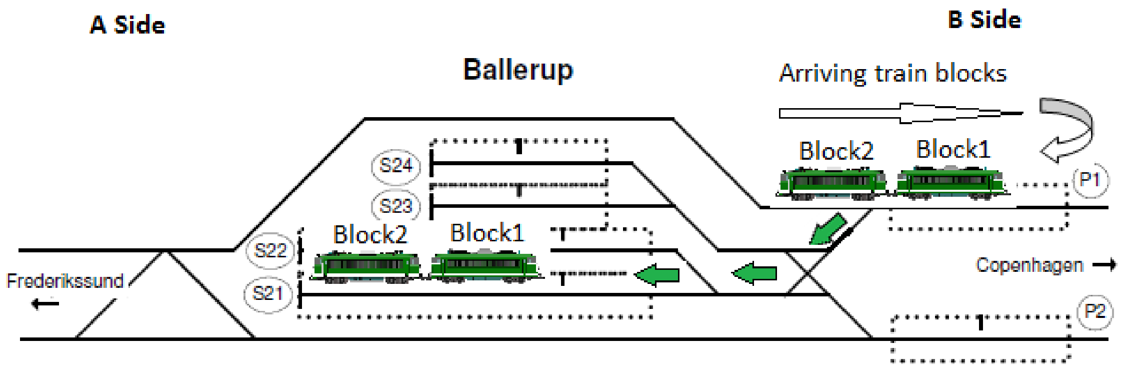


Figure 7 -Arriving train blocks need to turn at the platform

Figure 6 shows the situation where two blocks are arriving at the depot via the B side of the second platform, and moving to shunt track 24 using the A side of the platform and driving direction is not changed. On the other hand in Figure 7, the blocks are arriving at the depot via the A side of the first platform, and moving to shunt track 21 again using the A side of the platform and the driving direction is changed. As seen in the figure, block 1 is leading the leg when they arrive, then block 2 takes the control.

The second condition to be checked is whether the blocks turn during their movement to the shunt tracks or not. This turn condition depends on the topology and the connections between tracks in the shunt yard. At some connections it is impossible for the blocks to go straight ahead and pass the corners because a certain angle is required in order to drive the blocks at the corners.

For instance in Figure 8, the connection between platform 1 and shunt track 6 at Hillerød depot forces blocks turn on the way and change driving direction.

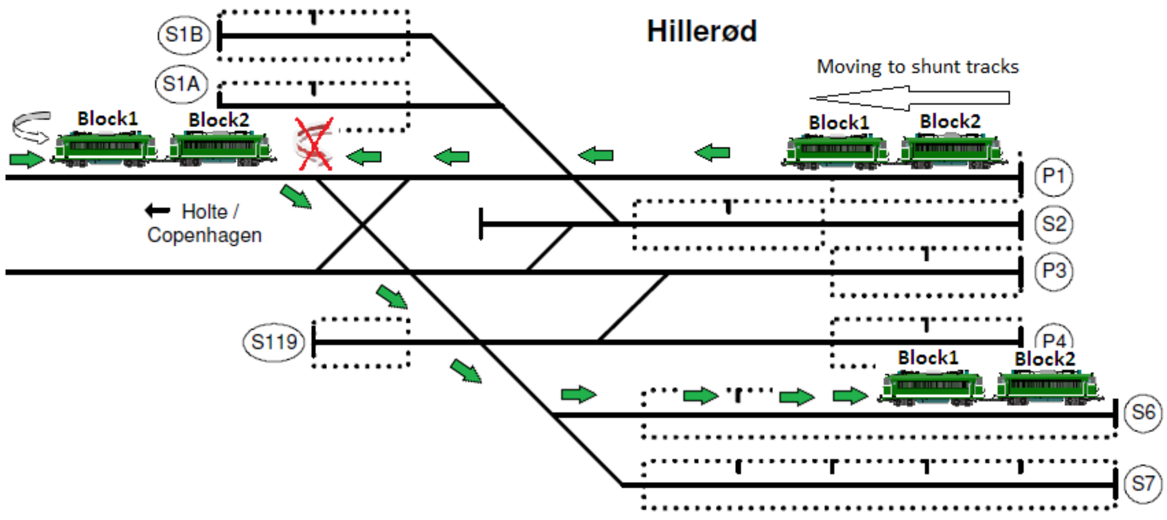


Figure 8 -Turn on the way at Hillerød depot

Having introduced the possible turning situations, in general a turning occurs if and only if just one of the turning conditions takes place. If both turnings occur they simply cancel each other. It can be interpreted as the XOR operation of two values.

Table 6 -XOR of turning situations

Turning Situation 1	Turning Situation 2	Turning in result
True	False	True
True	True	False
False	False	False
False	True	True

In conclusion, suppose that blocks *b1* and *b2* are arriving in the same leg where *b1* is positioned first, if a turning is required in order to park at shunt track $s \in S$ and *b2*'s departure is earlier than *b1*'s departure or a turning is not required but *b1*'s departure is earlier than *b2*'s departure then there happens a crossing between blocks *b1* and *b2* if they are parked together at shunt track $s \in S$ as connected units.

$sameDepLegTcrossing_s$: Set of pairs of consecutive blocks departing in the same leg and parking the blocks in the pairs at the same shunt track $s \in S$ violates their departure position orders. Each block has a *to position value* which states the order of that block in the departure leg. Similar to the strategy followed for the blocks in the same arrival leg above, both turn conditions (turn on platform before departure and turn during the movement to shunt track) are checked again and if parking blocks $b1$ and $b2$ at the same shunt track will result in the wrong order after they are moved from shunt track s to the departure platform, they are added to the pair set.

$arrAfterDep_{b1,b2,s}$: Depending on the parameter mentioned in section 3.2 (how tightly in time blocks can be parked at the same shunt track), it has value 0 if $b1 \in B$ departs from the depot before $b2 \in B$ arrives at, and 1 otherwise. Having 1 in other words means that $b1$ is the block arriving earlier and there is such a time interval that both blocks are in the depot at the same time. Its behavior according to the given tight parking level parameter is:

$arrAfterDep_{b1,b2,s} =$

- Option1:
$$\begin{cases} 1, departure\ time\ of\ block\ b1 < arrival\ time\ of\ b2 \\ 0, otherwise \end{cases}$$
- Option2:
$$\begin{cases} 1, Earliest\ departure\ time\ of\ b1\ from\ s \in S\ < arrival\ time\ of\ b2 \\ 0, otherwise \end{cases}$$
- Option 3:
$$\begin{cases} 1, (Departure\ time\ of\ b1\ from\ s \in S\ < \\ \quad Latest\ arrival\ time\ of\ b2\ at\ its\ arrival\ platform) \\ 0, otherwise \end{cases}$$
- Option 4:
$$\begin{cases} 1, (Earliest\ departure\ time\ of\ b1\ from\ s \in S\ < \\ \quad Latest\ arrival\ time\ of\ b2\ at\ its\ arrival\ platform) \\ 0, otherwise \end{cases}$$

U_b: Penalty cost of not parking block $b \in B$ anywhere.
A: Penalty cost of a broken arrival.
D: Penalty cost of a broken departure.
$movementCost_{b,s}$: The total movement cost of parking block $b \in B$ at shunt track $s \in S$. It contains the driving costs from the arrival platform of b to $s \in S$ and from $s \in S$ to the departure platform of b.

DECISION VARIABLES:

The model has the following binary decision variables:

$$y_b = \begin{cases} 1, if\ block\ b \in B\ is\ not\ parked\ anywhere \\ 0, otherwise \end{cases}$$

$$blockTrack_{b,s} = \begin{cases} 1, if\ block\ b \in B\ is\ parked\ at\ s \in S \\ 0, otherwise \end{cases}$$

$$brokenArr_b = \begin{cases} 1, if\ block\ pairs\ (b, b') \in arrBlockPairs\ are\ broken \\ 0, otherwise \end{cases}$$

$$brokenDep_b = \begin{cases} 1, if\ block\ pairs\ (b, b') \in depBlockPairs\ are\ broken \\ 0, otherwise \end{cases}$$

OBJECTIVE FUNCTION:

The objective function minimizes the total cost of unparked blocks, broken arrivals, broken departures and the movement costs.

$$Minimize \begin{pmatrix} (\sum_{b \in B} U_b x\, y_b) + \\ \left(A \sum_{\substack{(b,b') \in \\ arrBlockPairs}} brokenArr_b \right) + \\ \left(D \sum_{\substack{(b,b') \in \\ depBlockPairs}} brokenDep_b \right) + \\ (\sum_{b \in B} \sum_{s \in S} blockTrack_{b,s}\ x\ movementCost_{b,s}) \end{pmatrix} \qquad (2.1)$$

CONSTRAINTS:

1. These constraints make sure that each block is either parked at a single shunt track or is not parked at all.

$$\sum_{s \in S} blockTrack_{b,s} + y_b = 1 \quad \forall b \in \{B - L\} \qquad (2.2)$$

2. The following constraints satisfy the locks. A block $b \in L$ can either be parked at its locked shunt track or it is not parked at all.

$$blockTrack_{b,locks_b} + y_b = 1 \quad \forall b \in L \qquad (2.3)$$

3. The constraints below satisfy the conditions when block $b \in B$ cannot be parked at $s \in S$ because there is not a driving connection between $s \in S$ and arrival platform of b, or between $s \in S$ and departure platform of b.

$$blockTrack_{b,s} = 0 \quad \forall b \in B, \forall s \in S\,, where\ b\ cannot\ be\ parked\ at\ s \quad (2.4)$$

4. Crossing constraints state that at most one block of each block pair in $crossingGeneral_s$ set where $s \in S$, can be parked at shunt track s. In other words they cannot be parked at the same shunt track s.

$$blockTrack_{b,s} + blockTrack_{b',s} \leq 1 \; \forall s \in S, \forall (b, b') \in crossingGeneral_s \; (2.5)$$

5. Constraints 2.6 state that if blocks arriving in the same leg cannot be parked together at shunt track $s \in S$ as connected units, parking both of them at that track will indicate that the blocks were detached at the platform first then moved separately to the track which results in a broken arrival.

$$blockTrack_{b,s} + blockTrack_{b',s} - brokenArr_b \leq 1 \qquad (2.6)$$
$$\forall s \in S, \forall (b, b') \in sameArrLegTcrossing_s$$

