Branch-and-Cut-and-Price to solve a relaxation of the Train Unit Shunting and Service problem

Finding the Shortest Path with Waypoints with durations for train units on shunting yards

K.J.H. (Koen) Hendrikse, 502000

Master Thesis Econometrics & Management Science: Operations Research and Quantitative Logistics, Erasmus Universiteit Rotterdam

Company Supervisors

Ir. B. Huisman

Ir. J. Mulderij

Dr. D.D. Tönissen

University Supervisor

Dr. T.A.B. Dollevoet

Second assesor

Prof. dr. D. Huisman

Version of January 22, 2021

Abstract

Logistic Capacity determination of shunting yards of the Netherlands Railways is performed by constructing feasible shunting plans. It is essential to use a complete method for this, such that no potentially feasible solution is disregarded. By doing so, the capacity is not underestimated. This thesis proposes two complete MIP formulations to construct shunting plans for single train units. The main contribution is that it includes the Service sub-problem, by utilising waypoints with durations in a Multi-Agent Pathfinding framework. Furthermore, the parking, routing and (partial) matching sub-problem are taken into account when forming the paths. The second MIP formulation performs computationally better or equal on all tested scenarios and instances. It is advised to implement the gateway extension as proposed in the thesis, regarding the improvement of both computation times and tightness. The MIP formulation can be operated as a Pricing Problem in the Branch-and-Cut-and-Price framework. This framework is used to solve the Multi-Agent Pathfinding translation of the Train Unit Shunting and Service problem. Extensions such as the formulated additional gateway constraints and corridor conflicts could be added in order to speed up the method.

Contents

List of Notations

Sets

T	-	Set of available tracks at the yard.
T^{σ}	-	Set of available tracks for service σ at the yard.
Σ	-	Set of all services.
Σ^{a}	-	Set of services to be performed by train unit a.
L	-	Set of locations, the tracks at the yard.
E	-	Set of edges, the connections between the locations.
D	-	Set of gateway tracks.
T	-	Set of time steps.
T^{a}	-	Set of time steps relevant to train unit a.
V	-	Set of time-extended vertices.
$\hat{E}$	-	Set of time-extended edges.
S	-	Set of arrival nodes.
S^{a}	-	Set of arrival nodes relevant for agent a.
G	-	Set of departure nodes.
G^{a}	-	Set of departure nodes relevant for agent a.
$\delta_{in(l,t)}$	-	Set of ingoing edges of node (l,t).
$\delta_{out(l,t)}$	-	Set of outgoing edges of node (l,t).
$\delta_{self(l,t)}$	-	Set of self-recurring edges of node (l,t).
D^{s}	-	Set of arrival gateways.
D^{g}	-	Set of departure gateways.
$G = (L, E)$	-	Undirected graph.
$\hat{G} = (V, \hat{E})$	-	Time-expanded graph.
A	-	Set of agents, all train units.
P_a	-	Set of all possible paths for agent a.
P'_a	-	Set of some possible paths for agent a, smaller pool for Column Generation.
Δ	-	Set of possible departure times of all train units.
Ψ	-	Set of possible arrival times of all train units.

Variables

λ_p^a	-	Binary variable, indicating whether agent a chooses path p.
y_e	-	Binary variable, indicating whether edge e is used.
α_a	-	Dual variable for path selection constraints.
β_v	-	Dual variable for vertex constraints.
γ_e	-	Dual variable for edge constraints.
η_g	-	Dual variable for departure constraints.
$\pi_{a_1,a_2,l_1,l_2,t}$	-	Dual variable for corridor conflict constraints.

Parameters

l_τ	-	Length of track τ.
d_σ	-	Duration of service σ.
c_p^a	-	Cost of path p selected by agent a.
x_v^p	-	Binary, indicating whether vertex v is used by path p.
y_e^p	-	Binary, indicating whether edge e is used by path p.

Acronyms

NS	-	Nederlandse Spoorwegen, Netherlands Railways.
TUSS	-	Train Unit Shunting and Service problem.
TUSP	-	Train Unit Shunting Problem.
MAPF	-	Multi-Agent Pathfinding.
MAPFW	-	Multi-Agent Pathfinding with Waypoints with durations.
BCP	-	Branch-and-Cut-and-Price.
BCP-B	-	Branch-and-Cut-and-Price basic.
CBS	-	Conflict-Based Search.
TSP	-	Traveling Salesman problem.
LIFO	-	Last In First Out.
MIP1	-	First MIP formulation of the Pricing Problem.
MIP2	-	Second MIP formulation of the Pricing Problem.
ML	-	Machine Learning.

1 Introduction

1.1 Background of the Shunting Problem

At the end of the day, when the timetable operations of the trains of the Netherlands Railways (Nederlandse Spoorwegen, NS) have finished, the trains are parked at a shunting yard. Various tasks such as parking, cleaning, repairing, special washing, regular checks and maintenance should be performed at such a yard. Train unit shunting is the process of parking train units at a shunt yard and performing all the related processes (Kroon et al., 2008). A train is composed of several train units that could vary in exact type. Such a train unit is defined as an independently, bidirectionally moving unit. A feasible shunting plan should be generated, given the layout of the service site, the timetable and the list of service tasks. In van den Broek (2016) a feasible shunt plan is defined as a plan consisting of the following components: an assignment of arriving train units to departing ones, when and where service tasks should be performed, assignment of trains to tracks for parking and specified routes of the trains while avoiding conflicts between trains.
The shunting problem is considered as one of the most important rolling stock problems by Lusby et al. (2011) for various reasons. The process is a challenging problem as it is restricted from different angles. The shunt yards are often located in cities, thus the infrastructure on which all the movements of train units are performed is usually restricted in its size. An example of such a shunting yard in The Hague can be found in Figure 1. Furthermore, time is a limiting factor as all of the related tasks of a train unit should be performed before the concerned train unit is scheduled to depart. When the timetable of the next morning starts, the right type of train compositions should be ready at the right time, after each train unit having completed its own tasks.

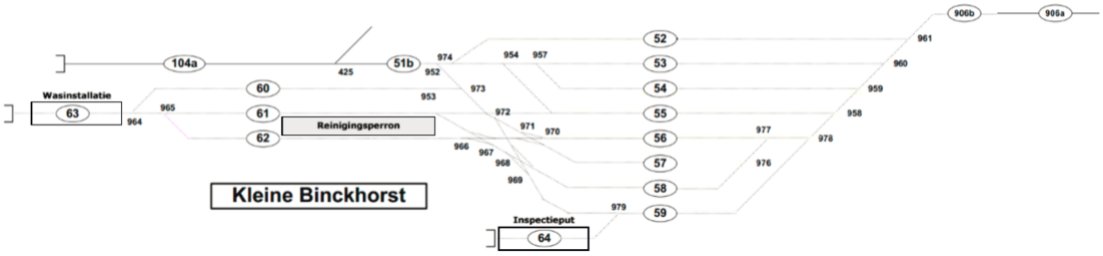

Figure 1: Shunting Yard "Kleine Binckhorst" (van den Broek, 2016)

Typically, the Train Unit Shunting and Service problem (TUSS) consists of the following sub-problems: matching of trains, service or cleaning of trains, parking the trains and routing the trains on the shunt yard. In the literature review, more detailed information about the sub-problems and solution approaches will be provided. Previously, the TUSS was often handled by sequentially solving the sub-problems (Mulderij et al., 2020). However, this could result in sub-optimal shunting plans and feasible plans that are disregarded.
The Multi-Agent Pathfinding problem (MAPF) has similarities with the routing sub-problem. It will be extended in this thesis such that it matches the entire TUSS problem more closely. It is

a relaxation of the TUSS problem. The MAPF is an NP-hard graph optimization problem that involves finding a minimum-cost set of paths for different agents. In this problem each agent should move from its start position to the goal such that no two agents cross paths. In order to include service tasks per train unit, waypoints are added to the MAPF formulation. These waypoints are given durations, such that longer service tasks can be performed.
The Branch-and-Cut-and-Price (BCP) framework will be used to solve the MAPF with Waypoints with durations problem (MAPFW). Due to the inclusion of waypoints, the complexity of the Pricing Problem is increased. In this framework, the Master Problem selects the added paths for the single agents, the Pricing Module adds feasible paths to the Master Problem and the Separator Module adds constraints to the Master Problem such that conflicting paths are prevented. All the components of the BCP framework interact with each other, therefore all will be explained. However, the main purpose of this thesis is to develop the Pricing Module within this framework.

1.2 Aim and Relevance of the thesis

The relevance of this thesis is that it enables NS to provide better bounds for the logistic capacity of their shunt yards. This capacity is described by van de Ven et al. (2019) as the total number of trains that can be served by a shunting yard in a 24-hour period. Bounds on the capacity are determined by testing several benchmark instances on a shunting yard and then calculating the percentage of instances for which feasible shunt plans could be generated. Therefore, the algorithm should be complete. This means that if there is a feasible solution; the algorithm should be able to find it. An exact approach where all constraints are included takes too much computation time. As no feasible solutions should be disregarded, the aim is to provide a relaxation of the real life problem.
Previous approaches, such as van den Broek (2016), did not provide a complete heuristic. As feasible shunt plans could be neglected by these approaches, it could not be concluded with certainty if it was not possible to generate feasible plans for a specific instance. This thesis aims to provide a complete solution approach to form single train unit paths, which results in a tighter formulation of the relaxation. This implies that the quality of the logistic capacity bound will be improved by utilising the proposed approach in the Branch-and-Cut-and-Price framework. The method currently used at NS excludes the routing sub-problem. Thus it is expected that the capacity determination could be improved by including this aspect via the MAPF formulation according to Mulderij et al. (2020).
Based on the logistic capacity bounds, investment decisions on a strategic level could be made by NS on acquiring new infrastructure. Furthermore, if the computational results are promising, the model could be extended in the future in order to use the generated shunt plans on a daily operational level. In the existing literature no articles are found that include waypoints (with durations) to the MAPF and use the BCP framework. As the inclusion of waypoints makes the Pricing Problem more complex than in reviewed literature, this thesis mainly contributes to

research in this area. Another contribution is made to the Netherlands Railways, where the process of logistic capacity determination is performed with more certainty due to the complete solution method. As a result, this research is also of practical relevance.

1.3 Research Questions

The purpose of this thesis is to develop a Pricing Problem within the Branch-and-Cut-and-Price framework, while ensuring completeness of the approach. As the context of the problem is introduced, the main research question is formulated as follows:

Q: How can a Pricing Problem within the Branch-and-Cut-and-Price framework be developed, such that completeness of the overall approach to solve the Train Unit Shunting and Service problem formulation is ensured?

In answering this question, the main question is divided into five sub-questions in order to keep the research orderly. Firstly, the Multi-Agent Pathfinding with Waypoints with durations formulation of the Train Unit Shunting and Service problem relaxation is discussed. Then the link between the solution approach, Branch-and-Cut-and-Price and the overlapping MAPFW formulation is investigated. The third sub-question focuses on the reduced costs, thus the objective function of the Pricing Problem, as it interacts with the other modules in the BCP framework. Furthermore, the real life service tasks and constraints for the Pricing Problem are examined. Lastly, the computational results and the quality of the approach are studied. Summarizing, this leads to the following sub-questions:

Q_1: How can the MAPF with Waypoints with durations formulation be used to model the relaxed TUSS problem such that completeness is ensured?

Q_2: How can the Branch-and-Cut-and-Price Framework be used to solve the MAPF with Waypoints with durations formulation of the TUSS problem?

Q_3: How can the objective function of the Pricing Problem(s) be constructed and be updated?

Q_4: Which requirements for feasible paths are there in practice and how can they be modelled in the Pricing Problem(s) such that completeness is ensured?

Q_5: What are the results concerning computational times and performance measures of the Pricing Problem(s)?

1.4 Outline of the thesis

Relevant literature on the Shunting Problem, the Multi-Agent Pathfinding problem, Branch-and-Cut-and-Price Framework and similar Pricing Problems are discussed in Section 2 of this thesis.

In this section, hypotheses are formed for all the sub-questions. Following, the assumptions and a detailed description of the problem are provided. In Section 4 the first sub-question is treated, as the MAPF with Waypoints with durations formulation is studied. This is followed by a section concerning the BCP framework, which covers the second sub-question. Next, the Pricing Problem is investigated in detail, thus Section 6 will answer the third and fourth sub-question. Therefore, Sections 4, 5 and 6 encompass the methodology of this thesis. The last sub-question is answered in Section 7, where the computational results and quality are compared. Finally, in Section 8 the main research question is answered and indications for future research are provided.

2 Literature Study

In the relevant literature, a distinction is made between the Train Unit Shunting Problem (TUSP), in which service tasks are not taken into consideration, and the Train Unit Shunting and Service problem (TUSS), in which they are. The TUSP solution approaches are examined first, followed by the TUSS solution approaches. As the TUSS problem is modelled using a Multi-Agent Pathfinding with Waypoints with durations formulation, relevant literature on the MAPF is discussed afterwards. This is followed by a review of papers that concern the Branch-and-Cut-and-Price Framework. Lastly, literature about problems similar to our Pricing Problem is elaborated on.

2.1 TUSP

In the Train Unit Shunting Problem (TUSP), the infrastructure of both the railway station and the shunt yard are given. In addition, the timetable providing information on the arrival and departure times along with the corresponding tracks is given as well. The TUSP, as it is defined in Huisman et al. (2005), consists of matching the arriving and departing train units and parking these at shunt tracks such that no crossings occur. Here, the shunting costs should be minimized. In the relevant literature, a clear distinction is made between LIFO- and free tracks. LIFO tracks are shunt tracks that are open at either the left or the right side. In contrast, free tracks can be accessed from both sides. A well-known strongly NP-hard problem is the bin-packing problem. It is derived by Haijema et al. (2006) that the shunting problem is more difficult than the bin-packing problem. Therefore the shunting problem is also strongly NP-hard.
The train units are not only restricted in their movements by the railway infrastructure but also by safety regulations (Freling et al., 2005). Train units moving on the same tracks and switches should maintain a specified minimum headway time in between their movements. This paper approaches the problem by first solving the matching sub-problem and afterwards assigning the train units to shunt tracks. The latter is called the parking sub-problem, which is solved by column generation. Furthermore, a solution approach, suggested by Lentink et al. (2006), is to decompose the TUSP and solve it in the following sequential steps: matching of the arriving and departing train units, estimating the routing costs of the train units, parking the train units on the shunt tracks and routing the train units. However, such a decomposition followed by a sequential approach can result in sub-optimal solutions and is not complete.
Another solution approach, by Kroon et al. (2008), solves the matching and parking problem simultaneously. This increases the complexity of the problem as a lot of crossing constraints are added. In van den Akker et al. (2008) an integrated method to solve the TUSP is used as well. A greedy heuristic and a dynamic programming approach are derived, in which trains are allowed to wait at platforms. However, the TUSP formulation does not yet include service tasks for the train units.

2.2 TUSS

In the Train Unit Shunting and Service problem (TUSS), the service tasks are included in the TUSP. Thus it should be decided when the service is provided to the train units. As multiple locations could be feasible to perform the service, the exact location where each service task is performed has to be decided upon as well. Another important factor of the TUSS is the routes that are driven by the train units, which obviously should be unobstructed. According to Geiger et al. (2018) the formulation of TUSS that was based closely on a real life situation could only be solved by heuristics due to the complexity and size. The algorithmic complexity of the problem emerges mainly from the interaction of the sub-problems (van den Broek, 2016). In the latter mentioned paper a local search method, using simulated annealing, is used to come up with feasible shunt plans for the TUSS. However, the quality of this heuristic cannot be estimated. The set of service tasks to be completed are specified on train unit level, although in practice the service is performed on the whole train composition. Thus, according to van den Broek (2016) the duration of the service depends on the composition of the train, as all individual train unit service task durations are added up.

In Lentink (2006) only cleaning tasks are considered in the set of service tasks. Thus the problem as described here could be seen as a special case of the TUSS. The advantages of integrating the matching and parking sub-problems are investigated, which show benefits compared to the sequential procedure. This is followed by solving the routing and cleaning crew scheduling sub-problems.

The problem described in Jacobsen and Pisinger (2011) cannot be truly classified as a TUSS problem. This is because not all sub-problems are included. The parking sub-problem is combined with workshop scheduling, in which maintenance of trains is planned while ensuring unobstructed trains and no delays. Two local searches and one simulated annealing approach are used to solve the problem.

The study of van de Ven et al. (2019) combines Machine Learning with Operations Research as it utilises local search and a Neural Network to obtain feasible shunting plans. The purpose is to speed up the search for feasible shunting plans, in order to then determine the logistic capacity of the shunt yards for multiple instances. This closely matches the purpose for which the solution methods in this thesis are proposed. A significant reduction in computation time is achieved by the method. However, this approach is not complete.

2.3 MAPF

The already mentioned approach of van den Broek (2016) encompasses the TUSS problem in all facets, but the heuristic cannot be considered to be complete. As the completeness of the solution method is an important factor in order to qualify instances as infeasible, Mulderij et al. (2020) came up with the idea to use a Multi-Agent Pathfinding formulation to model a relaxation of the

TUSS problem. This formulation will be used as the foundation for this thesis.

Prior to reviewing the link between MAPF and TUSS, the MAPF is explained in detail by discussing the work of Stern et al. (2019). In the MAPF problem, agents should be assigned to paths which they could traverse without colliding with each other. The input for the classical problem is an undirected graph with source and target vertices for all agents. During each discretized time step an agent could either wait at their current position or move to an adjacent position. A sequence of actions over the time span of the problem is defined as a single-agent plan. Thus a solution to the MAPF consists of single-agent plans for all individual agents, where the objective function usually minimizes the makespan or the total sum of costs. Various conflicts could occur, which should be prevented. Commonly vertex and edge conflicts are not allowed. In these conflicts a vertex is either occupied or an edge is traversed by multiple agents at the same time.

The classical MAPF problem is described above. Now, the relevant extensions for this thesis, as depicted in Stern et al. (2019) will be discussed. Where all actions are assumed to take one time step in the classical MAPF problem, this is extended by MAPF on weighted graphs where actions vary in duration. This is relevant, as van den Broek (2016) describes how different train movements, e.g. traversing, switches and saw moves, have different durations. Another extension focuses on large agents, as these could forbid other agents to visit nearby vertices at the same time due to their size. This has an obvious link with trains which could vary in size and therefore occupy a larger part of a track. The last extension, relevant to this thesis, focuses on assigning more than one task to agents. As a result, not only the end location is specified as a goal. An example of this is the so-called Colored MAPF, where agents are grouped into teams and every team has a set of targets.