6. There can also be a crossing between blocks of the same arrival leg when they are detached at the platform and are parked separately. For instance suppose that blocks $b1$ and $b2$ are arriving in the same leg together and $b1$ is the one close to the depot that will be moved first. If the blocks are detached at the platform before the shunting operations and if departure of $b1$ is earlier than departure of $b2$,then parking these two blocks separately at the same shunt track will result in a crossing and should not be allowed. When the blocks are broken, then crossing conditions should be taken into consideration.

$$blockTrack_{b,s} + blockTrack_{b',s} + brokenArr_b \leq 2 \qquad (2.7)$$
$$\forall s \in S, \forall (b, b') \in arrBlockPairs, where\ b\ and\ b'\ have\ LIFO\ crossing$$

7. These constraints handle the situations when block pairs $(b1, b2) \in depBlockPairs$ will violate the ordering positions when parked at shunt tracks $s \in S$ together. If the departure orders are violated, then it means that even though the blocks are parked at the same shunt track $s \in S$, we still have a broken departure.

$$blockTrack_{b,s} + blockTrack_{b',s} - brokenDep_b \leq 1 \qquad (2.8)$$
$$\forall s \in S, \forall (b, b') \in sameDepLegTcrossing_s$$

8. Another situation to be considered in the model is checking whether some block $b \in B$ is parked between blocks $b1, b2$ where $(b1, b2) \in depBlockPairs$, at any shunt track $s \in S$. The number of broken departures is being minimized in the model in order to decrease the attachment operations done at the platforms. That's why we try to park blocks $b1$ and $b2$ at the same track so that attachments can be done there before the departure takes place. On the other hand, even if blocks $b1$ and $b2$ are parked at the same shunt track $s \in S$, this does not necessarily mean that these blocks will be attached at the shunt track. If there is some other block parked between $b1$ and $b2$ it is impossible to do the attachment operation. The situation is illustrated in Figure 9.

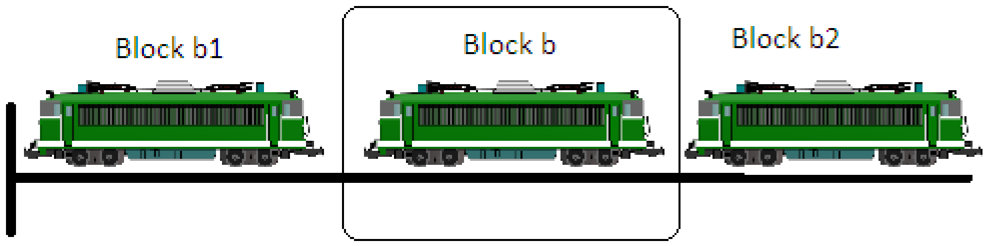


Figure 9-**Parking between blocks of same departure leg**

Suppose that $b1$ is the block arriving earlier (than $b2$), if a block $b \in B$ arrives at the platform after $b1$ but before $b2$ and parks at the same shunt track as $b1$, and departs at the same time with $b1$ and $b2$, depending on the tight parking levels mentioned before, it results in a broken departure. The blocks with different departure times than $b1$ and $b2$ are not needed to be checked since they are either handled in the crossing constraints or these blocks already depart from the depot before $b2$ arrives. If b was departing later than $b1$, it would be a crossing between blocks b and $b1$ and handled in constraints 2.5, if b was departing earlier than $b1, b2$ pair then it would depart before $b2$ arrives which would not lead to a broken departure, otherwise it would be a crossing between b and $b2$.

$$blockTrack_{b,s} + blockTrack_{b1,s} + blockTrack_{b2,s} - brokenDep_{b1} \leq 2 \ (2.9)$$
$$\forall s \in S, \forall (b1, b2) \in depBlockPairs, \forall b^*$$
$$* \, b \ arrives \ after \ b1 \ and \ before \ b2, and \ departs \ at \ the \ same \ time \ with \ them$$

9. The following constraints satisfy the capacity limitations on shunt tracks. At any time, the length of a shunt track $s \in S$ cannot be exceeded. For each block i and shunt track s, we traverse the blocks b that have arrived before i at s and sum their lengths if they have not departed yet when i has arrived at the depot. This summation should be less than or equal to the length of s.

$$\sum_{\substack{b \in B \text{ where } b \\ \text{arrives before } i}} blockTrack_{b,s} x\ lengthB_b\ xarrAfterDep_{b,i,s} \leq lengthS_s (2.10)$$

$$\forall s \in S, \forall i \in B$$

10. The constraints below force the value of $brokenArr_b$ be 1 if the pair $(b, b') \in arrBlockPairs$ is a broken arrival. The events resulting in a broken arrival for the basic model are shown in Table 7.

Table 7 -Events resulting in a broken arrival for the basic model

Event#	*block b*	*block b'*	Constraint satisfying the case
1	Parked at $s \in S$	Parked at $s' \in S$	2.11
2	Not parked at all	Parked at $s \in S$	2.12
3	Parked at $s \in S$	Not parked at all	2.11

10.1. The following constraints satisfy events 1 and 3.

$$brokenArr_b - blockTrack_{b,s} + blockTrack_{b',s} \geq 0 \qquad (2.11)$$
$$\forall s \in S, \forall (b, b') \in arrBlockPairs$$

10.2. Constraints 2.12 satisfy event 2.

$$brokenArr_b - y_b - \sum_{s \in S} blockTrack_{b',s} \geq -1 \qquad (2.12)$$
$$\forall (b, b') \in arrBlockPairs$$

11. Similar to the constraints above, these constraints below state that if there is a broken departure between pairs$(b, b') \in depBlockPairs$, then the $brokenDep_b$ value will be 1. The events that result in a broken departure are the same as the events above leading to a broken arrival for the basic model.

11.1. Constraints 2.13 cover events 1 and 3.

$$brokenDep_b - blockTrack_{b,s} + blockTrack_{b',s} \geq 0 \qquad (2.13)$$
$$\forall s \in S, \forall (b, b') \in depBlockPairs$$

11.2. Event 2 is covered by the following constraints.

$$brokenDep_b - y_b - \sum_{s \in S} blockTrack_{b',s} \geq -1 \qquad (2.14)$$
$$\forall (b, b') \in depBlockPairs$$

12. Finally, all the decision variables are binary.

$$y_b \qquad \in \{0,1\}\ \forall b \in B \qquad (2.15)$$
$$blockTrack_{b,s} \quad \in \{0,1\}\ \forall b \in B, \forall s \in S \qquad (2.16)$$

$$brokenArr_b \quad \in \{0,1\}\ \forall b \in B \tag{2.17}$$
$$brokenDep_b \quad \in \{0,1\}\ \forall b \in B \tag{2.18}$$

4.3. Model Extended To Include Platform Parkings (Model 2)

Having introduced the basic model, the next step is extending the model by including platform parkings. The main goal of the shunt planners in the parking stage is parking as many blocks as possible to the depot with the lowest consumption of time, energy and shunting operations. Shunt tracks were defined as tracks which are connected to at least one platform and are used for parking the blocks at when they are not operating. A platform was defined as a track where passengers get on/off the trains. Although the physical features of shunt tracks and platform tracks are almost the same, platform tracks were not considered as parking places. The main reason is that, platform tracks are much busier than shunt tracks since all the arrivals and departures take place from the platforms even if the blocks will not be parked at the depot, and the number of platform tracks is less than the number of shunt tracks. There are some security concerns as well which do not favor the use of platform tracks for parking, because they are the places where passengers are waiting.

Even though it is not favored to use platforms frequently, there are some cases in which parking at platforms is allowed with a certain penalty. For instance, a block arriving late at night to the depot and departing the first from its departure platform in the morning can be parked at its departure platform instead of a shunt track. There is no need to move that block to a shunt track first and retrieve it back in the morning.

Furthermore, for some platforms there are such time intervals in which no arrival and departure events take place. During these time intervals the platform is totally empty and there is no risk of parking a block. Whenever such a time interval is detected, only one block which departs the first from the platform after this time interval ends is allowed to park there. By doing so, the feasible region is being extended. The total number of blocks parked at the shunt yard can be increased if there is a capacity problem at the depot.

Although blocks arriving latest at the platforms and departing first from the platforms are preferred over others for platform parking, this is not always the case. The block departing first from platform p does not necessarily need to be the one arriving latest. Thus, we have two options for platform parking:

a. The block can directly be parked at its departure platform if it is the one arriving latest.

b. The block arrives at the depot, then first moves to a shunt track and parks there temporarily; afterwards it is moved to its departure platform and stays there till departure.

Firstly, in the system blocks which are allowed to park at their departure platforms at the expense of some penalty cost are identified. Then, for each block that can be parked at the platform via moving to depot first one more data entry is generated. This newly generated data entry is nothing but another block. The new block has the same features as the original one, e.g., the arrival time, arrival platform, departure platform, arrival leg, departure leg, type of train units in the block, length of the block, positions in the arrival and departure legs, except the departure time. All the properties of these two blocks are the same, but just their departure times. The departure time of the newly generated block is earlier than its original block's departure time, and it shows the time when that block can be moved from the shunt track it was parked at to the platform for the rest of its stay. This departure time is determined after analyzing the events in the timetable. The newly generated block will be called *variant*, and the block from which it was generated will be called its *original* block throughout the chapter.

For such blocks the model parks at most one of them: the original block or its variant. If the original block is chosen, it means that the block is parked at a shunt track like all other blocks. On the other hand, if the variant is chosen the block will have a platform parking. It will be parked at the depot first for a while, and then will be moved to the platform (a shunt track will be assigned first). If none is chosen the block is not parked at all.

The sets are modified in the following way for model 2:

NEW SETS INTRODUCED IN MODEL 2:

Set of all blocks B is divided into three different sets and replaced with sets OB, VB and NB.
OB: Set of original blocks of the variants in the problem.
VB: Set of variant blocks in the problem.
NB: Set of normal blocks in the problem. Normal blocks are the blocks that cannot be parked at a platform or the ones that can directly be parked at platforms without moving to depot first.

NEW AND MODIFIED NOTATIONS IN MODEL 2:

P_b: Penalty cost of parking block $b \in \{NB \cup OB\}$ at its departure platform.
$varOfOrg_b$: The variant of original block b.
$orgOfVar_b$: The original of variant block b.