In order to incorporate the different starting and departing times of train units, a space-time graph could be used in the MAPF as described in Mulderij et al. (2020). This graph allows for a more precise modeling of the shunting yards over time. In the MAPF problem, train units could be translated as agents. If service tasks at certain tracks occur, then non-colliding paths should be generated that lead to these tracks. In the basic MAPF formulation of Mulderij et al. (2020) not all sub-problems of the TUSS are taken into account, thus it is a relaxation of the problem. Several extensions to the basic MAPF relaxation of the TUSS problem are discussed such as adding the matching sub-problem, including restrictions on direction of movement of agents and including service tracks. These could all be used to represent the TUSS more accurately and thus improve the logistic capacity bound. In this thesis, the service tasks are incorporated into the MAPF model by waypoints.

The fact that the work of Ma and Koenig (2016) considers the allocation of tasks to the agents reinforces the importance of this work. First team-specific tasks are distributed over the agents, then feasible paths are generated. However, the agents are part of teams and each agent is assigned to one tasks, which is not the case in this thesis.

As the literature review concerning the TUSP, TUSS and MAPF is discussed, a hypothesis of the

first sub-question can be formed:

H1: The MAPF formulation can be used by considering train units as agents. By ensuring that the formulation is a relaxation of the TUSS, completeness can be guaranteed.

2.4 BCP Framework

Methods to solve the relevant scale of the TUSS problem are not available yet. Nonetheless, Branch-and-Cut-and-Price is proposed as a complete solution method to solve the MAPF relaxation by Mulderij et al. (2020). Their paper suggests to apply the framework as described in Lam et al. (2019) as this framework is able to solve MAPF problems for up to 100 agents. Another solver for MAPF problems is the Conflict-Based Search (CBS) method of Sharon et al. (2015). This method creates paths for single agents separately. Each time a conflict occurs between two agents, two branches are created in which one agent is given priority over the other. In Branch-and-Cut-and-Price, the decomposition and domain-specific reasoning strengths of CBS are combined with the search performance of MIP.

Branch-and-Cut-and-Price acts as a two-level algorithm. At the lower level single-agent pathfinding is performed. At the higher level, the multiple paths are assigned to the agents and consequently conflicts are resolved between them. The solution method identifies a Master Problem, which selects paths from the set P'. This set is filled by the Pricing Module, which adds paths that are promising to the set. The Separator Module is applied repeatedly, stopping at the moment when there do no longer occur any conflicts in the fractionally chosen paths. If any path is fractionally chosen, a branching rule creates two child nodes that do not contain the same fractional chosen path.

MIP problems are commonly solved with the enumeration technique called Branch-and-Bound, which is also utilised in the Branch-and-Cut-and-Price framework. In this method all solutions are enumerated in a search tree by solving a relaxation at each node. Often only a small fraction of the solution actually needs to be enumerated due to the obtained bounds (Mitten, 1970). The processes of branching and bounding are applied recursively. Branching consists of dividing sets of solutions into subsets. The process of bounding obtains bounds on the objective function values of these subsets.

When the Pricing Problem in Lam et al. (2019) declares that the problem is fractionally optimal, by not finding improving variables, then branching into two disjoint sets is performed in order to resolve the fractionality. In the BCP-B, the basic framework proposed in the paper, branching is performed on agent-vertex pairs. In one child node the vertex is visited by the agent, while in the other it is not.

Previous literature of Peeters and Kroon (2008), related to the Netherlands Railways, proposes a Branch-and-Price model to determine an efficient circulation of rolling stock on a number of interrelated train lines. Thus, their problem is initialized with a small set of variables and promising variables are added by the Pricing Module, which is an algorithm that generates potentially

better variables. Branch-and-Bound is thus extended by solving the LP-relaxations with a column generation approach.
It is important to note that Lam et al. (2019) use a grid based structure, where locations have neighbours in the following four directions: north, east, south and west. This could be less applicable to the infrastructure of a shunt yard. The study proposes two frameworks: BCP-B and BCP. The first consists of the Branch-and-Cut-and-Price framework in which edge and vertex conflicts are resolved. These are necessary and sufficient for modelling the MAPF formulation. BCP extends this first approach by tightening the LP-relaxation of BCP-B as it uses additional corridor and rectangle symmetry constraints. Furthermore, in the extension a second branching rule is suggested. Even more inequalities are proposed in Lam and Le Bodic (2020), which extends the separator module of the BCP framework by suggesting additional tightening constraints.
The second hypothesis can be formed as the Branch-and-Cut-and-Price framework is introduced:

H2: The BCP framework can be applied by generating feasible train unit paths in the Pricing Module, selecting these paths for all train units in the Master Problem and adding constraints in the Separator Module whenever conflicts occur.

2.5 Pricing Problem

The purpose of the Pricing Problem in Lam et al. (2019) is to find better paths, which are then added to the set of paths in the Master Problem. If this is not possible, the Pricing Problem proves that no improving paths exist which means that the current solution is optimal. As the Master Problem selects the paths for all agents, the Pricing Problem is a Shortest Path problem for each individual agent where the objective function is adapted.
However, the MAPF formulation in Lam et al. (2019) does not include waypoints. As the formulation in this thesis does include waypoints with durations, the complexity of the Pricing Problem is increased and it is not a Shortest Path problem anymore. A solution to this problem moves from the start location of the agent to the end location, where on its path it visits all locations that are specified in the agent specific set of waypoints. The order of the waypoints is not prefixed. This problem resembles the Traveling Salesman problem in some way, although the start and end locations are not identical. This version of the TSP, where the start and end points differ, is called a Hamiltonian Path problem. All vertices should be visited in the Hamiltonian Path problem described by Gurevich and Shelah (1987). In contrast, only vertices in the set of waypoints have to be visited in this thesis.
The Traveling Salesman Problem (TSP) as described by Matai et al. (2010) is defined as finding the shortest way that visits all cities and then returns to the starting point. The classical TSP is symmetric in distances. The problem is only classified as asymmetric when at least one distance from i to j differs from the distance from j to i. In Pataki (2003) two different TSP formulations are provided and compared. The Miller-Tucker-Zemlin formulation is small in size and routing preferences can be incorporated in an easy manner (Miller et al., 1960). However, the Subtour

formulation is tighter (Lawler, 1985). The exponential amount of constraints can be added by a separator module and do not need to be present from the start. The inclusion of waypoints makes this Pricing Problem at least as difficult as the Traveling Salesman Problem (Gomes et al., 2017). In the last mentioned paper, a formulation is proposed to find a disjoint path pair that visits a set of specified nodes.

The study of Amiri et al. (2020) endeavours to find a shortest walk from a starting location to an end location, while visiting all vertices in a predetermined set of waypoints. This waypoint Routing Problem is modelled on a capacitated graph, thus the edge capacties should be respected. In de Andrade (2013) only paths without loops that visit the locations of the waypoints exactly once are considered. Two new formulations of the problem were proposed. The last one, a primal-dual based formulation, was proven to be the easiest to solve based on computational experiments. As the literature review has covered all relevant aspects of the sub-questions, the last three hypotheses can be formed:

H3: The objective function of the Pricing Problem resembles that of a Shortest Path problem. It is expected that it can be updated per constraint set, by updating the costs of using an edge or a vertex.

H4: The service tasks can be modelled using waypoints with durations without a prefixed order. Completeness is ensured by considering relaxations of real life feasible train unit paths.

H5: It is expected that the final proposed method performs computationally worse than previous approaches as: it is not a heuristic, it is not sequential, it is complete and it considers service tasks as well. This is expected to improve the logistic capacity determination. Thus, the other performance measure, infeasible percentage of instances, will improve.

3 Problem Description

Train units that are not in operation are parked at shunting yards; this happens mostly at night. Before the trains are able to transport passengers on the next day, it is ensured on the shunting yard that they are prepared to start the day correctly. This is done by performing various activities including cleaning, repairing, special washing, regular checks and maintenance. NS is interested in determining the logistic capacity of these yards. The definition of the logistic capacity of a shunting yard is the total number of trains that can be served by that yard in a 24-hour period (van de Ven et al., 2019).

The current approach to determine this capacity is to simulate different instances for all the existing shunting yards in the Netherlands. For each instance it is determined whether it is feasible to produce a shunting plan on the corresponding yard. As the instances contain various numbers of trains, insight is gained in how much trains can be handled by each shunting yard; the logistic capacity.

The purpose of this thesis is to come up with a method that is able to determine whether an instance is infeasible for a shunting yard. In order to perform this task, the Train Unit Shunting and Service model is utilised. This formulation describes which constraints should be satisfied in order to produce a feasible shunting plan in real life. The formulation, based on the work of Lentink et al. (2006) and van den Broek (2016) is elaborated on in the problem description.

In this section the preliminaries of the problem are discussed first. Next, the mathematical definitions of the four TUSS sub-problems are given. These are combined to describe the problem formulation of the TUSS. Throughout this section, the scope, relaxations and assumptions of this thesis are discussed. Furthermore, the problem is clarified by providing a small example. Lastly, the measurement of quality is defined such that it is clear how the results could be interpreted.

3.1 Preliminaries

3.1.1 Shunting Yard Infrastructure

The most important aspect of the shunting yards are the train tracks which are situated on the yard. On these tracks, trains could be parked, make movements or obtain service. Considering one shunting yard, the set T denotes the available tracks at the specific yard. There are three important aspects of a track: length, service availability and accessibility. Its length is denoted as l_τ for each $\tau \in \mathrm{T}$, the same mathematical notation as in van den Broek (2016) is utilised. Per service task σ, it is specified on which tracks the specific service action could be performed, which can be found in set T^σ.

There are two types of tracks concerning accessibility: LIFO- and free tracks, depicted in Figure 2. LIFO tracks can only be accessed from one side (either side A or side B), thus the train that enters first should leave last because it cannot leave on the dead end side. Free tracks, on the other hand, could be assessed from both sides. Moreover, so-called gateway tracks connect the shunting

yard to the rest of the railway infrastructure in the Netherlands. Therefore, arriving and departing always occurs on the gateway tracks.

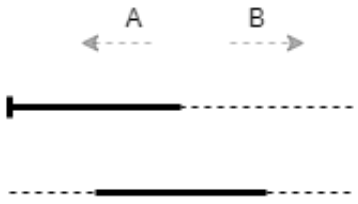

Figure 2: LIFO track accessible from B side (upper) and free track (lower)

3.1.2 Service Tasks

As denoted above, the set T^{σ} denotes on which tracks a service could be performed. Service tasks that are not track specific, could be performed on any track. Furthermore, a service σ in set Σ has a certain duration denoted by d_{σ}. In order to perform service σ, the train should stay for at least d_{σ} consecutive time units at the same track in T^{σ}.

3.1.3 Train Unit Compositions

Trains are often composed of train units. A train unit is defined as an independently, bidirectionally moving unit, which could vary in type. Only train units with matching types could be combined in a composition to form a larger train. The different compositions result in the fact that the length of trains could differ. However, in this thesis it is assumed that all trains consist of one and the same train unit type. Moreover, all trains are assumed to have a length of one train unit. Thus, the process of splitting and combining train units to form train compositions is not considered. This assumption is made in order to focus more on the routing and service sub-problems, which tend to be more promising considering the goal of this research (Mulderij et al., 2020). It is computationally not tractable to include all sub-problems, based on a real-life situation, and aim to solve real-life instances with an exact approach according to Geiger et al. (2018). Therefore, this relaxation in the field of the matching sub-problem is made.

3.1.4 Train Movements

In van den Broek (2016) is described how various train movements can have different durations. The durations are based on the number of tracks, switches and saw moves along the way. A saw move is made when a train should reverse its direction. These moves are quite time consuming as the driver should walk to the other end of the train when the direction is changed. The movements of the trains on the same tracks or switches are prohibited to occur directly after each other. A minimum headway time ensures the safety of the operations on the shunting yard.

3.1.5 Graph Representation

In van de Ven et al. (2019) and van den Broek (2016) an activity graph was used, thus nodes resemble activities as arrival, service, movement, parking and departure. In Mulderij et al. (2020) a shunting yard is represented by a graph. In this graph, each track is modelled as a number of nodes. The number of nodes for a track is determined by $\lceil \frac{l_\tau}{shortest\ train\ unit\ length} \rceil$, such that this is a relaxation of the reality. It is a relaxation as this form of modelling ensures enough space on the shunting yard. Due to the fact that only one train unit type is considered, it matches the real feasibility of the instances more closely. In order to incorporate the departure and arrival times of the train units, a time-expanded graph is used in this thesis. Accordingly, the time is discretized. The method to discretize time and its corresponding units are discussed in Section 4. In that same section it is also examined how this can be done without disregarding feasible solutions.

3.2 Sub-problems TUSS

The Train Unit Shunting and Service problem is described best by considering the sub-problems separately. The division of the TUSS into four sub-problems by Lentink (2006), is currently used the most by the Netherlands Railways and is therefore elaborated on. The sub-problems are considered in the following order: matching, service (which is a generalization of cleaning), parking and routing.

3.2.1 Matching

The Matching sub-problems focus on allocating arriving train units to departing train compositions. The exact order and types of train units that compose the train should be respected. It is preferred to keep train units together to the greatest extent possible as this results in fewer required movements.

The mathematical formulation can be found in Lentink et al. (2006). It is not relevant here as this thesis does not concern the entire Matching sub-problem. However, matching arriving train units to departing ones is still encompassed in this work. All trains are assumed to be of one and the same type. Therefore, it should be ensured that at least one train unit is ready at the gateway track when a train is scheduled to depart. This aspect is included in contrast to the matching of the exact order and types of the train compositions.

Train units that do not depart for the entire period can be included in the model by modelling a dummy variable that leaves after all departures took place. The same can be applied to train units that are already present at the shunt yard before the first arrival.

3.2.2 Service

In the study of van den Broek (2016) the cleaning sub-problem of Lentink (2006) is generalized to scheduling all sorts of service tasks. This generalization is relevant for this thesis. For all train units a, a set of service tasks Σ^a to be performed is given. Each of these services σ has a duration, d_σ, and a set of one or multiple tracks, T^σ, on which the service can be performed. The service tasks should be scheduled such that they are performed before the departure of the train unit.
It is assumed that there is no predefined order in which the service tasks should be performed. If a track that is able to perform services is not being used for this purpose, then it should be available for routing and parking (Mulderij et al., 2020). All service tasks require the presence of resources, such as personnel and equipment. As this thesis does not focus on Crew and Resource Scheduling, corresponding costs are not considered. Moreover, the service resources are assumed to be unlimited.
Due to the 'only one train unit type' assumption in this work, the service sub-problem translates to a generalization of the Open Shop Scheduling Problem, which is NP-complete. In this translation of van den Broek (2016), the train units are considered as jobs and the service tasks are considered as operations. Furthermore, release dates and deadlines are added to this translation depending on the exact time the train unit is present at the shunting yard.

3.2.3 Parking

In a shunting plan, trains are not always occupied with service, movements or shunting operations. Therefore, the train units should be parked at the shunting yard in the meantime. If the length of a track allows it, multiple train units could be parked on it. Due to the above mentioned modelling choice of the number of track nodes for a certain length, each track node can be utilised by one train unit at most. Obviously, the track length may not be exceeded by the summation of the length of the parked train units. Furthermore, no crossings may occur. Therefore, train units may not be parked in such a way that it obstructs a departing train from that track.
In the case that multiple trains are parked at a LIFO track, the arriving train will be parked in front of the train that arrived before it. In case of a free track, the arrival side of the train can be chosen as the track resembles a double ended queue. Thus, the train will be parked in front of the train that arrived last at the chosen side. The arrival side has to be specified for parking operations on free tracks.

3.2.4 Routing

The shunting plan schedules service tasks at service locations for the train units. Furthermore, departures can only occur at the gateway track(s). These locations should be visited by a route for each train unit. Routing is the process of finding unobstructed paths for the train movements that should occur on the shunting yard. The possibility to park a train unit at a track for a certain

period adds flexibility to the timing of the routing (Lentink et al., 2006). All movements of all train units on the shunting yard should be specified in the routing sub-problem, resulting in a path for each train unit. Preferably, the duration of these paths is minimized. It is assumed that traversing to an adjacent track takes the same amount of time for all train units.
Various types of collisions should be prevented, such that the formed routes are not obstructed. In the scope of this thesis, a minimum headway duration equal to zero is assumed. In vertex conflicts, multiple trains occupy the same vertex at the same time. While at edge conflicts, multiple trains traverse the same edge. Blocking a train unit that is arriving or departing is another type of conflict that should be prevented by the routing.

3.3 TUSS

3.3.1 Scope

As all sub-problems are explained one by one, the Train Unit Shunting and Service problem is clarified. This thesis does not aim to use the formulation of the TUSS problem, but utilises it to formulate a relaxation of the problem. Via the Multi Agent Pathfinding with Waypoints with durations formulation, a simplification of the reality is considered. As not all parts of the TUSS are included equally, it is important to state the scope of this thesis. It is expected by Mulderij et al. (2020) that focusing on the routing sub-problem could improve the process of logistic capacity determination of the shunting yards. By including waypoints, service tasks are considered as well. As parking is essential, such that train units are not always required to perform movements during a route, this sub-problem is included as well. However, the matching problem falls outside the scope of the thesis mainly due to increased computation time. Assuming no differences in train unit compositions and types enables a more concrete focus on the routing and service tasks, which is expected to improve the capacity determination.
This study focuses on investigating how the BCP framework could be used while completeness is ensured. The Pricing Problem component of this framework is proposed and tested. The purpose of this study is to determine the logistic capacity of individual shunting yards, where capacity is defined on a 24-hour interval. However, most of the arrivals and departures are likely to occur after the afternoon rush-hour and before the morning rush-hour, as most train units are then needed to serve railway passengers. Thus, most train movements on such a yard take place within an interval of eight hours, which mostly determines the capacity of a yard. Therefore, the scope of this thesis is to examine one shunt yard at a time for an eight hour period. This timespan could be easily extended later on. Shunting operations at different shunting yards are assumed to be unrelated.