$arrBlockPairs$: Pair of blocks $b1$ and $b2$ belonging to the same arrival leg where $b2$ is positioned right after $b1$. For each original block in any pair, the same pair but this time the variant block replacing its original will be added to the set as well. For instance suppose that blocks $b1$, $b2$, $b3$ and $b4$ arrive in the same leg in that order and block $b2$ is an original block, then the pairs will be ($b1$,$b2$), ($b1$,$varOfOrg_{b2}$), ($b2$,$b3$), ($varOfOrg_{b2}$,$b3$) and ($b3$,$b4$). Note that in model 2, legs consist of at most two blocks.

$depBlockPairs$: Modified similar to $arrBlockPairs$. It will now contain the variants as well. Note that a variant and a normal block or a variant and an original block cannot be in the same departure leg, since the departure time of a variant indicates the time it will be moved from the shunt yard to be parked at the platform, not the departure time from the depot. On the other hand, two different variants can form a leg.

$crossingGeneral_s$: The variants are also included in the crossing sets. Note that there is not a crossing between the variant of a block and its own original.

$sameArrLegTcrossing_s$: Blocks $b1$ and $b2$ where $(b1, b2) \in arrBlockPairs$ that include the variants this time, are checked with each other whether they can be parked together at shunt track $s \in S$ or not, and the set contains the ones which cannot be parked together as connected units.

$sameDepLegTcrossing_s$: By considering the variants as well the sets are extended in the same way as $sameArrLegTcrossing_s$.

<u>New And Modified Decision Variables In Model 2:</u>

Model 2 has the following binary decision variables:

$$y_b = \begin{cases} 1, if\ block\ b \in \{NB \cup OB\}\ is\ not\ parked\ anywhere \\ 0, otherwise \end{cases}$$

$$blockTrack_{b,s} = \begin{cases} 1, if\ block\ b \in \{NB \cup OB \cup VB\}\ \ is\ parked\ at\ s \in S \\ 0, otherwise \end{cases}$$

$$blockPpark_b = \begin{cases} 1, if\ block\ b \in \{NB \cup OB\}\ is\ parked\ at\ its\ departure\ platform \\ 0, otherwise \end{cases}$$

$$brokenArr_b = \begin{cases} 1, if\ (b, b') \in arrBlockPairs\ where\ b \in \{NB \cup OB\}\ is\ broken \\ 0, otherwise \end{cases}$$

$$brokenDep_b = \begin{cases} 1, if\ (b, b') \in depBlockPairs\ where\ b \in \{NB \cup OB\}\ is\ broken \\ 0, otherwise \end{cases}$$

O<span style="font-variant:small-caps"></span>BJECTIVE FUNCTION OF MODEL 2:

The new objective function minimizes the total cost of unparked blocks, broken arrivals, broken departures, movement costs and platform parking costs. Objective function 2.1 is replaced with 3.1.

$$Minimize \begin{pmatrix} \left(\sum_{b \in \{NB \cup OB\}} U_b x\, y_b + P_b\, x\, blockPpark_b\right) + \\ \left(A \sum_{\substack{(b,b') \in \\ arrBlockPairs}} brokenArr_b\right) + \\ \left(D \sum_{\substack{(b,b') \in \\ depBlockPairs}} brokenDep_b\right) + \\ \left(\sum_{b \in \{NB \cup OB \cup VB\}} \sum_{s \in S} blockTrack_{b,s}\ x\ movementCost_{b,s}\right) \end{pmatrix} \tag{3.1}$$

CONSTRAINTS MODIFIED FOR MODEL 2:

1. In this extended model each block can either be parked at a single shunt track or it can be parked at its departure platform (if allowed to), otherwise it is not parked at all. Constraints 2.2 are replaced with constraints 3.2.

$$\sum_{s \in S} (blockTrack_{b,s}) + y_b + blockPpark_b = 1 \qquad \forall b \in \{NB \cup OB - L\}\ (3.2)$$

2. The lock constraints are the same as before. A block $b \in L$ where $L \in \{OB \cup NB\}$ can either be parked at its locked shunt track or it is not parked at all.

$$blockTrack_{b,locks_b} + y_b = 1 \ \ \forall b \in L\ (2.3)$$

3. Constraints 2.4 are replaced with constraints 3.4 which satisfy the conditions of not having a driving connection between $s \in S$ and arrival platform of b, or between $s \in S$ and departure platform of b, where $b \in \{NB \cup OB \cup VB\}$.

$$blockTrack_{b,s} = 0 \qquad (3.4)$$
$$\forall b \in \{NB \cup OB \cup VB\}, \forall s \in S, where\ b\ cannot\ be\ parked\ at\ s$$

4. Crossing constraints, constraints 2.5, will basically remain the same. Just the sets $crossingGeneral_s$ are modified.

$$blockTrack_{b,s} + blockTrack_{b',s} \leq 1\ \forall s \in S, \forall (b,b') \in crossingGeneral_s\ (3.5)$$

5. If blocks arriving in the same leg cannot be parked together at shunt track $s \in S$ as connected units, the constraints below force them have a broken arrival when they

are parked at s so that they will be detached at the platform first and be parked separately. The sets $sameArrLegTcrossing_s$ were extended to include the variants. Constraints 2.6 should be modified for this extension. The variants do not have $brokenArr_b$ decision variables, so if there is a broken arrival between (b, b') where b is a variant, $brokenArr_{orgOfVar_b}$ should be used instead.

$$blockTrack_{b,s} + blockTrack_{b',s} - brokenArr_b \leq 1 \quad (3.6.1)$$
$$\forall s \in S, \forall (b, b') \in sameArrLegTcrossing_s \; where \; b \in \{NB \cup OB\}$$

$$blockTrack_{b,s} + blockTrack_{b',s} - brokenArr_{orgOfVar_b} \leq 1 \quad (3.6.2)$$
$$\forall s \in S, \forall (b, b') \in sameArrLegTcrossing_s \; where \; b \in \{VB\}$$

6. These constraints satisfy the cases when blocks of the same arrival leg are detached at the platform, but parking them separately at the same shunt track leads to a crossing between them. $arrBlockPairs$ set was extended to include the variants. A similar procedure as above will be applied to constraints 2.7 and will be replaced with constraints 3.7.1 and 3.7.2.

$$blockTrack_{b,s} + blockTrack_{b',s} + brokenArr_b \leq 2 \quad (3.7.1)$$
$$\forall s \in S, \forall (b, b') \in arrBlockPairs$$
$$where \; b \; and \; b' have \; LIFO \; crossing \; and \; b \in \{NB \cup OB\}$$

$$blockTrack_{b,s} + blockTrack_{b',s} + brokenArr_{orgOfVar_b} \leq 2 \quad (3.7.2)$$
$$\forall s \in S, \forall (b, b') \in arrBlockPairs$$
$$where \; b \; and \; b' have \; LIFO \; crossing \; and \; b \in \{VB\}$$

7. The extended version of constraints 2.8 will use the new $sameDepLegTcrossing_s$ sets. Similar to the extension done in constraints 3.6.1 and 3.6.2, we have the following modified constraints:

$$blockTrack_{b,s} + blockTrack_{b',s} - brokenDep_b \leq 1 \quad (3.8.1)$$
$$\forall s \in S, \forall (b, b') \in sameDepLegTcrossing_s \; where \; b \in \{NB \cup OB\}$$

$$blockTrack_{b,s} + blockTrack_{b',s} - brokenDep_{orgOfVar_b} \leq 1 \quad (3.8.2)$$
$$\forall s \in S, \forall (b, b') \in sameDepLegTcrossing_s \; where \; b \in \{VB\}$$

8. Constraints 2.9 are changed to the followings:

$$blockTrack_{b,s} + blockTrack_{b1,s} + blockTrack_{b2,s} - brokenDep_{b1} \leq 2 \; (3.9.1)$$
$$\forall s \in S, \forall (b1, b2) \in depBlockPairs \; and \; b1 \in \{NB \cup OB\}, \forall b^*$$

$$blockTrack_{b,s} + blockTrack_{b1,s} + blockTrack_{b2,s} - brokenDep_{orgOfVar_{b1}} \leq 2$$
$$\forall s \in S, \forall (b1, b2) \in depBlockPairs \; and \; b1 \in \{VB\}, \forall b^* \quad (3.9.2)$$
$$* \, b \; arrives \; after \; b1 \; and \; before \; b2, and \; departs \; at \; the \; same \; time \; with \; them$$

9. The capacity constraints will remain the same. The only change is that we will have the union of sets $\{NB \cup OB \cup VB\}$ in the summation instead of B.

$$\sum_{\substack{b \in \{NB \cup OB \cup VB\} \, where \, b \\ arrives \, before \, i}} blockTrack_{b,s} x lengthB_b x arrAfterDep_{b,i,s} \leq lengthS_s$$
$$\forall s \in S, \forall i \in \{NB \cup OB \cup VB\} \quad (3.10)$$

10. Addition of platform parkings to the problem changes the cases for broken arrival situations. Previously we had two conditions for each block: either it was parked at a shunt track or it was not parked at all. Introduction of platform parkings not only added one more condition, but also slightly changed the broken arrival cases because the way platform parking is done also has an impact and will be of interest.
Depending on the types of blocks in the pairs $(b, b') \in arrBlockPairs$, the cases will be analyzed separately. We have normal blocks, original blocks and variant blocks in the problem. As explained before, an original block and its variant basically represent the same block. Original block is used if the block is parked at a shunt track directly; on the other hand, using its variant means that the block will be parked at its departure platform via depot. Thus, a shunt track for temporary parking must be chosen first, it is where the variant plays a role. In result, for each block pair (b, b') we have four cases:
    1. b and b' are both normal blocks.
    2. b is an original/variant block and b' is a normal block.
    3. b is a normal block and b' is an original/variant.
    4. b and b' are both original/variant blocks.
For each case the situations resulting in a broken arrival will be analyzed, and constraints satisfying these situations will be shown in tables. The events which have the same constraints for all four cases i.e., that do not depend on the type of the blocks are provided in Table 8.

Table 8 **-Events leading to a broken arrival that do not depend on the types of the blocks and the corresponding constraints**

Event #	Block b	Block b'	Constraint satisfying the event
1	Not parked at all	Parked at $s \in S$	3.11
2	Not parked at all	Platform parked at p	3.11

10.1. Constraints 3.11 satisfy events 1 and 2

$$brokenArr_b - y_b + y_{b'} \geq 0 \quad (3.11)$$
$$\forall (b, b') \in arrBlockPairs \; where \; b, b' \in \{NB \cup OB\}$$

Case 1: Blocks b and b' are both normal blocks.

Table 9 -Events leading to a broken arrival for Case 1 and the corresponding constraints

Event #	Block b-normal block	Block b'-normal block	Constraint satisfying the event
3	Parked at $s \in S$	Parked at $s' \in S$	3.12
4	Parked at $s \in S$	Not parked at all	3.12
5	Parked at $s \in S$	Platform parked at p	3.12
6	Platform parked at p	Platform parked at p'	3.13
7	Platform parked at p	Parked at $s \in S$	3.14
8	Platform parked at p	Not parked at all	3.15

10.2. Constraints 3.12 satisfy events 3, 4 and 5

$$brokenArr_b - blockTrack_{b,s} + blockTrack_{b',s} \geq 0 \quad (3.12)$$
$$\forall s \in S, \forall (b, b') \in arrBlockPairs \; where \; b, b' \in NB$$

10.3. Constraints 3.13 satisfy event 6

$$brokenArr_b - blockPpark_b - blockPpark_{b'} \geq -1 \quad (3.13)$$
$$\forall (b, b') \in arrBlockPairs \; where \; b, b' \in NB \; and$$
$$depart \; from \; different \; platforms$$

10.4. Constraints 3.14 satisfy event 7

$$brokenArr_b - blockPpark_b + y_{b'} + blockPpark_{b'} \geq 0 \quad (3.14)$$
$$\forall (b, b') \in arrBlockPairs \; where \; b, b' \in NB$$

10.5. Constraints 3.15 satisfy event 8

$$brokenArr_b - blockPpark_b - y_{b'} \geq -1 \quad (3.15)$$
$$\forall (b, b') \in arrBlockPairs \; where \; b, b' \in NB$$

Case 2: b is an original block and b' is a normal block.

Table 10 -Events leading to a broken arrival for Case 2 and the corresponding constraints

Event #	Block b-original block	Block b'-normal block	Constraint satisfying the event
9	Parked at $s \in S$	Parked at $s' \in S$	3.16
10	Parked at $s \in S$	Not parked at all	3.16
11	Parked at $s \in S$	Platform parked at p	3.16
12	Platform parked at p via $s \in S$	Platform parked	3.16
13	Platform parked at p via $s \in S$	Parked at $s' \in S$	3.16
14	Platform parked at p via $s \in S$	Not parked at all	3.16

Events 12 and 13 in case 2 are slightly different than events 6 and 7 of case 1. If both blocks b and b' are normal blocks (case 1) then parking them at different platforms will result in a broken arrival but not if we park them at the same platform. On the other hand if b is an original block and b' is a normal block, even if we park them both at the same platform it will still be a broken arrival since b is an original block and will be moved to depot first before parking at the platform.
Parking b at platform and b' at a shunt track is always a broken arrival for case 1, but if b is an original block and b' is a normal block then parking b at platform via parking at shunt track $s \in S$ first and parking b' at s will not lead to a broken arrival.

10.6. Constraints 3.16 satisfy events 9, 10, 11, 12, 13 and 14

$$brokenArr_b - \left(blockTrack_{b,s} + blockTrack_{varOfOrgb,s}\right) + blockTrack_{b',s} \geq 0$$
$$\forall s \in S, \forall (b,b') \in arrBlockPairs \text{ where } b \in OB, b' \in NB \qquad (3.16)$$

Case 3: b is a normal block and b' is an original block.

Table 11 **-Events leading to a broken arrival for Case 3 and the corresponding constraints**

Event #	Block b-normal block	Block b'-original block	Constraint satisfying the event
15	Parked at $s \in S$	Parked at $s' \in S$	3.17
16	Parked at $s \in S$	Not parked at all	3.17
17	Parked at $s \in S$	Platform parked at p via $s' \in S$	3.17
18	Platform parked	Platform parked at p via $s \in S$	3.18
19	Platform parked	Parked at $s \in S$	3.18
20	Platform parked	Not parked at all	3.18

10.7. Constraints 3.17 satisfy events 15, 16 and 17

$$brokenArr_b - blockTrack_{b,s} + (blockTrack_{b',s} + blockTrack_{varOfOrgb',s}) \geq 0$$
$$\forall s \in S, \forall (b,b') \in arrBlockPairs \text{ where } b \in NB, b' \in OB \qquad (3.17)$$

10.8. Events 18, 19 and 20 are satisfied by Constraints 3.18

$$brokenArr_b - blockPpark_b \geq 0 \qquad (3.18)$$
$$\forall (b,b') \in arrBlockPairs \text{ where } b \in NB, b' \in OB$$

Case 4: Blocks b and b' are both original blocks.

In this case, if we park the blocks at platforms we need to check where their variants were moved at the depot first, because both blocks are original blocks. For instance in event 24, if both blocks will be parked at platforms we are not interested in which platforms they will be parked at, instead we need to check where in depot their variants will be parked temporarily.

Table 12 -Events leading to a broken arrival for case 4 and the corresponding constraints

Event #	Block b-original block	Block b'-original block	Constraint satisfying the event
21	Parked at $s \in S$	Parked at $s' \in S$	3.19
22	Parked at $s \in S$	Not parked at all	3.19
23	Parked at $s \in S$	Platform parked at p via $s' \in S$	3.19
24	Platform parked at p via $s \in S$	Platform parked at p via $s' \in S$	3.19
25	Platform parked at p via $s \in S$	Parked at $s' \in S$	3.19
26	Platform parked at p via $s \in S$	Not parked at all	3.19

10.9. The constraints 3.19 below satisfy events 21, 22, 23, 24, 25 and 26

$$brokenArr_b - (blockTrack_{b,s} + blockTrack_{varOfOrg_b,s}) + (blockTrack_{b',s} + blockTrack_{varOfOrgb',s}) \geq 0$$
$$\forall s \in S, \forall (b, b') \in arrBlockPairs\ where\ b \in OB, b' \in OB \qquad (3.19)$$

11. The cases for broken departures are changed in Model 2; here the events leading to a broken departure do not depend on the type of the blocks except event 4.

Table 13 -Events leading to a broken departure and the corresponding constraints

Event #	Block b	Block b'	Constraint satisfying the event
1	Parked at $s \in S$	Parked at $s' \in S$	3.20
2	Parked at $s \in S$	Not parked at all	3.20
3	Parked at $s \in S$	Platform parked	3.20
4	Platform parked at p	Platform parked at p	3.21.1, 3.21.2, 3.21.3
5	Platform parked	Parked at $s \in S$	3.22
6	Platform parked	Not parked at all	3.23
7	Not parked at all	Parked at $s \in S$	3.24
8	Not parked at all	Platform parked at p	3.24

11.1. Constraints 3.20 satisfy events 1, 2 and 3

$$brokenDep_b - blockTrack_{b,s} + blockTrack_{b',s} \geq 0 \qquad (3.20)$$
$$\forall s \in S, \forall (b, b') \in depBlockPairs\ where\ b, b' \in \{NB \cup OB\}$$

11.2. Constraints 3.21 satisfy the cases when both blocks b and b' are parked at platforms. These types of constraints depend on the types of the blocks.

- If $b \in NB$ and $b' \in NB$: parking both of them at platforms do not result in a broken departure, so no constraint is needed.
- If $b \in NB$ and $b' \in OB$ or $b \in OB$ and $b' \in NB$: one of the blocks is a normal block that will directly be parked at a platform, and the other one is an original block which will first park at a shunt track then will move to the platform. Parking both of them at platforms results in a broken departure.

$$brokenDep_b - blockPpark_b - blockPpark_{b'} \geq -1 \quad (3.21.1)$$
$$\forall (b, b') \in depBlockPairs\ where\ b\ \in OB, b'\ \in NB\ or\ b\ \in NB, b'\ \in OB$$

- If $b \in OB$ and $b' \in OB$: When both blocks are original blocks, whether parking them at platforms results in a broken departure or not depends on the variants. If the variants are parked at different shunt tracks or if the departure times of the variants are not the same, but both variants were parked, then we have a broken departure.

$$brokenDep_b - blockTrack_{varOfOrg_b,s} + blockTrack_{varOfOrgb',s} \geq 0 \quad (3.21.2)$$
$$\forall s \in S, \forall (b, b') \in depBlockPairs\ where\ b, b'\ \in OB$$

$$brokenDep_b - blockPpark_b - blockPpark_{b'} \geq -1 \quad (3.21.3)$$
$$\forall (b, b') \in depBlockPairs\ where\ b, b'\ \in OB\ and$$
$$dep.time\ of\ varOfOrg_{b'}! =\ dep.time\ of\ varOfOrg_b$$

11.3. Event 5 is satisfied by constraints 3.22

$$brokenDep_b - blockPpark_b + y_{b'} + blockPpark_{b'} \geq 0 \quad (3.22)$$
$$\forall (b, b') \in depBlockPairs\ where\ b, b'\ \in \{NB \cup OB\}$$

11.4. Constraints 3.23 satisfy event 6

$$brokenDep_b - y_{b'} - blockPpark_b \geq -1 \quad (3.23)$$
$$\forall (b, b') \in depBlockPairs\ where\ b, b'\ \in \{NB \cup OB\}$$

11.5. Events 7 and 8 are satisfied by constraints 3.24

$$brokenDep_b - y_b + y_{b'} \geq 0 \quad (3.24)$$
$$\forall (b, b') \in depBlockPairs\ where\ b, b'\ \in \{NB \cup OB\}$$

12. We know that original blocks can be parked at platforms via variants, and normal blocks can either be parked at platforms directly or they cannot be parked there at all. Constraints 3.25 state that blocks which are not allowed to park at platforms cannot be parked. Constraints 3.26 imply that if a variant of an original block is parked at a shunt track the original block is parked at platform.

$$blockPpark_b = 0 \quad (3.25)$$
$$\forall b \in NB\ where\ b\ is\ not\ allowed\ for\ platform\ parking$$

$$blockPpark_b - \sum_{s \in S} blockTrack_{varOfOrg_b,s} = 0\ \forall b \in OB \quad (3.26)$$

13. Finally, all the decision variables are binary.

$y_b \in\{0,1\}\ \forall b \in \{NB \cup OB\}$ (2.15)

$blockTrack_{b,s} \in\{0,1\}\ \forall b \in \{NB \cup OB \cup VB\}, \forall s \in S$ (3.27)

$brokenArr_b \in\{0,1\}\ \forall b \in \{NB \cup OB\}$ (2.17)

$brokenDep_b \in\{0,1\}\ \forall b \in \{NB \cup OB\}$ (2.18)

$blockPpark_b \in\{0,1\}\ \forall b \in \{NB \cup OB\}$ (3.28)

4.4. Model Extended To Include More Than Two Blocks in a Leg (Model 3)

In the previous models arrival/departure legs consisting of more than two blocks were not considered. Model 3 includes this extension and allows any number of blocks to form a leg. Some notations are introduced and some are modified as follows:

$arrPos_b$: denotes the position of block b in the arrival leg.
$depPos_b$: denotes the position of block b in the departure leg.
$arrLeg_b$: denotes the arrival leg of block b.
$depLeg_b$: denotes the departure leg of block b.