3.3.2 Data

In Haahr et al. (2017) an instance of the TUSP is described. As it concerns the TUSP and not the TUSS problem, service tasks per train unit are not included. In the instance, the activity type (arrival or departure), scheduled time of this activity and train unit type are specified. If the shunting yard infrastructure is known, these instances can be used as input for the BCP approach of the MAPF formulation without waypoints in order to identify computation times of various instance sizes for such a simpler method. However, the instances need to incorporate information about service tasks per train unit for the final method proposed in this thesis. Furthermore, the infrastructure of the shunting yard needs to clearly described.
A more accurate model requires also the arrival track of an arriving train unit. Furthermore, it shoulde be specified how many train units are present at the shunting yard at the start (before all arrivals) and at the end (after all departures). For all arriving train units, the needed service tasks are required.
Concerning the infrastructure, a more accurate model would require to describe the instance by providing information about all the track parts. It is denoted to which track parts or switches the A and B sides of each track are connected and whether these sides are open. Moreover, the track length is given and it is specified whether parking is allowed per track. For switches, one is informed about which track parts they connect. For all possible service tasks at the yard, the exact tracks are provided on which the service task can take place in combination with the minimum service duration.

3.3.3 Example

In order to further clarify the TUSS problem that is utilised in this work, an example is provided. First, the hypothetical shunt yard in Figure 3 could be considered. At this shunting yard, four tracks are present. The length of each track is specified between brackets. On track 2, a service task could be performed. For completeness, the gateway track and the A and B sides are indicated.

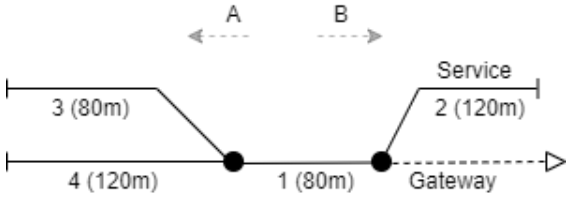

Figure 3: Example of Shunt Yard Infrastructure

Besides the infrastructure, an instance is provided in Table 1. All types of the train units are the same, as this matches the scope of the thesis. As all train compositions are of the same single type, the length is also the same. For arriving train units it is specified which service tasks should be performed. The column 'Time' indicates at what time the train unit arrives at the gateway track or should depart from it.

Activity	Train ID	Type	Length	Time	Tasks
Arrival	1	a	80	11:30	Service
Arrival	2	a	80	11:45	None
Departure		a		12:15	
Arrival	3	a	80	12:30	Service
Departure		a		15:30	
Departure		a		16:00	

Table 1: Example of Instance

As the shunting yard infrastructure and the instance are introduced, a feasible shunt plan could be constructed. If this is not possible, it could be concluded that the instance is infeasible for this specific shunting yard. The approach of the thesis is integrated, thus the sub-problems will not be handled sequentially. All previous discussed constraints should be satisfied, unless specified otherwise. An example of a feasible shunting plan is given by Table 2. In the shunting plan the subsequent tracks and corresponding activities are shown for each train unit. The following activities are included: arrival (A), traverse (T), service (S), parking (P) and departure (D). One should note that normally, the times should be determined for all movements and activities in a shunting plan. However, these are left out for the ease of exposition.

Train ID	Plan
1	G(A) - 1(T) - 2(S) - 1(T) - 3(P) - 1(T) - G(D)
2	G(A) - 1(T) - 4(P) - 1(T) - G(D)
3	G(A) - 1(T) - 4(P) - 1(T) - 2(S) - 1(T) - G(D)

Table 2: Example of Shunting Plan

3.4 Quality Measures

It is important to specify how the performance of a solution approach is evaluated. The capacity determination is performed for over 30 shunting yards in the Netherlands and at least 50 scenarios of different instances are tested on each yard (van de Ven et al., 2019). Thus the computational feasibility of the method is an important measure.

Furthermore, the tightness of the formulation plays an important role. There does not yet exist a measure for the tightness of this problem at the Netherlands Railways, thus this has to be developed. A tighter formulation is better able to point out infeasible instances as infeasible. This could be measured by the percentage of instances that are found infeasible by the method. As the approach is complete, the percentage of infeasible instances can never become higher than what is actually the case. This is due to the fact that feasible instances will never be denoted as infeasible by a complete method. Therefore, a higher percentage of infeasible instances is preferred as this implies a tighter formulation.

To clarify the quality measures, an example of five different instances on the same shunting yard with three solution approaches is given. All the three hypothetical solution approaches A, B and C

are complete. The approaches indicate whether the instances are feasible or infeasible, according to the approach, on the shunting yard given in Figure 3. It is important to note that 'feasible according to the approach' does not necessarily mean that the instance is feasible on the yard. It simply indicates that the complete approach could not determine that the instance is definitely infeasible. Often a feasible solution to a MIP is not directly a feasible real life solution, due to simplifications made. As all proposed approaches in this research are complete and are based on various relaxations and assumptions, this notion is important for the thesis.
The timetable instances are depicted in Appendix A in the same manner as Table 1 depicts the first instance. Beneath the instances in the appendix, the correct label ('Feasible' or 'Infeasible') is derived. However, even without knowing the correct labels, the quality of the solution approaches can be compared as they are all complete. In Table 3 the (not necessarily correct) labels given by the different solution approaches are shown. Based on the percentage of instances that are denoted as infeasible, solution approaches B and C are preferred as these are tighter. Solution approach B is preferred over C on these instances as the computation time is smaller. However, if computation time is a very important factor, solution approach A can be preferred despite its weaker tightness.

Instance	**Approach**		
	A	B	C
1	Feasible	Feasible	Feasible
2	Infeasible	Infeasible	Infeasible
3	Feasible	Infeasible	Infeasible
4	Feasible	Feasible	Feasible
5	Infeasible	Infeasible	Infeasible
Infeasible Percentage	40%	60%	60%
Computation time	3 min	8 min	10 min

Table 3: Labels of instances for different solution approaches

4 Multi-Agent Pathfinding with Waypoints with durations formulation of the TUSS problem

In this first section of the methodology, a translation is constructed between the Train Unit Shunting and Service problem and the Multi-Agent Pathfinding with Waypoints with durations formulation, based on Mulderij et al. (2020). In order to do so, the infrastructure of the shunting yard is represented mathematically by a graph. Secondly, a detailed explanation of the MAPF is presented. Then the translation between the TUSS and an extended MAPF is investigated, after which the TUSS sub-problems and their role in the MAPFW are discussed. The relaxations and completeness of the approach are considered throughout this section.

4.1 Shunting Yard Infrastructure

In the approach of Lam et al. (2019) the MAPF is solved on a grid with adjacent nodes in the directions: north, south, east and west. However, it seems to be more applicable to consider a graph infrastructure for the shunting yard. A shunting yard can be described by an undirected graph $G = (L, E)$. Here, L are the locations which represent the tracks of the yard. The gateway tracks, on which arrivals and departures can occur, are given in the set D which is a subset of L. The edges are denoted by E, these are the switches and connections between the tracks. A track could be represented by multiple nodes, depending on its length, l_τ, and the length of the shortest train unit. The exact number of nodes is determined by the equation derived in Section 3.1.5. As the length is divided by the shortest train unit length, enough parking space for all train units of varying length is ensured. This relaxation ensures completeness.

Although the infrastructure can be described by a connected undirected graph, some extensions need to be made such that the infrastructure can be used in this thesis. As described later in the thesis, one should keep track of the train movements and positions at specific times in order to prevent conflicts. Therefore, the concept of time is included into the graph. The set $T \subset Z_+$ encompasses all time steps. The set $V \subseteq L \times T$ denotes the time-extended vertex set, where $v = (l, t) \in V$ represents location l at time t. The time-extended edge set can be formed by connecting the time-extended vertices, $\hat{E} \subset V \times V$. Here, $e = (v_1, v_2) = ((l_1, t_1), (l_2, t_2))$ is included in the set $\hat{E}$, if and only if the edge connects time-extended vertices that are adjacent and one discretized time step away from each other. Thus, $t_2 = t_1 + 1$ and $(l_1, l_2) \in E$ holds for all edges in $\hat{E}$. Reverse edges, e', are edges that alter the direction of the movement by swapping locations, $e' = ((l_2, t_1), (l_1, t_2))$.

The inclusion of time results in the graph $\hat{G} = (V, \hat{E})$ which is used to model the infrastructure on in the rest of this thesis. It is a directed acyclic graph. Arrivals and departures can be specified by a node in this time-expanded graph. For arrivals and departures at node $v = (l, t)$, t represents the time of the activity and it holds that $l \in D$, where set D contains all possible gateway tracks. The set of arrivals (start nodes) is S and the set of departures (goal nodes) is G.

The set $\delta_{in(l,t)}$ encompasses all ingoing edges e for location l and time t. Such that $e \in \delta_{in(l,t)}$ if and only if $e = ((l_1, t_1), (l, t))$ with $(l_1, l) \in E$ and $t_1 = t - 1$. In the same manner, set of outgoing edges $\delta_{out(l,t)}$ is constructed. Here, $e \in \delta_{out(l,t)}$ holds if and only if $e = ((l, t), (l_1, t_1))$ with $(l, l_1) \in E$ and $t_1 = t + 1$.

In Figure 4 the graph representation is given of the shunting yard depicted in Figure 3. It can be seen that tracks 2 and 4 consists of two nodes each, due to the fact that $\lceil \frac{120}{80} \rceil = 2$. The gateway track is represented by one node, only one train unit can arrive or depart at the same time. In the next three subsections, multiple infrastructure extensions are considered.

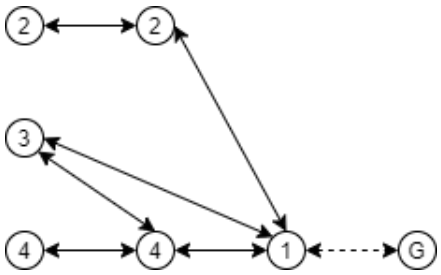

Figure 4: Graph Infrastructure of example yard

4.1.1 Reverse Movements

It is described by van den Broek (2016) how reverse movements and saw movements are quite time consuming. However, in an undirected graph with discretized time steps, this is not yet taken into account. Whenever a train unit is moving in a certain direction, the driver has to be in the front part of the train unit. Therefore, the driver has to change his place in the train unit by walking to the other side before reversing the direction of movement (Mulderij et al., 2020). Saw movements, as described earlier, always include reverse movements. Reverse movements could also occur without being part of a saw movement.

A directed graph is better suitable to incorporate the more time consuming reverse directions. This is proposed as an extension of the shunting yard infrastructure as twice as many nodes are created. The inclusion of this aspect results in a tighter, more reality based formulation. However, the exclusion of reverse movements does not disregard potentially feasible solutions. Therefore, excluding this aspect is a relaxation and the formulation of the infrastructure can be considered as complete.

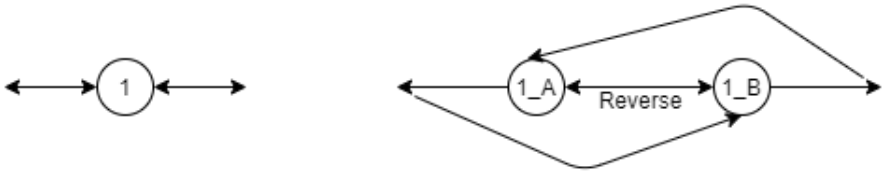

Figure 5: Extension to include reverse movements

Each node that is suitable for reverse movements is split up into two nodes, v_A and v_B. Node v_A (v_B) is visited by agents that arrive on the A (B) side. The edge between the A and B nodes is used for reverse directions and incorporates the time consumption for this action as is showed in Figure 5. Tracks on which reversion is not possible are also represented by two nodes, however the connecting edge is not included. As the A and B nodes are meant to keep track of the direction of the train unit and do not represent two physically different track parts, only one agent can use one of the nodes at the same time step.
Tracks 3 and 4 as shown in the left part of Figure 6, seem to be directly connected. However, as the angle of the turn is too small, a reverse movement has to be made by a train unit traversing between track 3 and 4. In the right part of Figure 6, reverse movements are included such that the routing aspect is incorporated in a more realistic manner. Considering a train unit of length 80 meters, track 1 is represented by 4 nodes.

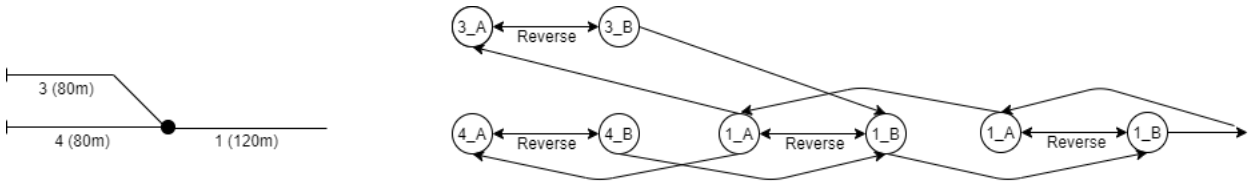

Figure 6: Translation of shunting yard to a graph with reverse extension

4.1.2 Waiting Movements

Another extension to the railway infrastructure is the inclusion of waiting movements, so-called parking. For tracks, on which it is possible, a train unit should be able to stay at the node for multiple consecutive time steps. On some nodes, such as the gateway track, parking is not possible. The set $\delta_{self(l,t)}$ denotes self-recurring edge e for location l and time t, the size of the set equals either 1 or 0. Such that $e \in \delta_{self(l,t)}$ if and only if $e = ((l,t),(l_1,t_1))$ with $(l,l_1) \in E$, $l_1 = l$ and $t_1 = t+1$. For clarity, self-recurring edge $e = ((l,t),(l_1,t_1))$ also belongs to the set of incoming edges of node (l_1,t_1) and to the set of outgoing edges of node (l,t). This extension is depicted in Figure 7, on tracks 4 and 1, where parking is possible, the waiting edges are included on the right side of the figure.

Figure 7: Inclusion of waiting edges.

4.1.3 Gateway Modelling

The last extension concerns the way gateway tracks are modelled. Multiple decisions could be made concerning this topic. The first proposition is to create two separate nodes for gateways, in order to ensure that trains could not traverse over this track. The other option is to exclude edges

to and from the gateway tracks if no arrivals or departures occur.
For the first option, two nodes are created for each gateway node as depicted in the center of Figure 8. The first one is the arrival gateway (Ga), from which train units can only enter the yard. Thus train units cannot visit it anymore once they are located at other tracks at the yard. The other one is the departure gateway (Gd), train units can only arrive at this node, thus not return. Arrival gateway locations can be found in set D^s and departure gateway locations in set D^g, where $D^s \cup D^g = D$.
The second option does not require multiple gateway nodes, it only excludes edges. Outgoing gateway edges are excluded if the starting location and time of the agent do not correspond with the edge. Ingoing gateway edges are excluded if the departure times do not correspond. This could be seen in the right part of Figure 8. As this method of modeling the gateways is more efficient than the first extension, only this method is implemented and tested.

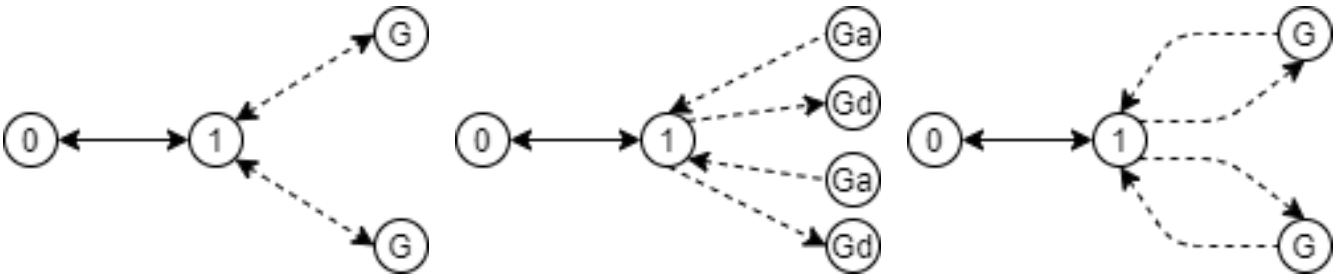

Figure 8: Left: No extension. Middle: Gateway extension 1. Right: Gateway extension 2.

4.2 Discrete Time

Without additional assumptions, the discretization of time will result in an incomplete approach. Once the size of a time step is decided, all actions will be performed on multiples of that time step (which could be found in the set T). Even if the time step is set equal to 1 second, it leads to an incomplete approach. This is due to the fact that it is possible for a train to leave a track after parking on a continuous scale of time for example. Therefore, feasible paths where movements occur that do not happen at exactly the multiples of that time step are disregarded. It is important to note this difficulty for further research. However, an approach can be formed that is 'complete in practice' according to the NS. One of the considerations is that decisions are not made continuously in practice, but at certain time-intervals.
For all movements (transitions between nodes), one time step occurs. The time step may not be greater than the time that is needed to perform a movement. This would lead to the need of more time for a movement and thus to an incomplete approach. Therefore, the time step size has to be smaller than (or equal to) the time of the smallest movement. Computational considerations lead to choosing the biggest time step possible, as this ensures the smallest number of variables. As for all movements time steps occur and the time step will be greater than zero, cycling between time-extended nodes is not possible. It can be seen in the time-expanded graph in Figure 9 that

movements are always forwards in time.

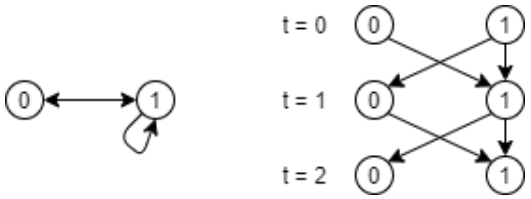

Figure 9: Transforming the left graph to the time-expanded graph on the right.