$arrBlockPairsConsecutive$: Pairs of blocks (b, b') where $b\ , b' \in \{NB \cup OB \cup VB\}$ arriving in the same leg and b is positioned just before b'.

$depBlockPairsConsecutive$: Pairs of blocks (b, b') where$b, b' \in \{NB \cup OB \cup VB\}$ departing in the same leg and b is positioned just before b'.

$allArrPairs$: All pairs of blocks (b, b') -not just the consecutive ones- where $b\ and\ b'$ are arriving in the same leg and b is positioned before b'.

$sameArrLegTcrossing_s$: Pairs of blocks $(b\ b') \in arrBlockPairsConsecutive$ that will have a crossing between when parked together at shunt track $s \in S$ as connected units.

$sameDepLegTcrossing_s$: Pairs of blocks $(b\ b') \in depBlockPairsConsecutive$ that will have a conflict in their departure positions when parked together at the same shunt track $s \in S$.

We have the same decision variables as in model 2. The objective function now iterates over $arrBlockPairsConsecutive$ set for broken arrival penalties and over $depBlockPairsConsecutive$ set for broken departure penalties. A point to note is that for each arrival leg a, the sum of the values of decision variables $brokenArr_b$ where $(b, b') \in arrBlockPairsConsecutive$, $b \in \{NB \cup OB\}$ and, $b$and b'belong to leg a, gives us the total number of detachments made at the platform for leg a, in other words number of costly operations made per arrival leg.

Similarly, for each departure leg d, the sum of $brokenDep_b$ values where $(b, b') \in depBlockPairsConsecutive$, $b \in \{NB \cup OB\}$ and, b and b' belong to leg d, gives us the total number of attachments made at the platform for leg d.

CONSTRAINTS MODIFIED FOR MODEL 3:

Constraints 3.7.1 and 3.7.2 are extended to constraints 4.7.1 and 4.7.2 which state that blocks $b\ and\ b'$ arriving in the same leg that will result in a crossing when parked separately, are not allowed to park at the same track if there occurred a detachment somewhere between the blocks. Put differently, the new constraints satisfy the cases when detaching the blocks b and b' somehow from each other leads to a crossing between them. A sample leg is depicted in Figure 10. There are five blocks in the leg and two of them have variants.

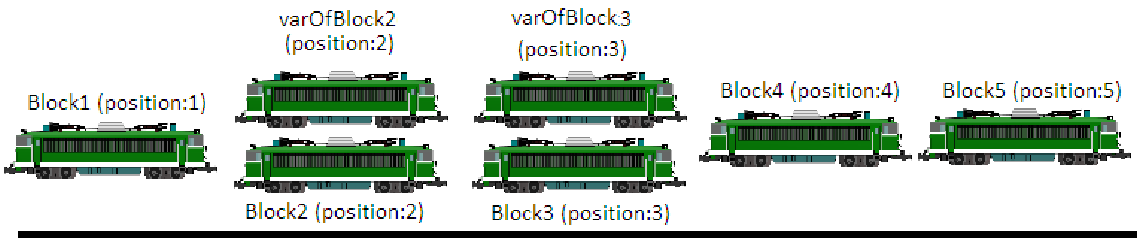


Figure 10 -A leg also showing the variants as options

For all pairs $(b, b') \in allArrPairs$, blocks b and b' will be checked with each other whether there occurs a crossing between them when they park separately. Blocks b and b' have a crossing between if the blocks are turning on the platform (see Figure 7) before the movement starts and the block positioned later e.g., block b', departs earlier than b or if the blocks are not turning on the platform but b departs earlier than b'. If the blocks are broken apart and there is a crossing, then they should not park at the same track. For the pairs $(block1,\ block4)$ in Figure 10, a brake can occur just after $block1$ or $block2$ or $block3$. Note that $brokenArr_b$ represents the break between $blockb$ and $block\ (b+1)$.

$$blockTrack_{b,s} + blockTrack_{b',s} + brokenArr_{b''} \leq 2 \qquad (4.7.1)$$
$$\forall s \in S, \forall (b, b') \in allArrPairs\ where\ b\ and\ b'\ have\ LIFO\ crossing\ at\ s,$$
$$\forall\ b'' \in \{OB\ \cup NB\}\ where\ arrLeg_{b''} =\ arrLeg_b\ and$$
$$arrPos_{b'} > arrPos_{b''} \geq arrPos_b\ and\ b''\ != orgOfVar_{b'}$$

$$blockTrack_{b,s} + blockTrack_{b',s} + brokenArr_{orgOfVar_{b''}} \leq 2 \quad (4.7.2)$$
$$\forall s \in S, \forall (b, b') \in allArrPairs\ where\ b\ and\ b'\ have\ LIFO\ crossing\ at\ s,$$
$$\forall\ b'' \in \{VB\}\ where\ arrLeg_{b''} =\ arrLeg_b and$$
$$arrPos_{b'} > arrPos_{b''} \geq arrPos_b\ and\ b''\ != varOfOrg_{b'}$$

On the other hand, for the blocks arriving in the same leg that cannot be parked together as connected units at shunt track $s \in S$, the sets $sameArrLegTcrossing_s$are extended as mentioned above. The constraints 3.6.1 and 3.6.2 below force them have a broken arrival when they are parked at the same track. Here we are not checking all the pairs in a leg, instead just the consecutive pairs, because if block 1 and block 4 cannot be parked together at a shunt track, the reason will be one of the following consecutive pairs: $(block1,\ block2)$, $(block2,\ block3)$ or $(block3,\ block4)$.

$$blockTrack_{b,s} + blockTrack_{b',s} - brokenArr_b \leq 1 \qquad (3.6.1)$$
$$\forall s \in S, \forall (b, b') \in sameArrLegTcrossing_s \ where \ b \in \{NB \cup OB\}$$

$$blockTrack_{b,s} + blockTrack_{b',s} - brokenArr_{orgOfVar_b} \leq 1 \quad (3.6.2)$$
$$\forall s \in S, \forall (b, b') \in sameArrLegTcrossing_s \ where \ b \in \{VB\}$$

Constraints 3.8.1 and 3.8.2 which satisfy the crossing situations between consecutive blocks of the same departure leg will remain the same, but will iterate over the new $sameDepLegTcrossing_s$sets.

$$blockTrack_{b,s} + blockTrack_{b',s} - brokenDep_b \leq 1 \quad (3.8.1)$$
$$\forall s \in S, \forall (b, b') \in sameDepLegTcrossing_s \ where \ b \in \{NB \cup OB\}$$

$$blockTrack_{b,s} + blockTrack_{b',s} - brokenDep_{orgOfVar_b} \leq 1 \quad (3.8.2)$$
$$\forall s \in S, \forall (b, b') \in sameDepLegTcrossing_s \ where \ b \in \{VB\}$$

Constraints 3.9.1 and 3.9.2 which state that for all pairs of blocks departing in the same leg and positioned next to each other, if some other block with the same departure time is parked between them there occurs a broken departure, will remain the same but using the $depBlockPairsConsecutive$ set.

$$blockTrack_{b,s} + blockTrack_{b1,s} + blockTrack_{b2,s} - brokenDep_{b1} \leq 2 \quad (3.9.1)$$
$$\forall s \in S, \forall (b1, b2) \in depBlockPairsConsecutive \ and \ b1 \in \{NB \cup OB\}, \forall b^*$$

$$blockTrack_{b,s} + blockTrack_{b1,s} + blockTrack_{b2,s} - brokenDep_{orgOfVar_{b1}} \leq 2$$
$$\forall s \in S, \forall (b1, b2) \in depBlockPairsConsecutive \ and \ b1 \in \{VB\}, \forall b^* \qquad (3.9.2)$$
$$*\, b \ arrives \ after \ b1 \ and \ before \ b2, and \ departs \ at \ the \ same \ time \ with \ them$$

4.5. Model Extended To Include Costs for Mixed Type of Blocks (Model 4)

The last extension is considering the types of blocks parked next to each other, and will be referred as Model 4. Some train units may be electric trains that need a catenary at the parked track, whereas some may be diesel trains and need to charge batteries at the shunt track or the train units may differ in some other way. In literature this extension was also mentioned. Lentink [2], grouped different types of train units with similar features into families, and some families were not allowed to park at some tracks which do not have the necessary equipment required. The number of different types of families parked at the tracks is also minimized.

In *DSB*, there are different types of train units with different features resulting in varieties within blocks, but since for all blocks there are necessary equipment at all tracks we do not group the types into families. If this was not the case, certain blocks would be prohibited from parking at some certain shunt tracks. This extension in our case does not violate the feasibility conditions; it is a soft constraint affecting the cost.