4.3 Multi-Agent Pathfinding

In the Multi-Agent Pathfinding problem, multiple agents are allocated to paths that do not collide with the other selected paths. The set A encompasses all agents a. The paths from which an agent can choose, are formed of adjacent nodes. The first node is the starting location and the last location is the goal. Agent a can select paths from the set P_a. Whether agent a decides to use path p is indicated by the binary variable λ_p^a. The cost of this path is given by c_p^a. For each agent, at least one path has to be picked and the paths can never be used negatively.
Via the parameters x_v^p and y_e^p, which follow directly from the path generation, the relation between the path and the graph is given. The parameter x_v^p equals 1 if path p has an edge that visits vertex $v = (l, t)$, it is 0 otherwise. Whether edge $e = (v_1, v_2)$ is visited in path p is indicated by the binary parameter y_e^p. The formulation of the LP-relaxation of the MAPF, where paths can be selected fractionally, is given below.

$$\min \sum_{a \in A} \sum_{p \in P_a} c_p^a \lambda_p^a \tag{1}$$

$$\text{s.t.} \sum_{p \in P_a} \lambda_p^a \geq 1 \qquad \forall a \in A \tag{2}$$

$$\sum_{a \in A} \sum_{p \in P_a} x_v^p \lambda_p^a \leq 1 \qquad \forall v \in V \tag{3}$$

$$\sum_{a \in A} \sum_{p \in P_a} (y_e^p + y_{e'}^p) \lambda_p^a \leq 1 \qquad \forall e \in \hat{E} \tag{4}$$

$$\text{Additional conflict constraints} \tag{5}$$

$$\lambda_p^a \geq 0 \qquad \forall a \in A, \forall p \in P_a \tag{6}$$

The proposed formulation also includes constraints that prohibit collisions. It is common to include constraints that prevent vertex and edge conflicts. Additional constraints (5) could tighten the formulation further and are described later. In constraint (3) and (4), the parameters y_e^p, $y_{e'}^p$ and x_v^p are utilised that indicate whether edge e, reverse edge e' or vertex v are used in path p. In the vertex constraints (3), agents are disallowed to occupy the same vertex at the same moment in

time. In the edge constraints (4), agents are prevented from traversing the same edge or its reverse edge simultaneously. By the path selection constraints (2), it is ensured that all agents select at least one path. The costs of all selected paths are minimized by the objective function (1).
It can be seen that in this formulation constraints occur on multi-agent level. The set P_a only includes paths that are feasible on single-agent level. For the single-agents cycling and waiting during a path is not forbidden in the MAPF problem, thus the set P_a can be infinite in size (Lam et al., 2019). This is a problem that will be addressed later on in the thesis.

4.4 TUSS translation to MAPF

As the MAPF is described above, a translation approach will be provided in this section. Each train unit can be translated to an agent in the MAPF formulation. The set A is thus filled with all arriving train units. The start and goal vertex for each agent, can be translated to the arrival and departure location on single-agent level. In order to have a complete approach, each train unit should be able to use all departure locations as goal location. Due to the 'only 1 train type' assumption, all arriving train units could be matched with all departures. The train units still have one start location each, however they have a set of possible goal locations now.
The graph that is used in the formulation is the previously explained time-expanded graph $\hat{G} = (V, \hat{E})$ on which the shunting yard infrastructure is modelled. Therefore, conflicts do not occur when the same vertices or edges are visited by agents at different times. This is due to the fact that vertices or edges on dissimilar times are formulated as different nodes and arcs in the time-expanded graph.

4.4.1 TUSS sub-problems

Components of the four TUSS sub-problems take place at different levels: either single-agent level, multi-agent level or both. The paths are created on single-agent level and they are selected on multi-agent level. In the matching sub-problem (in the scope of this thesis) arriving train units are assigned to departing ones. This occurs on both levels. On single-agent level, paths are created that decide which departure is picked for the arriving train unit. On multi-agent level, it should be ensured that all departure locations are selected at least once. The completeness of this formulation is guaranteed, as described in the previous section.
The parking sub-problem is mainly incorporated on single-agent level, as this is where the paths are created. Therefore, it is decided on this level where and when the train units park. The vertex conflicts that can occur due to parked trains are resolved on multi-agent level.
The same applies to the routing sub-problem. On single-agent level it is determined which vertices are visited by the train unit and which arcs are traversed at certain times. However, the conflicts that can occur from such a route are resolved on multi-agent level. Only infeasible solutions are prohibited by avoiding vertex and edge conflicts, as these scenarios cannot occur in a real life shunting plan. Therefore, regarding the routing and parking sub-problems the approach does not

disregard feasible solutions on multi-agent level. The completeness of the more detailed aspects of routing and parking, on the single-agent level, are discussed later on.
In the current formulation, service tasks are not incorporated. However, as an extension to the model the service tasks can be included on a single-agent level by defining a set of waypoints that should be visited. The waypoints represent the service locations. If the extension of the service tasks is not included, feasible solutions will not be disregarded as it is a relaxation of the reality. The completeness of including this extension will be discussed later on in the thesis when the exact approach is described.

4.4.2 Extensions to MAPF

Considering all the sub-problems, one can conclude that some extensions should be made to the Multi-Agent Pathfinding formulation presented in Section 4.3. Most extensions concern the feasibiliy of the paths on a single-agent level and are therefore described when the generation of paths in set P_a is discussed. Furthermore, additional conflict constraints can be added to tighten the formulation. In order to keep the formulation correct, it should be guaranteed that all departures are matched. This can be ensured by adding the following constraint to the formulation:

$$\sum_{a \in A} \sum_{p \in P_a} z_g^p \lambda_p^a \geq 1 \qquad \forall g \in G \tag{7}$$

Here, G is the set of all goal locations as denoted earlier. This set is not agent specific as all train units are of the same type. The constant z_g^p indicates whether goal location g is used for a departure in path p. All goal locations should be used at least once, thus all departures will be satisfied by including this constraint. Due to the minimization of the objective function, all goal locations will also be used at most once. It could be argued that the departure constraints are already satisfied by including the vertex constraints. This could be true, depending on the formulation of the graph infrastructure. For a graph where $\mid A \mid = \mid G \mid$, it is forced by the vertex constraints that all departures are used. A departure is a combination of time and location, thus multiple usage of one is seen as a vertex conflict. In the next section it will become evident why it could be valuable to include these departure constraints, (7), directly.
Lastly, to add the Service facet of the problem; the waypoints will be included on a single-agent level. The waypoints should be visited for at least a certain service duration. The graph representation of the infrastructure is extended to include service tracks. It is possible that multiple services have to be performed, which each have multiple corresponding service tracks. Thus, regarding completeness; at least one waypoint should be visited in each set of waypoints in order to perform the service corresponding to that set of waypoints. Concluding, for each service in the service set, a set of waypoints is added. For all waypoint sets, at least one waypoint has to be visited for at least the service duration. Thus, the problem extends to a Multi-Agent Pathfinding with Waypoints with durations (MAPFW).

5 Branch-and-Cut-and-Price framework to solve the MAPFW formulation

A common way to solve MIPs is utilizing the Branch-and-Bound method. Problems that contain a lot of variables are often solved by column generation. Applying column generation at each node in Branch-and-Bound is known as Branch-and-Price. Performing Row Generation (column generation for constraints) at the Branch-and-Bound nodes is called Branch-and-Cut. Combining the three, results in an approach named Branch-and-Cut-and-Price. This approach, based on the work of Lam et al. (2019), is explained in this section by splitting it up into Branch-and-Bound, Branch-and-Price and Branch-and-Cut. The framework is applied to the Multi-Agent Pathfinding with Waypoints with durations formulation of the Train Unit Shunting and Service problem.

5.1 Branch-and-Bound

MIP problems that are too hard to solve directly, are often split up into different smaller problems in the Branch-and-Bound approach. As our MAPFW formulation concerns a minimization problem, minimizing is considered in this explanation as well.

In the root node the whole feasible region of the problem is considered. The process of branching splits this region into disjoint sets, but first the bounds at the nodes need to be determined. In order to determine a lower bound of the node, a relaxation is solved. This is commonly the LP-relaxation, thus integrality of the variables is dropped. The relaxed solution is potentially infeasible due to the lack of integrality in the decision variables. The objective value can be used as a lower bound of the node. The upper bound is the best known feasible solution. These are usually found by applying a simple heuristic to construct a feasible solution from the relaxed solution.

The relaxed solution is oftentimes fractional when the LP-relaxation is applied at the node. In order to come a step closer to an integer solution, the Branch-and-Bound method branches on a fractional part of the solution. A branching rule constructs two disjoint sets from the original set of the node. In these two sets (the nodes that are investigated next) the fractional part will not be present. For example, if $\lambda_p^a = 0.2$, this fractionality could be resolved by creating one set with $\lambda_p^a = 0$ and another with $\lambda_p^a = 1$. The two newly formed sets are called the child nodes of the original node. After the creation of the child nodes, the same process is performed at these nodes. The integrality restrictions of all parent nodes hold at the child nodes. Thus, repeating this process will result in an overall integral solution.

After branching, it is determined which node to investigate next. Common rules are selecting depth-first or best-first. However, based on bounding strategies, it can be decided to not investigate a node further. This could be determined when the upper and lower bound of a node are equal, this particular node is then already solved to optimality. If the lower bound of a node is higher than the current best known solution (an upper bound), then the node does not contain the optimal solution, thus it can be disregarded. Tighter formulations, that lead to higher lower bounds, are

helpful to increase the speed of the Branch-and-Bound method as more nodes can be disregarded at an earlier stage.
It should be noted that in this BCP method, pruning could occur less due to the fact that the goal is to prove (in)feasibility. The upper bound is a feasible solution, often derived from the lower bound. However, if such a feasible solution is found, then it is not used to prune nodes by optimality or by a lower bound that is higher. Instead, the BCP method could be stopped directly, as a feasible solution is found.
The branching only removes solutions that contain fractional variables. As these could not occur in real life shunting plants, no feasible solutions are disregarded. In bounding, sub-optimal solutions are only removed when another superior feasible solution is already discovered. Thus, if a feasible shunting plan exists it will be found with the Branch-and-Bound approach. Therefore, the method can be considered as complete and be used for the approach in this thesis.

5.2 Branch-and-Price

As mentioned in Section 4.3, authorizing waiting and cycling for train units leads to a potentially infinite size of the pool of paths P_a for agent a. This applies to problems assuming an infinite time horizon. Waiting, which is known as parking, cannot be forbidden as it is possible in real life situations. Therefore, prohibiting waiting will lead to an incomplete approach. The same applies to cycling movements (over locations, with increasing time). Prohibiting them is not an option in a complete method due to the real life possibility of circular movements by train units in a feasible shunting plan.
Allowing waiting and cycling does not lead to an infinite set of paths in our problem due to the specified time interval in the scope of the problem (8-hours). However, the possibility to wait and cycle does substantially increase the amount of possible paths. A well-known method to deal with MIPs with a very large number of variables is column generation. Column generation applied in the Branch-and-Bound framework, to solve the relaxations at each node, is called Branch-and-Price.
As the additional column generation is applied at each node, the process at the nodes is described here. Initially some or all variables (the train unit paths), are excluded from the Master Problem. The Restricted Master Problem is the LP-relaxation of the original problem with fewer variables included in set $\hat{P}_a$. This is similar to prefix some or all variables λ_p^a to zero, these variables determine with what fraction path p is selected by train unit a. After solving the relaxation with this smaller number of variables, the Pricing Module is called. This module determines, by solving the Pricing Problem, which variables are promising to contribute to the Restricted Master Problem. Each iteration one or multiple train unit paths are added to the set $\hat{P}_a \subseteq P_a$. This is repeated until it is proven by the module that no additional variables could improve the objective function in the Restricted Master Problem. Then, it can be concluded that the determined lower bound is not different from the lower bound with all variables included.

Branch-and-Price includes a Pricing Module that generates paths for the train units. The Pricing Module should be complete and may not disregard potentially feasible single train unit paths. The bounds of Branch-and-Price are in accordance with the Branch-and-Bound method, thus the same solutions are considered as in the latter complete approach. Furthermore, it is proven by the Pricing Module that no promising variables are disregarded. Therefore, it can be concluded that Branch-and-Price is a complete approach if the Pricing Module is complete. It is assumed that the Pricing Module is complete for now, as it will be constructed later in this study. Therefore, the complete Branch-and-Price method can be utilised in this thesis.

5.3 Branch-and-Cut

As there is no need to include all variables, it could also be efficient to exclude some constraints. Constraints that prohibit vertex or edge conflicts that are currently not in use are not needed (Lam et al., 2019). Furthermore, after all train units have departed from the shunting yard, there is no need to check for conflicts at the yard. Moreover, the valid inequalities, which are also called cutting planes, could be added iteratively to strengthen the formulation. Applying this procedure in the nodes of the Branch-and-Bound framework, is called Branch-and-Cut.

In Branch-and-Cut some or potentially all constraints are omitted from the Master Problem at each node. The LP-relaxation is solved without these constraints. The constraints in the original problem are split up into classes. Constraints set (1) is included in the Restricted Master Problem from the start. Constraints sets (3) and (4), the vertex and edge conflicts, are excluded. Constraints set (7), the departure constraints, are not excluded from the Restricted Master Problem. The departure constraints are similar to those for vertex conflicts that only focus on the gateway departure vertices. As it is known that these gateways will be used in all generated paths, it could be valuable to include the constraints from the start. This could be examined by including constraints set (7) and by ignoring constraints set (7), such that the constraints will be iteratively generated as vertex conflicts. A module called the Separator is called for each class of constraints that is excluded.

The Separator module identifies per class whether one or more constraints are violated. Therefore, these violated constraints are considered helpful and should be added to the Restricted Master Problem. Otherwise, it is concluded that all constraints are currently satisfied in the solution. Then, the current solution at the Branch-and-Bound node is fractionally optimal. The Seperator module is called repeatedly until no violated constraints are found, afterwards new branches are created by the original Branch-and-Bound framework.

The completeness of the Branch-and-Cut approach, given that Branch-and-Bound is complete, follows quite logically from the fact that constraints are possibly excluded from the original formulation. As no additional constraints are formed, no potentially feasible solutions are disregarded compared to the original problem. Furthermore, the bounds correspond with the bounds proposed by Branch-and-Bound as it is proven by the Seperator module that no more violated constraints

can be added in the last iteration.

5.4 Branch-and-Cut-and-Price

Incorporating both Branch-and-Price and Branch-and-Cut into the Branch-and-Bound framework leads to an approach called Branch-and-Cut-and-Price. In this procedure the Branch-and-Bound approach is applied and therefore multiple nodes are created by branching. At each node the relaxation is solved by initially excluding some or all variables and constraints.
The Restricted Master Problem, that is solved at each node of the Branch-and-Bound problem corresponds with the following formulation:

$$\min \sum_{a \in A} \sum_{p \in \hat{P}_a} c_p^a \lambda_p^a \tag{8}$$

$$\text{s.t.} \sum_{p \in \hat{P}_a} \lambda_p^a \geq 1 \qquad \forall a \in A \tag{9}$$

$$\sum_{a \in A} \sum_{p \in \hat{P}_a} z_g^p \lambda_p^a \geq 1 \qquad \forall g \in G \tag{10}$$

$$\lambda_p^a \geq 0 \qquad \forall a \in A, \forall p \in \hat{P}_a \tag{11}$$

It can be seen that this resembles a smaller version of the Multi-Agent Pathfinding with Waypoints with durations formulation. One should note that the set $\hat{P}_a \subseteq P_a$ does not include all paths for train unit a. Furthermore, the vertex and edge constraints, (3) and (4), are excluded. For now, the departure constraints (10) are included. This is the problem that is initially solved at the first node, thus vertex and edge conflicts are allowed.
The sets $\hat{P}_a$ can be initialized by artificial variables with costs that are huge, such that the variables will never occur in a feasible optimal solution. However, the sets can also be filled with realistic paths. The local search method of van den Broek (2016) is used at the Dutch Railways to create shunting plans for operations. Feasible paths that follow from that model could be included as initial paths. Furthermore, a heuristic could be used to generate the first paths that are added initially. An example of such a heuristic is generating paths from the arrival node to only the first few following departures or the last few departures.
After the Separator module is called for all constraint classes, some combination of currently selected paths could be cut off. It is possible that for a specific train unit, all currently known paths are prohibited. However, as constraints set (9) states that all train units should select at least 1 path in the Restricted Master Problem, this behaviour is not possible. Thus, new paths need to be generated by the Pricing Module after the Separator added new constraints to the Restricted Master Problem. Fractionally selected paths can also be cut off by Valid Inequalities, these will be described in Section 5.4.2. After it is proven that no more violated constraints or promising variables can be added, the node is branched upon. The order of the algorithm of the

LP-relaxation at the node, thus within the Branch-and-Bound procedure, is described below:

Algorithm 1 Node LP-relaxation algorithm

1. Perform Column Generation (to ensure that paths can be selected after branching).
2. Iterate the following:
(a) Call Separator Module per constraint class (until proven that no more constraints are needed).
(b) Add Valid Inequalities (until arbitrarily selected rule determines to stop, not needed to add all).
(c) Column Generation (until proven that no more columns are needed).
3. Branch if the LP-relaxation is fractional.

As the obtained solution could consist of fractionally selected variables, branching is performed to resolve a specific fractionality. Two child nodes are created at which that fractionality could not appear. In this Multi-Agent Pathfinding with Waypoints with durations approach, the BCP branches on agent-vertex pairs as in Lam et al. (2019). Furthermore, the BCP could branch on departure time.

5.4.1 Branching

First, branching on agent-vertex pairs is explained. The earliest vertex utilised fractionally by at least two train units is chosen to branch upon. The train unit with the shortest path that uses the vertex fractionally is selected to use the vertex integrally in one child node and prohibited to use the vertex in the other child node. Prohibiting a vertex-agent pair in a child node is done by simply excluding that specific vertex from the graph of the Pricing Problem corresponding to the agent. Enforcing an agent to use a vertex, in the other child node, is performed by ensuring that the sum over all ingoing edges of that vertex is at least 1 in the Pricing Problem of the agent. Integer solutions are not disregarded as these could be found in one of the two branches, thus this branching rule is complete.
Additionally to this branching rule, it is also possible to branch on departure time. The selected departure time directly influences the path length as the path length is the summation of all time steps between the arrival and the departure. This branching rule chooses the agent with the shortest fractionally selected path to branch upon, from a set of all agents with multiple fractionally selected paths with different path lengths. In one child node, the paths should have a length less or equal to this shortest path length. In the other child node, all paths should be larger.
One should note that all paths of a train unit that satisfy the same departure are equal in length. This is due to the fact that the start and departure node are formed on a time-expanded graph, thus the time steps (path length) in between the arrival and departure are equal. Therefore, all departures that are after a certain point in time could be dismissed in the first child node. In the second child node, all departures before that point in time are disregarded. Therefore, this branching rule branches on departure time. Only fractional solutions are prohibited by this branching rule, thus no feasible integer solutions are disregarded.

5.4.2 Valid Inequalities

Constraints that are added to the Restricted Master Problem can be split up into problem constraints and redundant constraints. Without adding 'enough' problem constraints, the problem is not correctly described. The path selection, vertex and edge constraints are needed to describe the problem, therefore they are classified as problem constraints. The set of redundant constraints consists of constraints that are not necessary to form the problem formulation. However, these constraints could tighten the gap between the upper and lower bound by disallowing fractionality. In the case of this method, adding redundant constraints could improve the lower bound as this is the solution to the LP-relaxation. As the lower bound becomes higher, the gap between the upper and lower bound becomes tighter. Therefore, it could be decided earlier in the Branch-and-Bound process that a certain node does not need to be inspected anymore. Adding redundant constraints iteratively is called adding Valid Inequalities or adding Cutting Planes.
Due to the way the shunting yard is modelled as a graph, many corridors will form. This is caused by tracks with a length greater than the length of the shortest train unit, as these tracks are represented by multiple bidirectionally connected nodes. Besides, corridors will be formed by tracks that are only connected to each other.
The analysis of the graph structure indicates that adding Valid Inequalities focused on corridors are expected to be of value for our model. The corridor conflicts of Lam et al. (2019) ensure that agents cannot cross corridors in opposite directions. Figure 10 shows how the red agent and the blue agent can cross a corridor without violating edge and vertex conflicts by selecting all red and blue movements with 0.5 at t. At $t+1$, 0.5 fraction of the red agent is left on node 2 and 0.5 fraction of the blue agent is left on node 1. These fractions could pass each other in the next time step and thus the agents have crossed each other in the corridor at $t+2$.