Some of the papers in literature minimize the total number of different types of blocks parked at shunt tracks. On the other hand, in our case the hardest part of this extension is that *DSB* prefers solutions in which blocks parked at a shunt track next to each other any time are of the same type. Thus, the order of blocks parked should also be considered. For instance, suppose that there are two different types: $type1$ and $type2$. Parking the blocks in $type1$- $type1$- $type2$ order results in one penalty cost, but parking them in $type1$-$type2$-$type1$ order incurs two penalty costs. The new decision variables introduced and the final objective function are as follows:

$$mixture_b = \begin{cases} 1, if\ block\ b\ is\ parked\ next\ to\ b'\ \ where\ b, b' \in \{NB \cup OB \cup VB\}, \\ \quad b\ arrives\ earlier\ than\ b\ , and\ b\ and\ b' are\ of\ different\ \ types \\ 0, otherwise \end{cases}$$

$$Minimize \left(\begin{array}{c} \left(\sum_{b \in \{NB \cup OB\}} U_b x\, y_b + P_b x blockPpark_b \right) + \\ \left(A \sum_{\substack{(b,b') \in \\ arrBlockPairsConsecutive}} brokenArr_b \right) + \\ \left(D \sum_{\substack{(b,b') \in \\ depBlockPairsConsecutive}} brokenDep_b \right) + \\ \left(\sum_{b \in \{NB \cup OB \cup VB\}} \sum_{s \in S} blockTrack_{b,s}\ x\ movementCost_{b,s} \right) + \\ \left(M \sum_{b \in \{NB \cup OB \cup VB\}} mixture_b \right) \end{array} \right) \quad (5.1)$$

Constraints 5.29 below are added to the model:

$$blockTrack_{b1,s} + blockTrack_{b2,s} - mixture_{b1} - \sum_{b3*} blockTrack_{b3,s} \leq 1 \qquad (5.29)$$

$$\forall s \in S, \forall b1, b2 \in \{NB \cup OB \cup VB\}$$

$$where\ b1, b2\ are\ in\ depot\ at\ the\ same\ time\ and\ do\ not\ have\ a\ crossing\ at\ s,$$

$$and\ are\ of\ different\ types$$

$$*\ b3's\ arrival\ time\ is\ between\ b1\ and\ b2's\ arrival\ times$$

5. Results

The purpose of the thesis is improving the current system at *Jeppesen* developed for *DSB* by improving the parking solution methodology. In the current system it was seen that for some large real life instances it was impossible to find a feasible solution to the parking problem in a reasonable amount of time, especially in Hillerød depot where we have the most complex station topology.

To reach the goal, the disadvantages of the current system were analyzed and considering the requirements from the customer a new model was developed. Firstly, a basic model was introduced, some extensions were implemented later and finally all the requirements from *DSB* were covered and all the instances were solved. For small instances both models are head to head, but for medium and large instances the new model is much faster than the current one. Compared to the old model, all the feasibility rules and the rules which do not affect the feasibility but just the cost of the solution are modeled as constraints. In the old model, prior to solving the problem all possible assignments for each shunt track were generated, thus the cost calculations and the feasibility conditions were being included implicitly. Although having this advantage in the model, the exponential number of decision variables made the problem unsolvable for larger instances. The new model seems specific to the needs of the customer and since the customer does not have any free tracks –tracks open on both sides-, the new model does not include this extension for now. It was seen that for some instances the last extension with mixed type of blocks constraints are the most challenging ones for the new model. This extension is a soft constraint which does not affect the feasibility of the problem but just the cost of the solution.

The models were tested on 71 instances. Each instance contains a set of different depots to be solved one by one. In 34 of the instances, which are small cases, the models were head to head, thus they are not included in the tables below. The parking problem and the scheduling problem are solved in sequence and the data in the tables are the total measures for both problems.

Table 14 below compares the final MIP model (model 4) and the current model used in the *Jeppesen* system both in terms of memory usage and the running times; the ratios (model 4/current model) less than 1.0 indicate an improvement, the blank fields are where the models are head to head, and the ratios greater than 1.0 show a weakening.

Table 14- Comparison of the current system and the final MIP model
*Optimal solution was not obtained in 10 minutes, thus the test was terminated.

Test #	Memory (mb.)			Performance (sec.)		
	Current model	Model 4	Ratio (m4 / c.m.)	Current model	Model 4	Ratio (m4 / c.m.)
1	-	-	-	111,5	*600	5.3811659
2	275	213	0.774545455	25,2	4,8	0.1904762
3	359	250	0.69637883	311	12,5	0.0401929
4	282	213	0.755319149	26,3	4,5	0.1711027
5	250	212	0.848	15,2	3,5	0.2302632
6	250	212	0.848	-	-	-
7	491,63	322	0.654964099	109,7	27,6	0.2515953
8	-	-	-	19,1	16,3	0.8534031
9	-	-	-	-	-	-
10	527	237	0.44971537	138,7	24,1	0.1737563
11	-	-		6,1	21,6	3.5409836
12	500	239	0.478	118	35,3	0.2991525
13	550	490	0.890909091	43,4	30,4	0.7004608
14	-	-	-	2,7	25,5	9.4444444
15	-	-	-	24,45	29,2	1.194274
16	500	459	0.918	17	8,3	0.4882353
17	1048	496	0.473282443	145,55	53,5	0.3675713
18	528	471	0.892045455	28,3	9,4	0.3321555
19	580	487	0.839655172	19,1	14,7	0.7696335
20	-	-	-	17,9	10,5	0.5865922
21	590	510	0.86440678	75,3	23,5	0.312085
22	-	-	-	39,32	9,5	0.2416073
23	533	492	0.923076923	-	-	-
24	-	-	-	40,2	9,6	0.238806
25	3878	491	0.126611655	788,6	8,9	0.0112858
26	527	472	0.895635674	29,2	7,5	0.2568493
27	298	209	0.701342282	20,8	1,9	0.0913462

As seen in table 14 above, for almost all the cases there is an improvement both in running times and in memory usage, but in four of the test instances in which three of them are medium sized cases, the old model was better in terms of performance. In the first test case it was seen that the bottleneck of the MIP model was the huge number of constraints incurred from mixed type of block extension as a consequence of the structure of the timetable. Since this extension is a soft constraint, we decided to exclude this feature and see the results. Note that the model excluding this feature (model 3) may not result in the optimal solution. Table 15 shows the comparison between the current system and model 3. It is seen that we do not have any ratios greater than 1.0 indicating a weakening.

Table 15- Comparison of the current model and the MIP Model without mixed block type constraints

Test #	Memory (mb.)			Performance (sec.)		
	Current model	Model 3	Ratio (m3 / c.m.)	Current model	Model 3	Ratio (m3 / c.m.)
1	321	190	0.5919003	111,5	1,6	0.0143498
2	275	208	0.7563636	25,2	2,8	0.1111111
3	359	255	0.7103064	311,55	12	0.0385171
4	282	214	0.7588652	26,3	2,8	0.1064639
5	250	211	0.844	15,2	1,8	0.1184211
6	250	211	0.844	-	-	-
7	491,63	322	0.6549641	109,7	22,2	0.2023701
8	-	-	-	19,1	16,3	0.8534031
9	-	-	-	16,75	13,6	0.8119403
10	527	236	0.4478178	138,7	18,2	0.1312185
11	-	-	-	-	-	-
12	500	236	0.472	118,5	14,3	0.1206751
13	550	501	0.9109091	43,4	29	0.6682028
14	-	-	-	-	-	-
15	502,75	422	0.8393834	24,45	19	0.7770961
16	500	472	0.944	17	7,8	0.4588235
17	1048	506	0.4828244	145,55	49,2	0.3380282
18	528,38	471	0.8914039	28,3	9,2	0.3250883
19	580,63	493	0.8490777	59,1	12,4	0.2098139
20	-	-	-	17,9	10,3	0.575419
21	590	503	0.8525424	75,3	22,2	0.2948207
22	-	-	-	39,32	9,4	0.2390641
23	533	505	0.9474672	-	-	-
24	-	-	-	40,2	9,6	0.238806
25	3878	499	0.1286746	788,6	8,3	0.010525
26	527,75	472	0.8943629	29,2	7	0.239726
27	298	209	0.7013423	20,8	1,7	0.0817308

6. Conclusion

In the thesis, the depot planning (shunt planning) problem was studied. As mentioned in chapter 1, the problem contains the decisions to be made in shunt yards by railway companies. Since it is a huge problem and there are different situations to be analyzed and interpreted, the overall problem is divided into sub-problems and each of them is solved separately. The focus of the thesis is on the parking sub-problem. A new mathematical model was developed which is able to solve all the instances. The findings and the comparisons with the current system were presented in chapter 5.

6.1. Further Research

Further research to the model can be studying how to introduce free tracks to the model and making the model more general. Another direction for improvement can be modifying the mixed block type constraints since they seem to be the bottleneck of the model. They can be handled in some other way or not all the constraints but some crucial ones that have a higher probability to increase the cost can be analyzed and added. Moreover, integration of the matching problem and the parking problem or the parking problem and the scheduling problem can be studied.

6.2. Research Questions

What is the importance of the problem?

As mentioned in chapter 1, most of the railway companies have done the overall planning by hand for years. All the sub-problems mentioned throughout the thesis and explained in chapter 3 are related with each other. The solution of one problem has an impact on the next problem, because the output of a problem is the input to the other one. Although the operations in railways seem to be fixed, there are many factors leading to sudden changes in plans and may require replanning of the remaining steps. That’s why the shunt planners do not prefer having detailed long term plans for the future. The later the plan is created the possibility that it will be modified is minimized. On the other hand, the sub-problems of interest are difficult to solve. As a consequence, the usage of decision support systems is increasing and helping the shunt planners create more robust plans in a small amount of time.

How can the quality of a solution be measured?

The quality of a solution depends on several aspects. Since the model was developed in such a way that satisfies the needs of shunt planners, some of the factors are already included in the objective function. The primary goal is minimizing the number of unparked units which has a huge penalty cost and later the number of broken arrivals and departures which have the second biggest impact on the cost.

As already mentioned previously, in the railway operations there can be sudden changes and disturbances anytime. As a consequence of this, planners prefer robust plans so that these sudden changes can be handled with fewer modifications. Robustness is increased by parking the same arrival/departure leg trains at the same shunt track as much as possible to decrease the workload at platforms. The attachment/detachment operations can be done at the shunt yard any time between the trains' arrival and departure times. On the other hand, if these operations were to be done at the platforms we wouldn't have that much time window, because platforms are much busier than shunt tracks. Furthermore, parking the same train units at the same shunt track provides flexibility to the planners and increases the robustness of the plan. In case of any disturbances, the same train units can be used interchangeably. The movement costs in the problem help us obtain better solutions in the scheduling part, any undesired or costly movements that will increase the possibility of having simultaneous moves can be minimized. Finally, the resources used are another metric especially for large instances.

What kind of methods can help solve the problem?

The overall problem and the parking problem have been studied in the literature and getting more popular recently. Having mentioned in chapter 2, different papers attack the problem in different ways. Exact methods such as mathematical programming and some dynamic programming techniques have been applied and some meta-heuristics and CSP techniques are also giving promising results. We have used an exact mathematical model and were able to solve the instances in a reasonable amount of time.

References

[1] R. Freling, R. M. Lentink, L. G. Kroon, and D. Huisman. Shunting of passenger train units in a railway station. *Transportation Science*, 39(2):261-272, May 2005.

[2] Ramon M. Lentink. Algorithmic Decision Support for Shunt Planning. PhD thesis, Erasmus University Rotterdam, Rotterdam, February 2006.

[3] Jesper Hansen. Rolling Stock Optimization at *DSB*. *Jeppesen* internal system specification, October 2005.

[4] Peter Føns. Desicion Support for Depot Planning in the Railway Industry. MSc thesis, Technical University of Denmark, Lyngby, April 2006.

[5] Frederik S. Hillier and Gerald J. Lieberman. Introduction to Operations Research. McGraw-Hill, 7th edition, 2001.

[6] U. Blasum, M. R. Bussieck, W. Hochstättler, C. Moll, H.-H. Scheel, and T. Winter. Scheduling trams in the morning. *Mathematical Methods of Operations Research*, 49(1):137-148, March 1999.

[7] Thomas Winter and Uwe T. Zimmermann. Real-time dispatch of trams in storage yards. *Annals of Operations Research*, 96:287–315, 2000.

[8] A. Schrijver. Planning van opstelsporen (Planning of shunt tracks). Technical report, Centrum voor Wiskunde en Informatica, 2003.

[9] Alexander Schrijver, Ramon M. Lentink, and Leo G. Kroon. Shunting of Passenger Train Units: an Integrated Approach. Technical report, Erasmus University Rotterdam, Rotterdam, 2005.

[10] E. Abbink. Intelligent shunting: Dealing with constraints (satisfaction). In W. van Wezel, R. Jorna, and A. Meystel, editors, *Planning in Intelligent Systems: Aspects, Motivations, and Methods*, pages 391-414. John Wiley & Sons, 2006.

[11] R. Haijema, C. Duin, and N. van Dijk. Train shunting: A pratical heuristic inspired by dynamic programming. In W. van Wezel, R. Jorna, and A. Meystel, editors, *Planning in Intelligent Systems: Aspects, Motivations, and Methods*, pages 437-475. John Wiley & Sons, 2006.

[12] D. Jekkers. Train shunt planning using genetic algorithms. Master's thesis, Erasmus School of Economics, Erasmus University Rotterdam, April 2009.

[13] Akker van den, Marjan and Baarsma, Hilbrandt and Hurink, Johann and Modelski, Maciej and Paulus, Jacob Jan and Reijnen, Ingrid and Roozemond, Dan and Schreuder, Jan (2008) Shunting passenger trains: getting ready for departure. In: 63rd European Study Group Mathematics with Industry, 28 Jan -1 Feb 2008, Enschede, The Netherlands.

[14] S. Cornelsen and G. Di Stefano. Track assignment. *Journal of Discrete Algorithms*, 5(2):250–261, 2007.

[15] *Jeppesen Systems AB*. 2011 [retrieved 2011-08]. Accessible at: http://www.jeppesen.com.

[16]*DSB*. 2011 [retrieved 2011-08]. Accessible at: http://www.dsb.dk/Global/PDF/Skole%20og%20studiemateriale/Virksomheden%20DSB%20S-tog.pdf

[17] V. J. Rayward-Smith, I. H. Osman, C. R. Reeves and G. D. Smith. Modern Heuristic Search Methods, 1996.

[18] G. A. Croes. A method for solving traveling salesman problems. *Operations Research* 6, pages 791-812, 1958.

[19] Francesca Rossi, Peter Van Beek, Toby Walsh. Handbook of constraint programming. Elsevier Science, 2006.

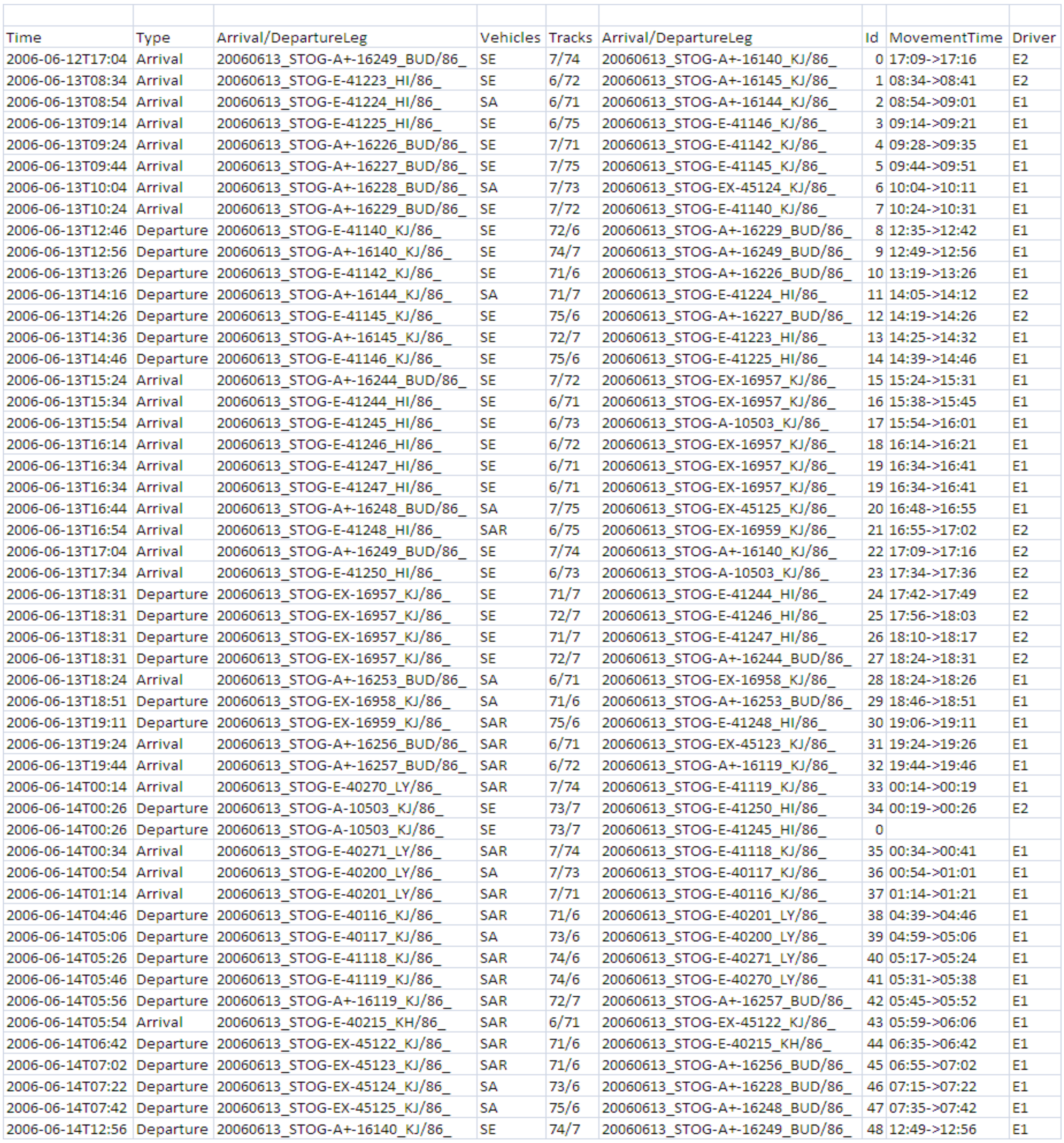
Appendix A - Terminology

Rolling stock	All the vehicles moving on a railway. In some regions locomotives are excluded.
Train unit	A single train vehicle. A train is composed of a set of train units.
Block	A set of connected train units always kept together throughout the problem.
Timetable	A plan showing the relevant information for blocks, i.e., arrival/departure times, arrival/departure platforms, the legs, configurations and lengths of blocks.
Crossing	The situation where two blocks cannot be parked at the same track, because one of them obstructs the departure of other block.
Shunt track	A track in a station which is connected to at least one platform and used for parking of blocks when they are not operating.
LIFO track	A track open on a single side. The blocks arriving/departing from a LIFO track can only use that open side, thus LIFO principle works.
Free track	A track that is open on both sides. The blocks arriving/departing from a free track can use both sides.
Platform	A track where passengers get on/off the trains.
Shunt yard	The whole area where the shunt tracks are located.
Shunting	The operation of moving train units from one track to another.
Depot	The whole station which includes a shunt yard.
Topology	The layout of the station showing all the connections between tracks and lengths of the tracks.
Track assignment	The assignment of a block to a track for parking during its stay at the depot.
Leg	A unique portion of a trip that is defined by two end points. Each block has an arrival leg and a departure leg, and a leg

	may consist of more than one block.
Front position of a block	The position of a block in the arrival leg. Usually the value is 1, but can change if there is more than one block in the leg.
To position of a block	The position of a block in the departure leg. Usually the value is 1, but can change if there is more than one block in the leg.
Attachment	Connecting a block to another block so that these blocks will move together as connected units.
Detachment	Disconnecting attached blocks so they can move separately.
Broken arrival	The term used for describing the situation when two blocks arriving together are detached at the platform and cannot be parked together on the tracks.
Broken departure	The term used for describing the situation when two blocks departing together are retrieved from different tracks or are parked in wrong order at the same track, thus have to be attached at the platform.
Platform parking	Parking a block at a platform instead of a shunt track.
Original blocks	A term specific to the thesis. For each block which is allowed to park at a platform via depot, there are two data entries: original block and its variant. If the block is normally parked at a shunt track its original version is used in the model.
Variant blocks	For each block which is allowed to park at a platform via depot, choosing the variant and parking it means that the block will be platform parked via depot. First a shunt track will be assigned for temporary parking, and then the block will move to the platform for the rest of its stay. Explained in detail in chapter 4.
Normal blocks	The set of blocks which are not allowed to park at platforms or the ones which can be parked at platforms directly – without moving to depot first-.
Driver	A person driving the trains in the shunt yard.
Turn at platform	The term describing the situation when

	the driving direction of a train is changed at the platform.
Turn on connection	The term describing the situation when the driving direction of a train is changed on the way from platform to the shunt track or from shunt track to the platform.