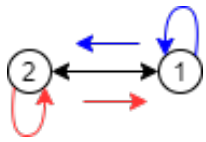

Figure 10: Corridor crossing

The constraint that resolves corridor conflicts ensures that at most 1 crossing movement is selected on $(t, t+1)$ and $(t+1, t+2)$ by both agents. In this case a crossing movement is defined as a movement from location 1 to location 2, or in reverse. The constraint is formed as follows:

$$\sum_{p \in P_{a_1}} y^p_{((l_1,t),(l_2,t+1)} \lambda^{a_1}_p + \sum_{p \in P_{a_1}} y^p_{((l_1,t+1),(l_2,t+2)} \lambda^{a_1}_p + \sum_{p \in P_{a_2}} y^p_{((l_2,t),(l_1,t+1)} \lambda^{a_2}_p + \sum_{p \in P_{a_2}} y^p_{((l_2,t+1),(l_1,t+2)} \lambda^{a_2}_p \leq 1 \qquad \forall (a_1, a_2, l_1, l_2, t) \quad (12)$$

Here $a_1, a_2 \in A$, $t \in T$ and $l_1, l_2 \in L$, where it must hold that the agents are not the same and the

locations are adjacent in a corridor. This constraint could be further strengthened by considering all agents a_2. This is justified by the fact that a train, a_1, moving from location 1 to 2 in a corridor, could not cross any approaching trains. Thus all movements from location 2 to 1 for all other train units could be included into the constraint.

$$\sum_{p \in P_{a_1}} y^p_{((l_1,t),(l_2,t+1)} \lambda^{a_1}_p + \sum_{p \in P_{a_1}} y^p_{((l_1,t+1),(l_2,t+2)} \lambda^{a_1}_p + \sum_{a_2 \in A \setminus \{a_1\}} \sum_{p \in P_{a_2}} y^p_{((l_2,t),(l_1,t+1)} \lambda^{a_2}_p + \sum_{a_2 \in A \setminus \{a_1\}} \sum_{p \in P_{a_2}} y^p_{((l_2,t+1),(l_1,t+2)} \lambda^{a_2}_p \leq 1 \qquad \forall (a_1, l_1, l_2, t) \quad (13)$$

Regarding the completeness of the problem, it is not possible to also include all agents a_1. This would lead to considering the movements of the same agent in both directions; from location 1 to 2 and the reverse movements from 2 to 1. In a feasible path, it is possible that a train unit first moves from 1 to 2 by using $y^p_{((l_1,t),(l_2,t+1)}$ and then moves back by using $y^p_{((l_2,t+1)(l_1,t+2)}$ a time step later. Therefore, not all agents a_1 could be considered as this could potentially cut off feasible paths. Considering the main scope of the thesis, the Pricing Problem, currently valid inequality (12) is used and inequality (13) is recommended to be applied in further research.

The Rectangle symmetry constraints of Lam et al. (2019) seem to be less applicable to a graph structure. This set of Valid Inequalities is formed for grid maps, where each node has neighbours in four directions. In the work of Lam and Le Bodic (2020), five additional Valid Inequalities are formed. These could be included in the Branch-and-Cut-and-Price framework. However, for the explanation behind the Valid Inequalities, Lam and Le Bodic (2020) is referred to as this falls outside the scope of this thesis.

Valid Inequalities are used to cut off fractional solutions. Integer solutions are not disregarded. As feasible solutions should consist solely of integers, adding Valid Inequalities is complete.

5.4.3 Costs of Shunting Plans

As discussed in Section 4.1, about the infrastructure of the graph, the graph is constructed in such a way that all movements between nodes take 1 discretized time step. Therefore, the cost of each move equals 1. One should note that the objective value is constant, as it always equals the sum over all the departure times minus the sum over all the arrival times: $\textit{Constant objective value} = \sum_{i=0}^{|A|} \Delta_i - \sum_{i=0}^{|A|} \Psi_i$. Where Δ is the set of all possible departure times and Ψ is the set for all arrival times. Both set sizes equal the number of train units.

It could be altered by assigning less costs to waiting edges, which could be preferred by the NS as it requires less performed actions. Secondly, the costs of reversing could be increased and incorporated. However, such considerations are more relevant for projects with an operational scope. As this thesis aims for determining the feasibility of an instance to determine the logistic capacity it is less relevant to adapt the cost of waiting and reverse edges.

Furthermore, the constant objective value can be altered by including the usage of artificial variables (single agent paths) with relatively high costs. The moment the objective value equals the known constant it is known that a shunting plan is feasible as no artificial variables are selected. At this point, the method of Branch-and-Cut-and-Price can immediately be stopped as a feasible shunting plan is found. The goal was to prove (in)feasibility, not optimality.

6 Pricing Problem

The Pricing Problem is able to add promising new paths as the objective function equals the reduced costs. How this works is described in this section. Furthermore, the practical requirements of the singe train unit shunting paths is covered. Based on these requirements two MIP formulations are created. Throughout the section completeness of the approach is considered.

6.1 Objective Function

The objective function of the Pricing Problem is constructed such that it equals the reduced costs of paths in the Restricted Master Problem. It can be derived from the dual variables of the Restricted Master Problem. Therefore, dual variables are introduced for all constraints. The dual variables for path selection constraints (2) are denoted by $\alpha_a \geq 0, \quad \forall a \in A$. For the vertex constraints (3), the dual variables are $\beta_v \leq 0$ for all vertices in set V. Variables $\gamma_e \leq 0, \quad \forall e \in \hat{E}$ represent the duals of the edge constraints (4). The dual variables $\eta_g \geq 0$ for all departure nodes correspond to the departure constraints (10). Lastly, $\pi_{a_1,a_2,l_1,l_2,t} \leq 0$ denote the dual variables of the corridor constraints (13).

As discussed in 4.4.2, constraint set (10) could be included but is not necessary for formulating the model correctly. Inclusion of these constraints and thus the dual variables, leads to potentially negative edge costs. While exclusion results in non-negative edges, which is proved in the next section. This also leads to the observation that the objective function of the Pricing Problem has resemblance with the objective function of the Pricing Problem in Lam et al. (2019).

Originally, the primal cost of a path, c_p, is calculated by summing all the needed time steps of the movements. All movements in the graph take exactly 1 time step. Therefore, paths between the same arrival node (thus same train unit) and the same departure node cost the same amount. However, one could imagine that some paths are preferred over others. For example, paths that use edges or vertices on which no conflicts occur are preferred over paths where conflicts occur. This preference is taken indirectly into account by assigning the reduced costs to the used edges or vertices in the path, as specified in the objective function below.

Based on the dual variables, it becomes evident that the reduced costs in the objective function consists of multiple parts with different purposes. The original costs c_p equals the sum of all time steps in the path and hence is always positive. The need for a new path for agent a is indicated by the term $-\alpha_a$, which is always less than or equal to 0. The term that penalizes usage of vertex v in path p, is represented by $-\sum_{v \in V} \beta_v x_v^p$. This term is always non-negative as $x_v^p \in \{0, 1\}$ and $\beta_v \leq 0$. Additionally, there is a term that penalizes usage of edge e or e' in the path by assigning costs $-\sum_{e \in \hat{E}} \gamma_e (y_e^p + y_{e'}^p)$ to paths. This penalty is always non-negative as $\gamma_e \leq 0$ and $y_e^p \in \{0, 1\}$. Using edges that occur in corridor conflicts of constraint (12) is penalized for agent a_1 by the non-negative term $-\sum_{e \in \hat{E}} (y^p_{((l_1,t),(l_2,t+1))} + y^p_{((l_1,t+1),(l_2,t+2))}) \sum_{a_2 \in A \setminus \{a_1\}} \pi_{a_1,a_2,l_1,l_2,t}$. Here is summed over all edges instead of over time t and neighbours l_1 and l_2, as the edges encompass the movements

between neighbours at all times. Furthermore, the value of the dual variable $\pi_{a_1,a_2,l_1,l_2,t} = 0$ for all edges that are not located in corridors.
All individual terms are introduced, thus the objective function can be constructed. The reduced costs of a path p equals

$$\hat{c}_p = c_p - \alpha_{a_1} - \sum_{v \in V} \beta_v x_v^p - \sum_{e \in \hat{E}} \gamma_e (y_e^p + y_{e'}^p) - \sum_{e \in \hat{E}} (y^p_{((l_1,t),(l_2,t+1))} + y^p_{((l_1,t+1),(l_2,t+2))}) \sum_{a_2 \in A \setminus \{a_1\}} \pi_{a_1,a_2,l_1,l_2,t} \tag{14}$$

for agent a_1. If constraint set (10) is included, the dual costs are expanded by including the term $-\sum_{g \in G} \eta_g z_g^p$, which is non-positive. More insight into the interaction of reduced costs and the selected path is provided in Section 7.1.

6.1.1 Linking penalties to edges

To assign the reduced costs to the components of paths, it is decided to link the penalties with the edges because most dual cost terms are expressed in edge usage. The total reduced costs for a path p for agent a_1 is constructed as $\hat{c}_p = -\alpha_{a_1} + \sum_{e \in \hat{E}} c_e y_e$. Thus apart from $-\alpha_{a_1}$, all terms should be expressed in edge usage. The vertex usage penalty for vertex v are assigned to all incoming edges of that vertex, thus all edges in set $\delta_{in(v)}$. Therefore the dual variable $\beta_{(l_2,t+1)}$ should be used when costs for edge $((l_1,t),(l_2,t+1))$ are determined. In a path, at most 1 incoming edge is used, thus the penalty cannot be assigned too many times in a correct path. In order to construct c_p all edge usage costs are increased by 1, as usage of an edge corresponds with usage of 1 time step. The corridor conflicts corresponding to the dual variables $\pi_{a_1,a_2,l_1,l_2,t}$ and $\pi_{a_1,a_2,l_1,l_2,t-1}$ are relevant to edge $e = ((l_1,t),(l_2,t+1))$ for agent a_1 and should thus be included.
It can be seen that no negative costs are assigned to the usage of an edge. In fact, the only negative costs that are part of the objective function are $-\alpha_a$. Therefore, the usage of edges never leads to negative costs. The costs of an edge is given by

$$c_e = c_{((l_1,t),(l_2,t+1))} = 1 - \beta_{(l_2,t+1)} - \gamma_{((l_1,t),(l_2,t+1))} - \gamma_{((l_2,t),(l_1,t+1))} - \sum_{a_2 \in A \setminus \{a_1\}} (\pi_{a_1,a_2,l_1,l_2,t} + \pi_{a_1,a_2,l_1,l_2,t-1}) \tag{15}$$

If constraint set (10) is included, the edge costs could be expressed in a similar way as in (15) and the reduced cost function could be expanded. Then the reduced costs of path p is represented by $\hat{c}_p = -\alpha_{a_1} + \sum_{e \in \hat{E}} c_e y_e + \sum_{g \in G} \sum_{e \in \delta_{in(g)}} \eta_g y_e$.

6.1.2 Adding new variables

The goal of the Pricing Problem is to add new valuable variables to the Restricted Master Problem. Based on the reduced costs of the new variable, it could be decided whether this variable should

be added. The objective function of the Pricing Problem is formed based on the reduced costs. The Restricted Master Problem is a minimization problem, thus variables with negative reduced costs should be added as they appear promising. Therefore, the Pricing Problem seeks to minimize the reduced costs. Variables with $\hat{c_p} < 0$ should be included. If the optimal value of the Pricing Problem is non-negative, then it is proven that no feasible paths with negative reduced costs exists.

6.2 Requirements Feasible Paths

In the Pricing Problem feasible paths for the train units are generated. These paths are described by the edge movements of the agent. Due to the inclusion of time, nodes and edges could only be used once in a path. Thus an elementary path is generated. Such a path should be feasible in real life if vertex and edge conflicts do not occur. This section indicates the requirements that should hold on a single-agent level for the generated paths.

In each Pricing Problem one train unit is considered. Each train unit is represented by a distinct combination of arrival time and location. Thus, the Pricing Problem is executed once for all different arrivals. Hence, for a single Pricing Problem the size of the set of arrivals equals 1 train unit. This train unit should be free to choose a suitable departure from all departures, the matching is not prefixed. The size of the set of departures equals the number of train units in the instance. The flow during a feasible shunting plan should equal 1. The arrival node has a negative flow balance, as there is 1 outgoing flow and no flow enters. At the departure nodes flow could only occur at the incoming edges, resulting in a non-negative flow balance. At all other nodes flow conservation should hold, thus flow that arrives at the nodes should also depart the next time step. For a node at time t, waiting movements at $t-1$ are considered as incoming flow, while the same movements at t are outgoing flow.

Cycling constraints are not present in the Pricing Problem as cycle movements are possible for train units in a feasible shunting plan. Cycling is only allowed over locations if the time steps increase with each movement. Cycling in stationary time should not be allowed and is discouraged by the positive edge costs. Parking, thus waiting movements, should be possible. However, it is disallowed for train units to wait on tracks for which it is specified that parking is not possible, such as gateway tracks.

A train unit should have performed each service task in the set of its service tasks. For all tasks, it should be able to choose the time and location where the service task is performed. This allocation should not be prefixed. A service task is only considered 'to be performed' by the NS when the train unit is for at least the duration of the service task present at the same service track. Thus, performing a service could not be split up. The train unit can perform its service on one of the possibly multiple service tracks. Furthermore, a service track cannot be used to performed multiple different types of service. For all train units, it has to be allowed to traverse over service tracks when no service is provided.

6.3 MIP1 Formulation

The flow of a path in the Pricing Problem is represented by binary variables y_e that equal 1 if the path uses edge e in the path and 0 otherwise. For train unit a, the first MIP formulation (referred to as MIP1) could be constructed as follows:

$$\min \sum_{e \in \hat{E}} c_e y_e \tag{16}$$

$$\text{s.t.} \sum_{\delta_{in(v)}} y_e - \sum_{\delta_{out(v)}} y_e = \begin{cases} 0 & \forall v \in V \setminus \{S^a, G^a\} \\ -1 & \forall v \in S^a \end{cases} \tag{17}$$

$$\sum_{e \in \delta_{in(\tau,t)}} y_e + \sum_{t_1=t}^{t+d_\sigma-2} \sum_{e \in \delta_{self(\tau,t_1)}} y_e \geq d_\sigma \gamma^\sigma_{t,\tau} \qquad \forall \sigma \in \Sigma^a, \forall t \in T^a, \forall \tau \in \mathrm{T}^\sigma \tag{18}$$

$$\sum_{\tau \in \mathrm{T}^\sigma} \sum_{t \in T^a} \gamma^\sigma_{t,\tau} = 1 \qquad \forall \sigma \in \Sigma^a \tag{19}$$

$$\gamma^\sigma_{t,\tau} \in \{0,1\} \qquad \forall \sigma \in \Sigma^a, \forall t \in T^a, \forall \tau \in \mathrm{T}^\sigma \tag{20}$$

$$y_e \in \{0,1\} \qquad \forall e \in \hat{E} \tag{21}$$

It is evident that the total costs of the path are minimized by (16). The flow conservation constraints (17) are adjusted in order to include possibly more than one sink, as multiple gateway tracks are allowed in the TUSS problem. Flow could only 'escape' by using one of the departure gateway tracks (with corresponding time) in the set G^a. Set $G^a \subseteq G$ encompasses all relevant departure nodes for agent a, which could be smaller than the whole set G if the arrival time of agent a takes place after some departures occurred. The set $S^a \subseteq S$ consists of the arrival of agent a, while set S contains all arrivals. Thus $\mid S^a \mid = 1$, as the Pricing Problem is considered for all train units separately.

Constraint set (18) is utilized to ensure that the service tasks are satisfied and is explained in detail. A train unit that has to perform service σ should stay for at least d_σ consecutive time steps at a service track in the set T^σ. This is represented by the following constraint, for a fixed time step t, a fixed service σ and a fixed service site τ.

$$\sum_{e \in \delta_{in(\tau,t)}} y_e + \sum_{t_1=t}^{t+d_\sigma-2} \sum_{e \in \delta_{self(\tau,t_1)}} y_e \geq d_\sigma \tag{22}$$

After the train unit arrives at the service track τ at time t, it should wait at least $d_\sigma - 1$ time steps at the track in order to perform the service. Therefore, the summation over the waiting edges includes exactly $d_\sigma - 1$ time steps. The constraint should hold for all different services in set $\Sigma^a \subseteq \Sigma$. The train unit should be able to choose at which time step the service is performed, thus this constraint should hold for all time steps relevant for agent a in $T^a \subseteq T$. Lastly, service

completion should be possible at all service tracks $\tau \in \mathrm{T}^{\sigma}$.
In order to ensure that for all services the service is only performed once and not on all possible starting times and on all possible service tracks, the binary variables $\gamma^{\sigma}_{t,\tau}$ are introduced. If starting time t and service track τ are selected for service σ, then $\gamma^{\sigma}_{t,\tau} = 1$. This leads to the final version of the constraint:

$$\sum_{e \in \delta_{in(\tau,t)}} y_e + \sum_{t_1=t}^{t+d_\sigma-2} \sum_{e \in \delta_{self(\tau,t_1)}} y_e \geq d_\sigma \gamma^{\sigma}_{t,\tau} \qquad \forall \sigma \in \Sigma^a, \forall t \in T^a, \forall \tau \in \mathrm{T}^{\sigma} \tag{23}$$

If $\gamma^{\sigma}_{t,\tau} = 0$, then the constraint will always hold as the left hand side could not be smaller than 0. Each service has to be performed at least once, this is ensured by (19). The γ^{σ}_{t} should be binary and not fractional, otherwise services could split up into multiple shorter stays at service tracks. Given the requirements of the feasible paths in the previous section, this formulation is complete as no potentially feasible single train unit shunting plans are disregarded.
The importance of the number of time steps becomes evident in this MIP formulation. More time steps result in more constraints (18) and also in more nodes in V, thus more constraints (17). The set $\hat{E}$ also increases in size, leading to more y_e variables. Lastly, more $\gamma^{\sigma}_{t,\tau}$ variables are needed when more time steps are included.

6.4 MIP2 Formulation

Analyzing the graph infrastructure could change this formulation by reducing the number of variables and constraints. In the formulation above, constraint (18) ensures service completion by forming a constraint for all service tracks. By forming it for all service tracks separately, it is ensured that a train unit could not pick different tracks at the same time to perform one service. However, this could also be ensured by summing over all service sites as shown below.

$$\sum_{\tau \in \mathrm{T}^{\sigma}} \Big(\sum_{e \in \delta_{in(\tau,t)}} y_e + \sum_{t_1=t}^{t+d_\sigma-2} \sum_{e \in \delta_{self(\tau,t_1)}} y_e \Big) \geq d_\sigma \gamma^{\sigma}_{t} \qquad \forall \sigma \in \Sigma^a, \forall t \in T^a \tag{24}$$

The summation over all service sites enables the train unit to freely choose from the set of service tracks. However, it is not possible to perform the same service at different tracks due to the fact that at least 1 time step is needed to move to another service track. All tracks are at least 1 time step apart from each other. This movement between different service tracks is not possible as all subsequent time steps should be used to wait at the service site of arrival at time t in order to perform the service. Otherwise, the left side of constraint (24) could not equal d_σ. As the constraint is not formed for all service tracks, the binary decision variable γ^{σ}_{t} becomes only dependent on the time and service task. Constraint (19) that ensures that the service is performed

at least once, changes to the following:

$$\sum_{t \in T^a} \gamma_t^\sigma = 1 \qquad \forall \sigma \in \Sigma^a \tag{25}$$

Regarding computation time, it is expected to be preferred to sum over all possible service tracks instead of having one constraint for every service track and needing additional binary decision variables for all possible service tracks. However, this could alter the tightness of the MIP, as the LP-relaxation of this constraint is more general. In a LP-relaxation, different edges on multiple service tasks could be selected fractionally, while this is not possible in the previous formulation. In the previous formulation, only edges of one service task are selected in the LP-relaxation. This is the only difference between the two formulations, no feasible solutions are cut off. Thus, the newly formulated model is complete as well. This second MIP formulation, referred to as MIP2, is the following:

$$\min \sum_{e \in \hat{E}} c_e y_e \tag{26}$$

$$\text{s.t.} \sum_{\delta_{in(v)}} y_e - \sum_{\delta_{out(v)}} y_e = \begin{cases} 0 & \forall v \in V \setminus \{S^a, G^a\} \\ -1 & \forall v \in S^a \end{cases} \tag{27}$$

$$\sum_{\tau \in \mathrm{T}^\sigma} \Big(\sum_{e \in \delta_{in(\tau,t)}} y_e + \sum_{t_1 = t}^{t+d_\sigma - 2} \sum_{e \in \delta_{self(\tau,t_1)}} y_e \Big) \geq d_\sigma \gamma_t^\sigma \qquad \forall \sigma \in \Sigma^a, \forall t \in T^a \tag{28}$$

$$\sum_{t \in T^a} \gamma_t^\sigma = 1 \qquad \forall \sigma \in \Sigma^a \tag{29}$$

$$\gamma_t^\sigma \in \{0,1\} \qquad \forall \sigma \in \Sigma^a, \forall t \in T^a \tag{30}$$

$$y_e \in \{0,1\} \qquad \forall e \in \hat{E} \tag{31}$$

In order to decide which formulation is preferred in which situation, MIP1 and MIP2 are tested on various scenarios in the next section.

7 Results

In this section MIP1 and MIP2 are compared purely as Pricing Problems, outside the overlapping scope of Branch-and-Cut-and-Price. Furthermore, the manner gateway tracks are modelled is tested by implementing a gateway extension. Lastly, more insight is provided in how adjusting reduced costs could steer a generated path in certain directions.

Up to my knowledge, no complete methods exists yet that determine single train unit shunting plans with services with durations, thus the methods proposed in this study could not be benchmarked with other algorithms. Therefore, the different formulations are compared with each other. The comparison is on computational results, as MIP1 and MIP2 always have the same objective value. The results are obtained on a computer with an Intel (R) Core (TM) i7-4710MQ CPU @ 2.50GHz (4 CPUs) processor and memory of 8GB RAM. In C++, CPLEX version 12.10.0 was used.

The instance on which the majority of the tests is performed, corresponds with the shunting yard depicted in figure 11 and is called the standard instance for this section. The standard instance contains 2 services and 3 service tracks. On nodes 0 and 1, service 0 could be performed. Service 1 could be performed on node 3. The train unit, thus the agent, should have service 0 and 1 performed on its generated path. It starts at time 0 at gateway 11. After completing the route, it could choose to depart from one of the three gateways at the last time step.

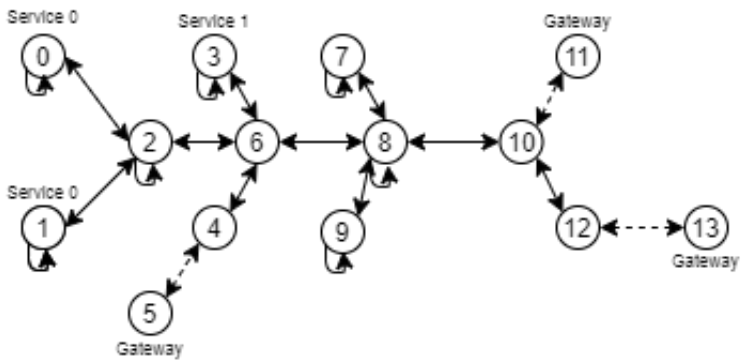

Figure 11: The standard shunting yard

In order to test various settings and infrastructure decisions, multiple changes are made to this shunting yard in the next sections. The standard instance consists of 14 nodes, where nodes 5, 11 and 13 are used as gateway. It is an artificially created instance for testing purposes and smaller than most shunting yards in the Netherlands. Additional costs for reverse movements are not taken into account for this standard setting. Costs for the edges are randomly picked between 1 and 10, with a fixed seed such that the research can be validated. Edges where movements in both directions are possible are represented by two distinct edges. Furthermore, waiting edges are included. This leads to a total of 33 edges for the standard instance. For each minute a time step is considered in an interval of 8 hours, resulting in 480 time steps in the standard setting. This gives 6720 different nodes and 15840 edges, as nodes and edges on different times are distinct.

The train unit that is used in the instances is assumed to have a length of 60 meters. This is a realistic length for the shortest train units, which consists of 3 segments of around 20 meters. As a modeling choice, explained in Section 3.1.5, all tracks are split up in nodes with a length of the shortest train unit, thus 60 meters. Including the starting up and braking speeds, 10 m/s is a realistic maximal average speed on shunting yards for train units. Thus, the time the smallest movement takes minimally, which is the traversal of one node, equals 6 seconds. Therefore, as substantiated in Section 4.2, this is a preferred time step size. As the maximal speed is used in this calculation, the method is complete. Time steps of 6 seconds corresponds with 4800 time steps in a period of 8 hours. Also this number of time steps is tested. However, the time step size could be adjusted, therefore scaling the time step size is tested computationally. Above calculations are made for the sake of testing more realistically, all parameters can be changed easily. Furthermore, fewer time steps are needed if trains are less than 8 hours on the shunting yard.
The routes computed by both formulations are validated on feasibility and completeness. This is done by examining the conservation of flow, the usage of (multiple) gateways, the performance of services and the total costs for different agents, services and instances. An extension to the feasibility study is testing how the routes are formed for different scenarios of reduced costs. This is described in Section 7.1. All tests lead to the conclusion that both formulations are valid and could be used by the NS.

7.1 Test Reduced Costs scenarios

As an extension to the feasibility study, the influence of the reduced costs are examined. Based on the dual variables, the reduced costs could steer the train unit in certain directions. In a setting with multiple trains, certain nodes could be visited more often than others. At the time that these nodes are more frequently visited, usually the reduced costs will increase as it is less likely that non-conflicting paths include these nodes. This steers the train unit to choose a path where other nodes are used at that time step. This behaviour attempts to reduce conflicts between the multiple trains on a yard.
This is tested for the standard instance and MIP2 by considering the gateways used by the train unit. The train unit can utilise gateway nodes 5, 11 and 13. In a setting where all integer reduced costs are generated by a uniform distribution between 1 and 10, gateway 5 is used for 42% of 100 sample runs. However, if we consider a scenario where node 4 is more frequently used by other trains; this percentage could change. Frequent usage of node 4, could be represented by increasing the reduced costs of all edges towards node 4. Thus, usage of gateway 5 is expected to decrease, as it could only be reached via node 4.
In the standard setting, the dual costs of all edges are drawn from a uniform distribution, $U(a, b)$ with $a = 1$ and $b = 10$. Gradually, the reduced costs of edges towards node 4 are increased. This is done by increasing a and b by 1 each step. The usage of the three gateways for the gradually increasing costs is depicted by Figure 12.

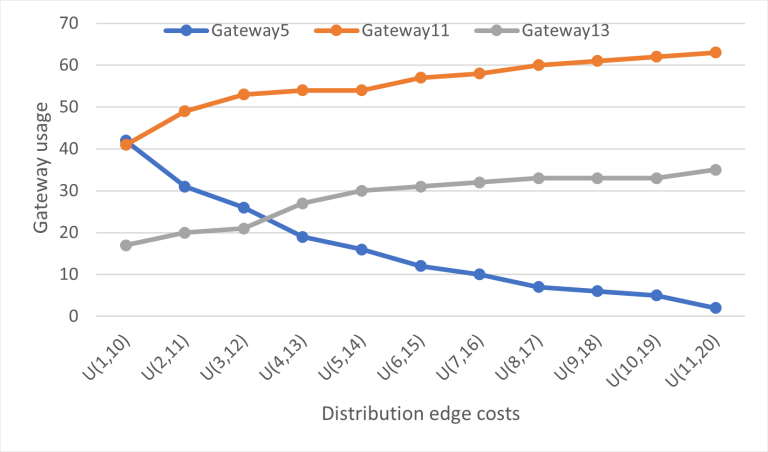

Figure 12: Gateway usage frequency for increasing edge costs towards node 4

It can be seen, that the usage of gateway 5 decreases as the expected reduced costs of edges that go towards node 4 increase. Naturally, gateways 11 and 13 are used more often. The first explanation is obviously that the direct path towards gateway 5 has higher costs, thus more often other gateways are used. Secondly, the train unit is less likely to use all edges towards node 4; thus it will be more likely that at the final stages it is located at another place on the shunting yard. The behaviour seen in Figure 12 shows that the reduced costs influence the path of the train units in the correct way.

7.2 Different MIP formulations

The MIP formulation as described in Section 6.3 and in Section 6.4 are denoted as respectively MIP1 and MIP2. As described in Section 6.4, MIP2 consists of fewer variables and constraints compared to MIP1. However, the LP-relaxation of the second formulation is weaker. For the standard instance, MIP1 contains 17280 variables and 8173 constraints. The second formulation consists of fewer: 16800 variables and 7693 constraints.

The difference in computational results between the two formulations for the Shortest Path with Waypoints with durations problem are tested for various numbers of time steps, numbers of nodes, different infrastructure decisions, various instances and service scenarios. The computation time encompasses constructing the CPLEX model and solving it. Reading in the shunting yard and constructing the corresponding graph is not included in the computation time, in the BCP framework this could be done once and is not needed for all Pricing Problems individually. The wall time is used as time measure, as a single computer is used to obtain the results and the wall time is easily interpretable.

In order to compare the formulations, the two-tailed t-test is used. As no assumptions on equal variances are made, the Welch's t-test for unequal variances is used with a significance level of 0.05. The outcomes should provide insight for the NS in what scenarios which MIP should be used to determine the shunting plans.

7.2.1 Test number of time steps

Both MIP formulations are tested for 100 distinct seeds on 10 different numbers of time steps, ranging in between 480 and 4800. Where the latter corresponds with a time step size of 6 seconds for a total period of 8 hours. This leads to 67200 nodes and 158400 edges. Figure 13 below depicts the computation times for both formulations.

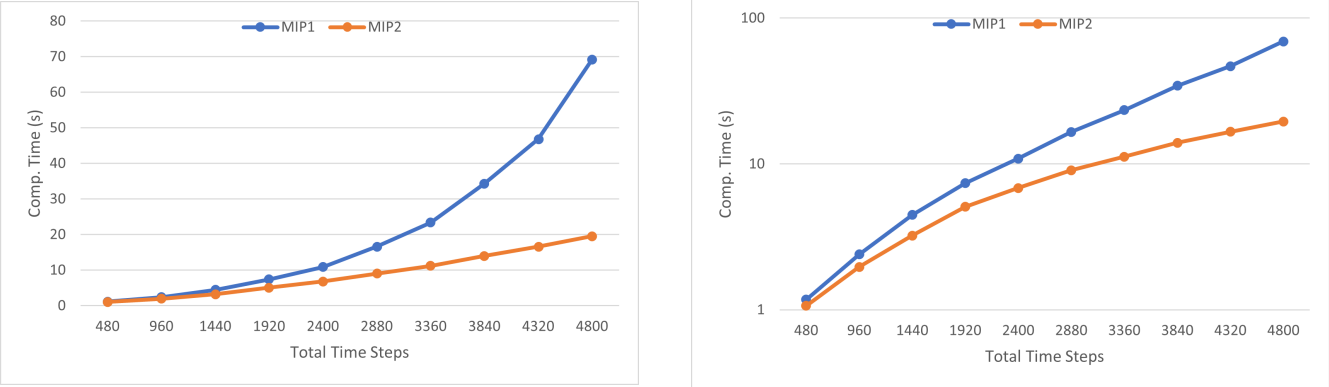

Figure 13: Computation Times of the two models for varying numbers of time steps. Right: Logarithmic scale.

It can be seen in Figure 13 that for all numbers of time steps, the average computation times of the second formulation are smaller. Determining a route for 480 time steps (6720 nodes and 15840 edges) takes MIP1 and MIP2 both slightly more than 1 second, respectively 1.2 and 1.1 seconds. For a significance level of 0.05, there is no significant difference between the computation times of the methods on 480 time steps as the p-value of the Welch's t-test is 0.126. For the larger number of time steps, all p-values are smaller than $8.1 \cdot 10^{-21}$, thus a statistical difference is confirmed. The statistical tests outcomes could be found in Appendix C.1.
For 4800 time steps, and thus 10 times more nodes and edges, the computation times are respectively 69.2 and 19.5 seconds. Thus, for the largest number of time steps, the second model reduces the computation time by 72 percent for the standard instance. The figure on the right, with the logarithmic y-axis, shows a decreasing slope for both MIPs for the first time steps. From 1920 time steps, a linear trend occurs for the first formulation, thus indicating an exponential growth of computation time. A linear trend, with a smaller slope, seems to occur for the larger numbers of time steps for the first formulation as well. A logistic transformation was made in order to perform a linear regression, this is explained in Appendix D. The linear trend on the logarithmic scale is underlined by an R^2 of 0.97 for the first MIP. 93% of the variance is declared by the linear regression model of the second MIP as an R^2 of 0.93 was found.

7.2.2 Test number of nodes

The standard instance is extended by adding multiple nodes. The additional nodes are added consecutively in between the node pairs 0-2, 1-2, 3-6, 7-8 and 9-8. A total of 10 cases are considered,

thus a maximum of two additional nodes are placed between the different node pairs. Therefore, various graphs with more than 14 nodes are obtained, such that the influence of the number of nodes on the computation time could be tested. The computational results are shown in Figure 14 below. Here, the average computation times of 100 runs are depicted in seconds.

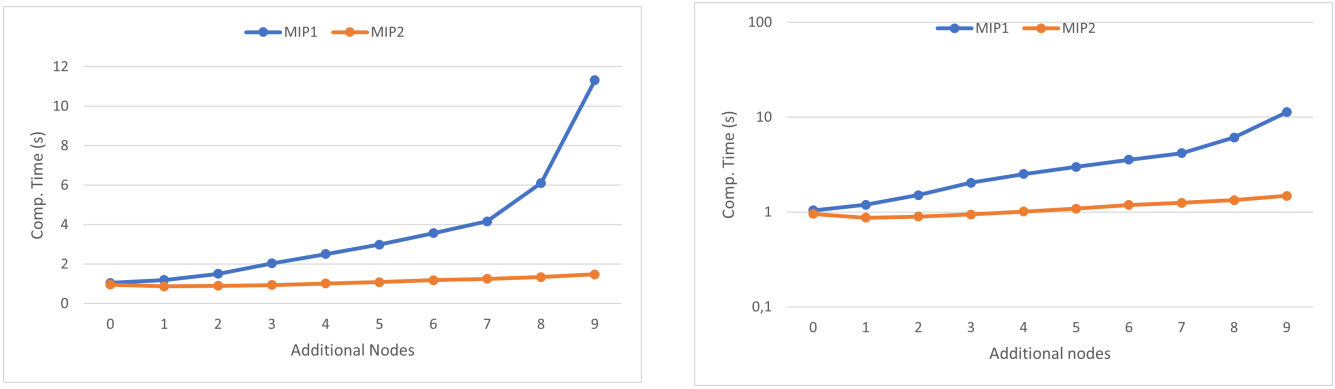

Figure 14: Computation Times of the two models for varying numbers of nodes. Right: Logarithmic scale.

One should note that with each additional node three additional edges are constructed. The total number of extra edges has to be multiplied with the number of time steps (480), thus 1440 edges are added for each additional node. The left part of Figure 14 suggests an exponential growth for the first MIP. This is confirmed by the linear trend of the model when a logarithmic scale is used. The second model could be at the start of an exponantial growth of computation time, its linear trend has a smaller slope compared to the first model. Linear regressions of the logistic transformation result in R^2 values of 0.50 and 0.53, the analysis could be found in Appendix D.
The computation times of both models seem to be comparable when no nodes are added. However the Welch's t-value of 0.005 shows significant statistical difference, the second model is faster. As more nodes are added, the difference between the performance of the models become more significant with all p-values below $1.4 \cdot 10^{-18}$. The statistical outcomes could be found in Appendix C.2. For the largest number of additional nodes, the computation time is reduced by 85 percent by the second model. In order to investigate the influence of nodes and edges further, the two formulations are tested on various instances.