Appendix B – A Sample Test Result



Time	Type	Arrival/DepartureLeg	Vehicles	Tracks	Arrival/DepartureLeg	Id	MovementTime	Driver
2006-06-12T17:04	Arrival	20060613_STOG-A+-16249_BUD/86_	SE	7/74	20060613_STOG-A+-16140_KJ/86_	0	17:09->17:16	E2
2006-06-13T08:34	Arrival	20060613_STOG-E-41223_HI/86_	SE	6/72	20060613_STOG-A+-16145_KJ/86_	1	08:34->08:41	E2
2006-06-13T08:54	Arrival	20060613_STOG-E-41224_HI/86_	SA	6/71	20060613_STOG-A+-16144_KJ/86_	2	08:54->09:01	E1
2006-06-13T09:14	Arrival	20060613_STOG-E-41225_HI/86_	SE	6/75	20060613_STOG-E-41146_KJ/86_	3	09:14->09:21	E1
2006-06-13T09:24	Arrival	20060613_STOG-A+-16226_BUD/86_	SE	7/71	20060613_STOG-E-41142_KJ/86_	4	09:28->09:35	E1
2006-06-13T09:44	Arrival	20060613_STOG-A+-16227_BUD/86_	SE	7/75	20060613_STOG-E-41145_KJ/86_	5	09:44->09:51	E1
2006-06-13T10:04	Arrival	20060613_STOG-A+-16228_BUD/86_	SA	7/73	20060613_STOG-EX-45124_KJ/86_	6	10:04->10:11	E1
2006-06-13T10:24	Arrival	20060613_STOG-A+-16229_BUD/86_	SE	7/72	20060613_STOG-E-41140_KJ/86_	7	10:24->10:31	E1
2006-06-13T12:46	Departure	20060613_STOG-E-41140_KJ/86_	SE	72/6	20060613_STOG-A+-16229_BUD/86_	8	12:35->12:42	E1
2006-06-13T12:56	Departure	20060613_STOG-A+-16140_KJ/86_	SE	74/7	20060613_STOG-A+-16249_BUD/86_	9	12:49->12:56	E1
2006-06-13T13:26	Departure	20060613_STOG-E-41142_KJ/86_	SE	71/6	20060613_STOG-A+-16226_BUD/86_	10	13:19->13:26	E1
2006-06-13T14:16	Departure	20060613_STOG-A+-16144_KJ/86_	SA	71/7	20060613_STOG-E-41224_HI/86_	11	14:05->14:12	E2
2006-06-13T14:26	Departure	20060613_STOG-E-41145_KJ/86_	SE	75/6	20060613_STOG-A+-16227_BUD/86_	12	14:19->14:26	E2
2006-06-13T14:36	Departure	20060613_STOG-A+-16145_KJ/86_	SE	72/7	20060613_STOG-E-41223_HI/86_	13	14:25->14:32	E1
2006-06-13T14:46	Departure	20060613_STOG-E-41146_KJ/86_	SE	75/6	20060613_STOG-E-41225_HI/86_	14	14:39->14:46	E1
2006-06-13T15:24	Arrival	20060613_STOG-A+-16244_BUD/86_	SE	7/72	20060613_STOG-EX-16957_KJ/86_	15	15:24->15:31	E1
2006-06-13T15:34	Arrival	20060613_STOG-E-41244_HI/86_	SE	6/71	20060613_STOG-EX-16957_KJ/86_	16	15:38->15:45	E1
2006-06-13T15:54	Arrival	20060613_STOG-E-41245_HI/86_	SE	6/73	20060613_STOG-A-10503_KJ/86_	17	15:54->16:01	E1
2006-06-13T16:14	Arrival	20060613_STOG-E-41246_HI/86_	SE	6/72	20060613_STOG-EX-16957_KJ/86_	18	16:14->16:21	E1
2006-06-13T16:34	Arrival	20060613_STOG-E-41247_HI/86_	SE	6/71	20060613_STOG-EX-16957_KJ/86_	19	16:34->16:41	E1
2006-06-13T16:34	Arrival	20060613_STOG-E-41247_HI/86_	SE	6/71	20060613_STOG-EX-16957_KJ/86_	19	16:34->16:41	E1
2006-06-13T16:44	Arrival	20060613_STOG-A+-16248_BUD/86_	SA	7/75	20060613_STOG-EX-45125_KJ/86_	20	16:48->16:55	E1
2006-06-13T16:54	Arrival	20060613_STOG-E-41248_HI/86_	SAR	6/75	20060613_STOG-EX-16959_KJ/86_	21	16:55->17:02	E2
2006-06-13T17:04	Arrival	20060613_STOG-A+-16249_BUD/86_	SE	7/74	20060613_STOG-A+-16140_KJ/86_	22	17:09->17:16	E2
2006-06-13T17:34	Arrival	20060613_STOG-E-41250_HI/86_	SE	6/73	20060613_STOG-A-10503_KJ/86_	23	17:34->17:36	E2
2006-06-13T18:31	Departure	20060613_STOG-EX-16957_KJ/86_	SE	71/7	20060613_STOG-E-41244_HI/86_	24	17:42->17:49	E2
2006-06-13T18:31	Departure	20060613_STOG-EX-16957_KJ/86_	SE	72/7	20060613_STOG-E-41246_HI/86_	25	17:56->18:03	E2
2006-06-13T18:31	Departure	20060613_STOG-EX-16957_KJ/86_	SE	71/7	20060613_STOG-E-41247_HI/86_	26	18:10->18:17	E2
2006-06-13T18:31	Departure	20060613_STOG-EX-16957_KJ/86_	SE	72/7	20060613_STOG-A+-16244_BUD/86_	27	18:24->18:31	E2
2006-06-13T18:24	Arrival	20060613_STOG-A+-16253_BUD/86_	SA	6/71	20060613_STOG-EX-16958_KJ/86_	28	18:24->18:26	E1
2006-06-13T18:51	Departure	20060613_STOG-EX-16958_KJ/86_	SA	71/6	20060613_STOG-A+-16253_BUD/86_	29	18:46->18:51	E1
2006-06-13T19:11	Departure	20060613_STOG-EX-16959_KJ/86_	SAR	75/6	20060613_STOG-E-41248_HI/86_	30	19:06->19:11	E1
2006-06-13T19:24	Arrival	20060613_STOG-A+-16256_BUD/86_	SAR	6/71	20060613_STOG-EX-45123_KJ/86_	31	19:24->19:26	E1
2006-06-13T19:44	Arrival	20060613_STOG-A+-16257_BUD/86_	SAR	6/72	20060613_STOG-A+-16119_KJ/86_	32	19:44->19:46	E1
2006-06-14T00:14	Arrival	20060613_STOG-E-40270_LY/86_	SAR	7/74	20060613_STOG-E-41119_KJ/86_	33	00:14->00:19	E1
2006-06-14T00:26	Departure	20060613_STOG-A-10503_KJ/86_	SE	73/7	20060613_STOG-E-41250_HI/86_	34	00:19->00:26	E2
2006-06-14T00:26	Departure	20060613_STOG-A-10503_KJ/86_	SE	73/7	20060613_STOG-E-41245_HI/86_	0		
2006-06-14T00:34	Arrival	20060613_STOG-E-40271_LY/86_	SAR	7/74	20060613_STOG-E-41118_KJ/86_	35	00:34->00:41	E1
2006-06-14T00:54	Arrival	20060613_STOG-E-40200_LY/86_	SA	7/73	20060613_STOG-E-40117_KJ/86_	36	00:54->01:01	E1
2006-06-14T01:14	Arrival	20060613_STOG-E-40201_LY/86_	SAR	7/71	20060613_STOG-E-40116_KJ/86_	37	01:14->01:21	E1
2006-06-14T04:46	Departure	20060613_STOG-E-40116_KJ/86_	SAR	71/6	20060613_STOG-E-40201_LY/86_	38	04:39->04:46	E1
2006-06-14T05:06	Departure	20060613_STOG-E-40117_KJ/86_	SA	73/6	20060613_STOG-E-40200_LY/86_	39	04:59->05:06	E1
2006-06-14T05:26	Departure	20060613_STOG-E-41118_KJ/86_	SAR	74/6	20060613_STOG-E-40271_LY/86_	40	05:17->05:24	E1
2006-06-14T05:46	Departure	20060613_STOG-E-41119_KJ/86_	SAR	74/6	20060613_STOG-E-40270_LY/86_	41	05:31->05:38	E1
2006-06-14T05:56	Departure	20060613_STOG-A+-16119_KJ/86_	SAR	72/7	20060613_STOG-A+-16257_BUD/86_	42	05:45->05:52	E1
2006-06-14T05:54	Arrival	20060613_STOG-E-40215_KH/86_	SAR	6/71	20060613_STOG-EX-45122_KJ/86_	43	05:59->06:06	E1
2006-06-14T06:42	Departure	20060613_STOG-EX-45122_KJ/86_	SAR	71/6	20060613_STOG-E-40215_KH/86_	44	06:35->06:42	E1
2006-06-14T07:02	Departure	20060613_STOG-EX-45123_KJ/86_	SAR	71/6	20060613_STOG-A+-16256_BUD/86_	45	06:55->07:02	E1
2006-06-14T07:22	Departure	20060613_STOG-EX-45124_KJ/86_	SA	73/6	20060613_STOG-A+-16228_BUD/86_	46	07:15->07:22	E1
2006-06-14T07:42	Departure	20060613_STOG-EX-45125_KJ/86_	SA	75/6	20060613_STOG-A+-16248_BUD/86_	47	07:35->07:42	E1
2006-06-14T12:56	Departure	20060613_STOG-A+-16140_KJ/86_	SE	74/7	20060613_STOG-A+-16249_BUD/86_	48	12:49->12:56	E1

Figure 11- A sample solution for Køge Depot