7.2.3 Test different instances

The 5 different instances of shunting yards, on which the models are tested, are represented visually by the graphs in Appendix B. The standard instance is not included in this appendix, as it is depicted in Figure 11. Some of the instances are of the "carroussel" type, this does not force LIFO behaviour. Some contain long service corridors or contain multiple gateways. The latter enables train units to determine the preferred departure location between multiple locations. Instance 5 is based on a real shunting yard located in Heerlen in the south of the Netherlands.
Table 4 shows the computational results of both formulations tested on the various instances.

The results in the left table are based on instances with 480 time steps, while the table on the right represents the 5 instances on 4800 time steps. All computation times are averages of 100 randomized runs with different seeds. Based on the outcomes, the p-values of the Welch's t-test are obtained.

Inst.	**Computation Times (s)**		
	MIP1	MIP2	p-value
1	1.07	0.99	0.008
2	0.59	0.56	0.090
3	4.19	2.72	$2.39 \cdot 10^{-5}$
4	4.23	3.12	$5.60 \cdot 10^{-9}$
5	5.49	5.12	$5.00 \cdot 10^{-4}$

Inst.	**Computation Times (s)**		
	MIP1	MIP2	p-value
1	23.23	20.59	$7.68 \cdot 10^{-6}$
2	10.95	9.78	$6.63 \cdot 10^{-8}$
3	73.63	52.90	$2.42 \cdot 10^{-40}$
4	76.97	59.69	$6.22 \cdot 10^{-24}$
5	155.04	137.77	$5.87 \cdot 10^{-12}$

Table 4: Computation times on 5 different instances, time steps: left 480 and right 4800.

Based on the table above, it could be concluded that MIP2 is significantly faster than MIP1 on 4 out of 5 instances for 480 time steps. Only instance 2 has a difference in computation time that is not significant. For 4800 time steps, all instances are solved significantly faster by MIP2.

7.2.4 Test service scenarios

Different scenarios are created in order to test the influence of the services on the computational performance of the models. After starting with a scenario with no services and service tracks, different services are added with corresponding service tracks. The scenarios which are used for testing can be found in Table 5.

Scenario	**Description**
1	No services and no service tracks
2	Only service 0 on service tracks 0 and 1
3	Standard instance
4	Add service 2 on service tracks 2 and 7 to standard instance
5	Add service 3 on service tracks 8 and 9 to scenario 4
6	Services 0, 1, 2, 3, 4, 5 and 6 on respectively nodes 0, 1, 2, 3, 7, 8 and 9

Table 5: Scenarios used for service tests

The computational results of 100 runs for both models on all instances can be found in Tables 6 and 7. Besides the means and standard deviations in seconds, the p-values for the Welch's t-test are obtained. For scenarios 2, 3 and 5 the second MIP formulation has a lower average computation time, these differences are significant. There is no significant statistical difference for scenario 1, which is intuitive as the models should behave in the same manner as no services are included. However, it is a surprising result that the average computation time is higher for scenario 1 than for scenarios 2, 3 and 4. It is surprising due to the fact that in the latter three (multiple) services are included. The amount of possible single train unit routes has increased due to the absence of

service requirements, this could possibly have increased computation times.
Furthermore, in scenarios 4 and 6 no significant difference is found. The first MIP formulation contains more variables and constraints if and only if at least one service could be performed at multiple tracks. This is not the case in scenarios 1 and 6, which explains that there is no difference in performance found between the models. The influence of multiple service tracks for one particular service is further tested in the next section. In conclusion, MIP2 performs significantly better for 50% of the scenarios. For the other instances, the null hypothesis of similar computation times cannot be rejected.

	Scenario 1		**Scenario 2**		**Scenario 3**	
Comp. Time (s)	Average	St. dev.	Average	St. dev.	Average	St. dev.
MIP1	1.20	0.212	0.79	0.11	1.01	0.17
MIP2	1.22	0.22	0.73	0.08	0.94	0.18
p-value	0.559		$1.87 \cdot 10^{-4}$		$8.46 \cdot 10^{-3}$	

Table 6: Statistical results for the service scenarios 1, 2 and 3

	Scenario 4		**Scenario 5**		**Scenario 6**	
Comp. Time (s)	Average	St. dev.	Average	St. dev.	Average	St. dev.
MIP1	1.18	0.27	1.48	0.38	2.10	0.93
MIP2	1.16	0.28	1.22	0.41	2.07	0.85
p-value	0.591		$9.02 \cdot 10^{-6}$		0.815	

Table 7: Statistical results for the service scenarios 4, 5 and 6

No logical reasons are found for no significant difference in performance between the two models for scenario 4. For 4800 time steps, MIP2 performs significantly better (p-value of 0.022) than MIP1 on the fourth instance. For the other scenarios, the same conclusions are derived based on 4800 time steps. The statistical outcomes can be found in Appendix C.3.

7.2.5 Test number of service tracks

Interesting insights in the behaviour of both models could be gained by varying the number of service tracks available for a single service. This is relevant as the MIP2 has a weaker LP-relaxation for services with multiple tracks. For the standard instance in Figure 11, nodes 2, 7, 8 and 9 are subsequently added as service tracks for service 1. This is possible as waiting is allowed on these nodes. Also a situation without a service track for service 1 is considered, obviously the agent is not required to perform the service in that scenario. The results are depicted in Figure 15 below. It could be seen that the performance starts to diverge when the instance consists of multiple possible service tracks for service 1. The first MIP seems to have a linear trend on a logarithmic scale, indicating an exponential relationship. The linear regression models of the logistic transformations declare respectively 87 and 32 percent of all variance in the data for both MIPs.

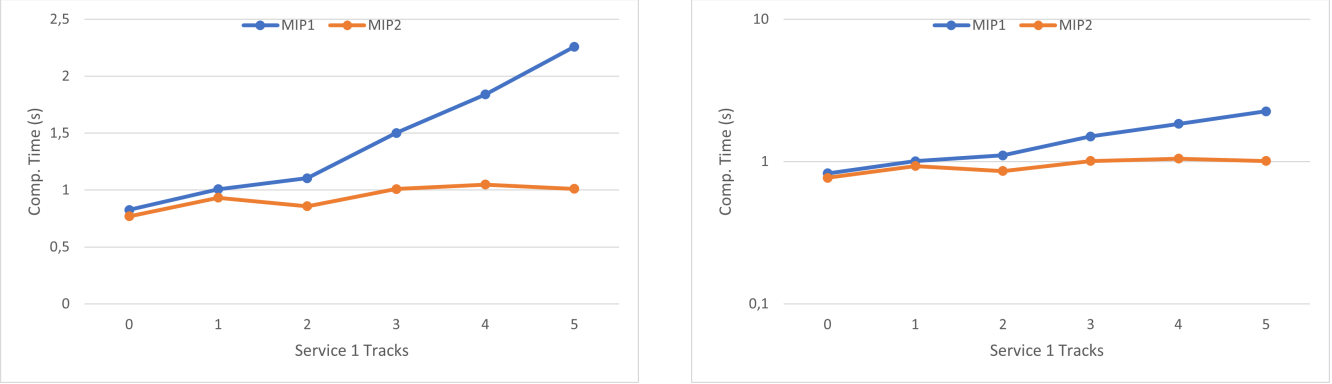

Figure 15: Computation Times of the two models for varying numbers of service 1 tracks. Right: Logarithmic scale.

Based on the Welch's t-test it could be concluded that adding more than 1 service track to service 1, leads to a significant statistical difference in the performance of the models with all p-values below $8.2 \cdot 10^{-30}$ in this case. In the scenario with no service tracks for service 1, thus only two tracks for service 0, MIP2 also performs significantly better. For the standard instance, thus one service track for service 1, no statistical differences are found. The statistical outcomes of the tests could be found in Appendix C.4. Using MIP2 for the largest number of tracks for service 1, leads to a reduction of 55 percent in computation time.

7.3 Test different infrastructure decisions

In order to make the current relaxation tighter, without losing completeness, the gateways are modelled differently than normal tracks. As described in Section 4.1.3, the gateways could be modelled differently by two extensions. In practice, train units could not traverse over gateways; this behaviour is prevented by both extensions. Therefore, the relaxation becomes tighter. As the second gateway extension consists of fewer nodes, it is preferred over the first one. The second gateway extension model is compared with the normal model on both computation time and objective values. Both models are relaxations that are complete, thus the highest objective value is preferred as this more closely represents the reality.

On 5 instances, the gateway extension is compared with the normal model. As MIP2 performs better, based on the previous subsection, the comparisons are made with this formulation. Thus, the gateway extension will be implemented in MIP2. As it is tighter, the gateway extension does not always have the same objective values as the normal model. Therefore, the objective values are compared as well. It is ensured that the randomly assigned reduced costs are equal for the same edges, such that the comparisons of objective values are fair.

The computation times and objective values for the standard instance of both models are depicted in Tables 8 and 9. In Appendix C.5, Tables 26 to 32 represent the same statistics for instances 2 to 5. In the tables, the percentage 'tighter' represents the percentage of sample runs for which the gateway extension obtains a higher objective value. Considering 480 time steps, it can be seen

that the gateway extension is significantly faster for instance 1. This also applies for 4800 time steps. For the latter number of time steps, all sample runs are tighter. 99% of the objective values are higher for 480 time steps.

Model	**Computation Times (s)**			**Objective Values**	
	Average	St. Dev	p-value	Average	Tighter
Normal	0.95	0.19		1060.25	
Extension	0.81	0.18	$1.36 \cdot 10^{-7}$	1071.65	99%

Table 8: Gateway extension for Instance 1, 480 time steps

Model	**Computation Times (s)**			**Objective Values**	
	Average	St. Dev	p-value	Average	Tighter
Normal	18.91	1.77		10470.23	
Extension	14.33	1.16	$1.05 \cdot 10^{-50}$	10580.59	100%

Table 9: Gateway extension for Instance 1, 4800 time steps

For instances 2, 3 and 4, there is no significant difference in computation times for 480 time steps. The gateway extension is significantly faster for instance 5. The fact that no statistical differences are found for instances 2 and 3 can be explained by the fact that these instances only contain 1 gateway. As there are less edges from and towards gateways, the gateway extension has less influence on the computation time. The fourth instance, with multiple gateways, is solved significantly faster by the gateway extension for 4800 time steps. For the other instances, the conclusions of the Welch's t-test do not change for this number of time steps.

Regarding the objective values, one can conclude that the gateway extension tightens the formulation further. For at least 99 of the 100 sample runs (with randomized reduced costs), a higher objective value was found with the extension on instances 1, 2 and 4. For the remaining sample run, the objective values are equal as the gateway edges were not used in the optimal solution without the extension. The objective value cannot become lower when using the extension. Due to the fact that the formulation becomes more realistic and more strict, the objective values are higher on average for all instances on both number of time steps. The formulation becomes more strict, as a large number of gateway edges could not be used in the extension. Thus, fewer paths are possible to be formed. For 4800 time steps, 100% of all sample runs are tighter for all instances.

In conclusion, for all instances the formulations becomes tighter by applying the gateway extension. This better representation of reality does not come at a cost of computation time. For some instances the extension is significantly faster, while for others no statistical difference is obtained.

8 Conclusion

The logistic capacity determination of shunting yards is an important issue for the NS, as strategic investment decisions are made based on the current capacity. In this determination, feasible shunting plans are generated. Whether a complete shunting path generating method could be developed within the BCP framework, including the service tasks, is the main question of this thesis.

In the thesis, first a translation is made between the Multi-Agent Pathfinding problem and the Train Unit Shunting and Service problem. Train units should be considered as agents and a time-expanded graph should be used. The most important constraints are the vertex and edge conflicts, additional departure constraints can be added. It is derived that adding waypoints is necessary in order to include service for the train units. The waypoints should not be only visited, but the train units should stay at the waypoints for at least the service duration. It can be concluded that completeness can be guaranteed for this translation and this answers the first sub-question.

The formulated MAPFW can be solved by the Branch-and-Cut-and-Price framework. Here, not all constraints and variables are included initially, but are added iteratively with the goal to decrease computation time. Corridor conflicts, extended to hold for all approaching train units, could be added to strengthen the formulation and speed up the algorithm. Conflicts between the multiple train units traversing over the yard, are removed by the Separator Module. This concludes the second sub-question as the framework is complete and a relaxation of the real problem.

The third and fourth sub-question focus on the Pricing Problem within the BCP framework. Here, single train unit paths are generated. The reduced costs in the objective function steer the generated path based on the dual costs. Most dual costs are expressed in edge usage. These can be updated iteratively based on the current multi-agent situation on the yard. As in practice, the train unit can select any feasible departure and is able to traverse over unused service tracks. Parking on certain tracks is possible, which is also needed to perform all services. Currently, no models are known that form shunting paths including waypoints with durations. Two complete MIP formulations are formed, which differ in the way service completion is ensured by the models. The first has a stronger LP-relaxation, while the second has fewer variables and constraints.

From the computational results, it becomes quite clear that the second MIP formulation is preferred. At worst, no statistical difference is found with the first model. However, in various settings the computation time is significantly less, while the objective values are equal. Percentual reductions of computation time of up to 85 percent were found, which are only expected to increase with increasing instance sizes. Moreover, it is concluded that the model could be further improved by modelling the gateways differently. This extension makes the model more realistic by being more restrictive. For instances with multiple gateways, this also leads to a significant computational performance improvement. In conclusion, it is advised to the NS to utilise the second MIP formulation including the gateway extension as a Pricing Problem in the Branch-and-Cut-and-Price framework. It outperforms the other models and is a complete relaxation of the TUSS.

8.1 Further Research

Future research could focus on further reducing computation times. Both for the Pricing Problem on its own, as for the overall Branch-and-Cut-and-Price framework. The latter could possibly be improved by implementing suggestions made in this thesis such as using the BCP framework with more valid inequalities. Performance of the corridor conflict extension as developed in this thesis could be tested in the overall framework. Including these extended corridors is expected to be very beneficial, as many corridors are formed on the shunting yards. In order to reduce computation times of the overall framework, the BCP method could be stopped at the first moment a feasible solution is found. This is enough to prove feasibility, which was the purpose of the logistic capacity determination process.

The model could be extended by not only considering single train units. Trains, thus compositions of various train units are used on the shunting yard in practice. Incorporating this into the model would more closely describe reality. Furthermore, it would be interesting to analyse the performance of the developed MIP within the BCP framework and analyse the interaction. Also, the discretisation of time could be further investigated. Using discrete time steps leads to an approach that is 'complete in practice' according to the NS. Preferably, a method is developed that is also 'complete in theory' such that it could be applied to all kind of problems where completeness is required.

Currently, the aim is to incorporate the second MIP formulation in the BCP framework such that the logistic capacity of shunting yards can be determined in a complete manner. This is done by analysing instances with increasing amounts of trains. Determining the capacity of the majority of the instances could be done with the faster Machine Learning (ML) models. However, for 'critical' instances where these models fail to find feasible shunting plans, the complete BCP framework could be used. The speed of the ML is utilised, while it is ensured that all feasible solutions are found by the completeness of the BCP approach. More research could be performed on this promising combination of ML and Operation Research techniques like BCP.

References

Amiri, S. A., Foerster, K.-T., and Schmid, S. (2020). Walking through waypoints. *Algorithmica*, 82:1784–1812.

de Andrade, R. (2013). Elementary shortest-paths visiting a given set of nodes. *Simpósio Brasileiro de Pesquisa Operacional*, pages 16–19.

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

Geiger, M. J., Huber, S., Langton, S., Leschik, M., Lindorf, C., and Tüshaus, U. (2018). Multi-attribute assignment of trains to departures in rolling stock management. *Annals of Operations Research*, 271(2):1131–1163.

Gomes, T., Martins, L., Ferreira, S., Pascoal, M., and Tipper, D. (2017). Algorithms for determining a node-disjoint path pair visiting specified nodes. *Optical Switching and Networking*, 23:189–204.

Gurevich, Y. and Shelah, S. (1987). Expected computation time for hamiltonian path problem. *SIAM Journal on Computing*, 16(3):486–502.

Haahr, J. T., Lusby, R. M., and Wagenaar, J. C. (2017). Optimization methods for the train unit shunting problem. *European Journal of Operational Research*, 262(3):981–995.

Haijema, R., Duin, C., and Van Dijk, N. M. (2006). Train shunting: A practical heuristic inspired by dynamic programming. *Planning in Intelligent Systems: Aspects, Motivations, and Methods*, pages 437–475.

Huisman, D., Kroon, L. G., Lentink, R. M., and Vromans, M. J. (2005). Operations research in passenger railway transportation. *Statistica Neerlandica*, 59(4):467–497.

Jacobsen, P. M. and Pisinger, D. (2011). Train shunting at a workshop area. *Flexible Services and Manufacturing Journal*, 23(2):156–180.

Kroon, L. G., Lentink, R. M., and Schrijver, A. (2008). Shunting of passenger train units: an integrated approach. *Transportation Science*, 42(4):436–449.

Lam, E. and Le Bodic, P. (2020). New valid inequalities in branch-and-cut-and-price for multi-agent path finding. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 184–192.

Lam, E., Le Bodic, P., Harabor, D. D., and Stuckey, P. J. (2019). Branch-and-cut-and-price for multi-agent pathfinding. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 28, pages 1289–1296.

Lawler, E. L. (1985). The traveling salesman problem: a guided tour of combinatorial optimization. *Wiley-Interscience Series in Discrete Mathematics*.

Lentink, R. (2006). *Algorithmic decision support for shunt planning*. PhD thesis, Erasmus University Rotterdam.

Lentink, R. M., Fioole, P.-J., Kroon, L. G., and Van't Woudt, C. (2006). Applying operations research techniques to planning of train shunting. *Planning in Intelligent Systems: Aspects, Motivations, and Methods*, pages 415–436.

Lusby, R. M., Larsen, J., Ehrgott, M., and Ryan, D. (2011). Railway track allocation: models and methods. *OR Spectrum*, 33(4):843–883.

Ma, H. and Koenig, S. (2016). Optimal target assignment and path finding for teams of agents. *arXiv preprint arXiv:1612.05693*.

Matai, R., Singh, S. P., and Mittal, M. L. (2010). Traveling salesman problem: an overview of applications, formulations, and solution approaches. In *Traveling salesman problem, theory and applications*, volume 1, chapter 1, pages 1–24. InTech, Croatia.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329.

Mitten, L. (1970). Branch-and-bound methods: General formulation and properties. *Operations Research*, 18(1):24–34.

Mulderij, J., Huisman, B., Tönissen, D., van der Linden, K., and de Weerdt, M. (2020). Train unit shunting and servicing: a real-life application of multi-agent path finding. *arXiv preprint arXiv:2006.10422*.

Pataki, G. (2003). Teaching integer programming formulations using the traveling salesman problem. *SIAM Review*, 45(1):116–123.

Peeters, M. and Kroon, L. (2008). Circulation of railway rolling stock: a branch-and-price approach. *Computers & Operations Research*, 35(2):538–556.

Sharon, G., Stern, R., Felner, A., and Sturtevant, N. R. (2015). Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66.

Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., Li, J., Atzmon, D., Cohen, L., Kumar, T., et al. (2019). Multi-agent pathfinding: Definitions, variants, and benchmarks. *arXiv preprint arXiv:1906.08291*.

van de Ven, A., Zhang, Y., Lee, W.-J., Eshuis, R., and Wilbik, A. (2019). Determining capacity of shunting yards by combining graph classification with local search. In *International Conference on Agents and Artificial Intelligence*, volume 11, pages 285–293.

van den Akker, J., Baarsma, H., Hurink, J., Modelski, M. S., Paulus, J., Reijnen, I., Roozemond, D. A., and Schreuder, J. (2008). Shunting passenger trains: getting ready for departure. *CWI Syllabus*, 63(1):1–19.

van den Broek, R. W. (2016). Train shunting and service scheduling: an integrated local search approach. Master's thesis.

A Example Timetable Instances

Activity	Train ID	Type	Length	Time	Tasks
Arrival	1	a	80	11:30	Service
Arrival	2	a	80	11:45	None
Departure		a		12:15	
Arrival	3	a	80	12:30	Service
Departure		a		15:30	
Departure		a		16:00	

Table 10: Example of Instance 1

The correct label of instance 1 is 'Feasible', a feasible shunting plan is constructed in Table 2.

Activity	Train ID	Type	Length	Time	Tasks
Arrival	1	a	80	11:30	Service
Arrival	2	a	80	11:45	None
Arrival	3	a	80	12:15	None
Arrival	4	a	80	12:30	None
Arrival	5	a	80	12:45	Service
Arrival	6	a	80	13:00	None
Departure		a		13:30	
Departure		a		15:00	
Departure		a		15:30	
Departure		a		16:00	
Departure		a		16:15	
Departure		a		16:00	

Table 11: Example of Instance 2

The correct label of instance 2 is 'Infeasible' as the total train unit length exceeds the total track length at the shunting yard.

Activity	Train ID	Type	Length	Time	Tasks
Arrival	1	a	80	11:30	Service
Arrival	2	a	80	11:35	Service
Departure		a		11:55	
Departure		a		12:05	

Table 12: Example of Instance 3

The correct label of instance 3 is 'Infeasible' as there is not enough time to perform both service tasks for the train units before the departures.

Activity	Train ID	Type	Length	Time	Tasks
Arrival	1	a	80	11:30	Service
Departure		a		12:30	
Arrival	2	a	80	12:55	None
Departure		a		14:00	

Table 13: Example of Instance 4

The correct label of instance 4 is 'Feasible' as at most 1 train unit at a time is present at the shunting yard and there is enough time to perform the service task.

Activity	Train ID	Type	Length	Time	Tasks
Arrival	1	a	80	11:30	Service
Arrival	2	a	80	11:45	None
Departure		a		13:00	
Departure		a		13:00	

Table 14: Example of Instance 5

The correct label of instance 5 is 'Infeasible' as two departures are scheduled at the same time while only one gateway track is present at the yard.

B Shunting yards

B.1 Instance 2

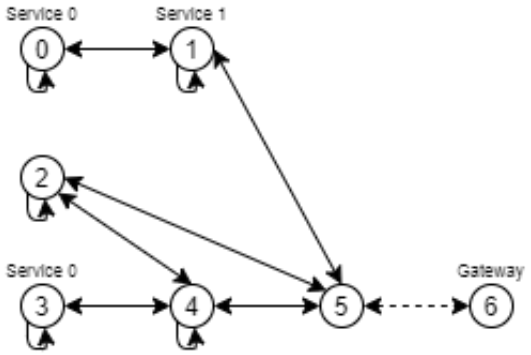

Figure 16: Shunting yard of instance 2

This shunting yard corresponds with the shunting yard in Figure 4. In this instance, the agent starts at gateway track 6 at begin time 0.

B.2 Instance 3

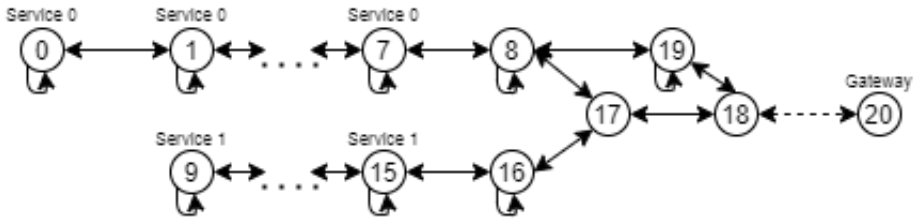

Figure 17: Shunting yard of instance 3

This shunting yard contains long service corridors between nodes 0 and 8 and between nodes 9 and 16. The agent starts at gateway track 20 at time 0.

B.3 Instance 4

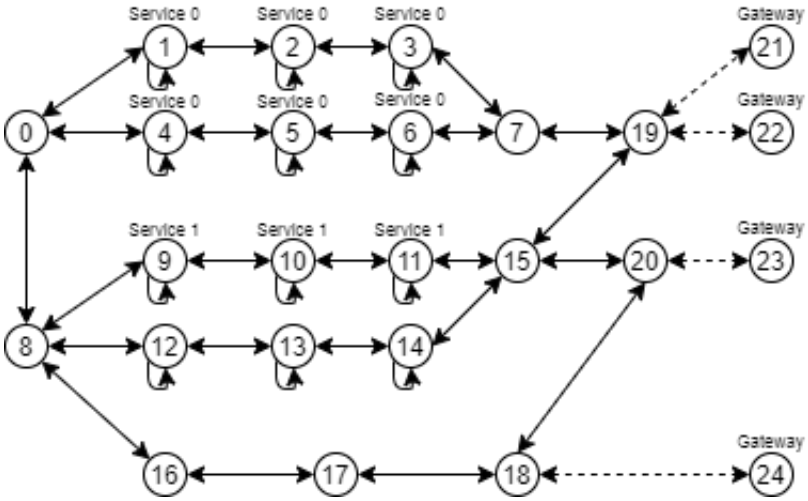

Figure 18: Shunting yard of instance 4

This shunting yard is of the "carroussel" type, which means that almost all tracks are accessible from both sides, therefore LIFO behaviour is not enforced for the train units. Furthermore, multiple corridors are available for performing services. The yard contains 4 gateways, the agent starts from gateway track 21 at time 0.

B.4 Instance 5

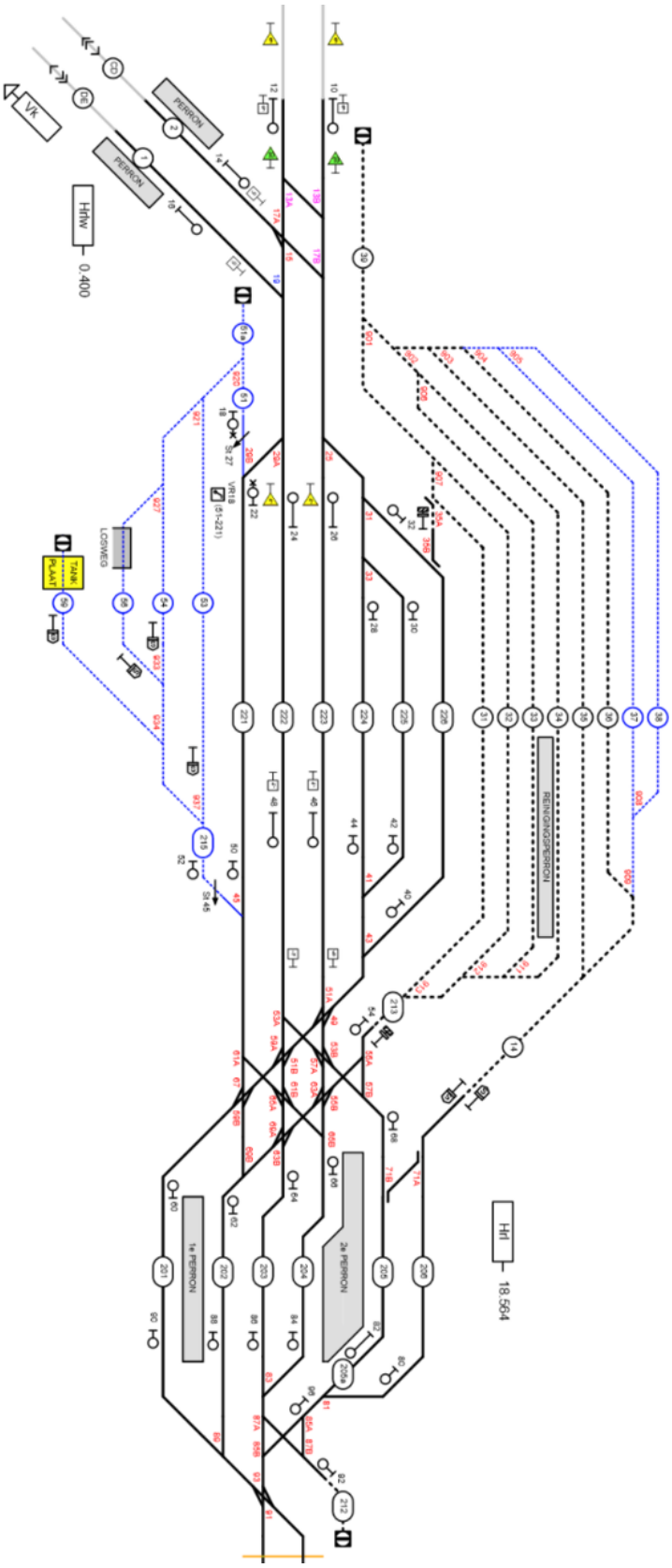

Figure 19: Shunting yard of instance 5

Instance 5 is based on the real NS shunting yard located in Heerlen, the source of this image is sporenplan.nl. This yard has long corridors, multiple gateways and multiple service tracks. According to the NS this yard is also of the "carroussel" type.

C Statistical Output

C.1 Number of time steps

	Time steps: 480		**Time steps: 960**		**Time steps: 1440**	
Comp. Time (s)	Average	St. dev.	Average	St. dev.	Average	St. dev.
MIP1	1.18	0.58	2.40	0.32	4.47	0.50
MIP2	1.07	0.41	1.97	0.25	3.23	0.46
p-value	0.126		$8.03 \cdot 10^{-21}$		$5.39 \cdot 10^{-44}$	

Table 15: Statistical results for time steps: 480 - 1440

	Time steps: 1920		**Time steps: 2400**		**Time steps: 2880**	
Comp. Time (s)	Average	St. dev.	Average	St. dev.	Average	St. dev.
MIP1	7.39	0.99	10.86	1.05	16.56	1.89
MIP2	5.08	0.62	6.83	0.77	9.05	0.53
p-value	$2.19 \cdot 10^{-45}$		$2.27 \cdot 10^{-74}$		$3.32 \cdot 10^{-67}$	

Table 16: Statistical results for time steps: 1920 - 2880

	Time steps: 3360		**Time steps: 3840**		**Time steps: 4320**	
Comp. Time (s)	Average	St. dev.	Average	St. dev.	Average	St. dev.
MIP1	23.36	1.99	34.32	3.66	46.81	3.74
MIP2	11.22	1.14	13.98	1.05	16.61	1.48
p-value	$1.86 \cdot 10^{-102}$		$3.91 \cdot 10^{-83}$		$1.41 \cdot 10^{-108}$	

Table 17: Statistical results for time steps: 3360 - 4320

	Time steps: 4800	
Comp. Time (s)	Average	St. dev.
MIP1	69.15	8.60
MIP2	19.53	2.36
p-value	$2.23 \cdot 10^{-84}$	

Table 18: Statistical results for time steps: 4800

C.2 Number of nodes

	Additional nodes: 0		**Additional nodes: 1**		**Additional nodes: 2**	
Comp. Time (s)	Average	St. dev.	Average	St. dev.	Average	St. dev.
MIP1	1.08	0.22	1.44	0.33	1.80	0.45
MIP2	0.99	0.21	1.04	0.16	1.06	0.12
p-value		0.005		$3.17 \cdot 10^{-21}$		$1.95 \cdot 10^{-30}$

Table 19: Statistical results for additional nodes: 0 - 2

	Additional nodes: 3		**Additional nodes: 4**		**Additional nodes: 5**	
Comp. Time (s)	Average	St. dev.	Average	St. dev.	Average	St. dev.
MIP1	2.23	0.53	2.62	0.83	3.44	1.37
MIP2	1.22	0.14	1.20	0.14	1.43	0.31
p-value		$1.12 \cdot 10^{-35}$		$2.40 \cdot 10^{-31}$		$6.91 \cdot 10^{-27}$

Table 20: Statistical results for additional nodes: 3 - 5

	Additional nodes: 6	
Comp. Time (s)	Average	St. dev.
MIP1	4.04	2.41
MIP2	1.42	0.17
p-value		$1.37 \cdot 10^{-18}$

Table 21: Statistical results for additional node: 6

C.3 Service scenarios

	Scenario 1		**Scenario 2**		**Scenario 3**	
Comp. Time (s)	Average	St. dev.	Average	St. dev.	Average	St. dev.
MIP1	214.72	9.60	17.77	1.85	21.23	3.35
MIP2	215.77	10.17	17.05	1.74	18.92	2.12
p-value	0.457		0.005		$2.91 \cdot 10^{-8}$	

Table 22: Statistical results for the service scenarios 1, 2 and 3 for 4800 time steps

	Scenario 4		**Scenario 5**		**Scenario 6**	
Comp. Time (s)	Average	St. dev.	Average	St. dev.	Average	St. dev.
MIP1	25.10	3.98	29.33	5.44	35.35	7.66
MIP2	23.86	3.70	25.62	4.52	35.21	7.55
p-value	0.022		$4.24 \cdot 10^{-7}$		0.891	

Table 23: Statistical results for the service scenarios 4, 5 and 6 for 4800 time steps

C.4 Number of service tracks

	Additional tracks: 0		**Additional tracks: 1**		**Additional tracks: 2**	
Comp. Time (s)	Average	St. dev.	Average	St. dev.	Average	St. dev.
MIP1	0.83	0.12	1.01	0.23	1.10	0.15
MIP2	0.77	0.10	0.93	0.20	0.86	0.10
p-value	$3.00 \cdot 10^{-4}$		0.014		$8.16 \cdot 10^{-30}$	

Table 24: Statistical results for additional service tracks: 0 - 2

	Additional tracks: 3		**Additional tracks: 4**		**Additional tracks: 5**	
Comp. Time (s)	Average	St. dev.	Average	St. dev.	Average	St. dev.
MIP1	1.50	0.16	1.84	0.15	2.26	0.18
MIP2	1.01	0.11	1.05	0.09	1.01	0.10
p-value	$5.74 \cdot 10^{-62}$		$1.61 \cdot 10^{-94}$		$1.76 \cdot 10^{-109}$	

Table 25: Statistical results for additional service tracks: 3 - 5

C.5 Gateway extension

Model	Computation Times (s)			Objective Values	
	Average	St. Dev	p-value	Average	Tighter
Normal	0.67	0.22		1074.95	
Extension	0.64	0.18	0.230	1096.78	100%

Table 26: Gateway extension for Instance 2, 480 time steps

Model	Computation Times (s)			Objective Values	
	Average	St. Dev	p-value	Average	Tighter
Normal	9.80	1.36		10640.80	
Extension	9.64	1.40	0.431	10858.74	100%

Table 27: Gateway extension for Instance 2, 4800 time steps

Model	Computation Times (s)			Objective Values	
	Average	St. Dev	p-value	Average	Tighter
Normal	2.36	0.61		1032.77	
Extension	2.39	0.69	0.767	1038.29	92%

Table 28: Gateway extension for Instance 3, 480 time steps

Model	Computation Times (s)			Objective Values	
	Average	St. Dev	p-value	Average	Tighter
Normal	62.02	22.61		10167.90	
Extension	63.16	26.23	0.405	10209.23	100%

Table 29: Gateway extension for Instance 3, 4800 time steps

Model	Computation Times (s)			Objective Values	
	Average	St. Dev	p-value	Average	Tighter
Normal	2.74	1.00		1007.04	
Extension	2.67	0.60	0.567	1016.39	99%

Table 30: Gateway extension for Instance 4, 480 time steps

Model	Computation Times (s)			Objective Values	
	Average	St. Dev	p-value	Average	Tighter
Normal	59.97	8.56		9953.94	
Extension	53.04	6.69	$1.36 \cdot 10^{-9}$	10040.33	100%

Table 31: Gateway extension for Instance 4, 4800 time steps

Model	**Computation Times (s)**			**Objective Values**	
	Average	St. Dev	p-value	Average	Tighter
Normal	4.77	0.80		1019.86	
Extension	4.41	1.05	0.007	1022.56	69%

Table 32: Gateway extension for Instance 5, 480 time steps

Model	**Computation Times (s)**			**Objective Values**	
	Average	St. Dev	p-value	Average	Tighter
Normal	155.32	51.68		10037.33	
Extension	139.94	34.22	0.014	10052.94	100%

Table 33: Gateway extension for Instance 5, 4800 time steps

D Logistic Regression

Visually, the graphs indicate an exponential relation between the computation time and the independent variable. An exponential relation could be described by the following formula:

$$y = c \cdot e^{d \cdot x} \tag{32}$$

Applying a logistic transformation gives the following:

$$\ln y = \ln \left(c \cdot e^{d \cdot x} \right) = \ln c + d \cdot x \tag{33}$$

This can be seen as the standard formula of linear regression:

$$\phi = a + d \cdot x \tag{34}$$

Here, $\phi = \ln y$ and $a = \ln c$. Thus, applying a logistic transformation to the data enables testing for an exponential trend by using linear regression. This analysis is performed for both MIP1 and MIP2 on the number of time steps, number of nodes and number of service tracks. The y variable represents the computation time in seconds. The intercept is represented by a. The variable b represents respectively the number of time steps, additional nodes or service tracks.

D.1 Number of time steps

For MIP1, the following regression statistics are found.

Regression Statistics	
R Square	0.9687
Standard Error	0.2259
Observations	1000

	Coefficients	**Standard Error**	**t Stat**	**P-value**
Intercept	0.0351	0.0154	2.2732	0.0232
X variable	0.0009	$5.182 \cdot 10^{-6}$	175.6383	0.0000

For MIP2, the following regression statistics are found.

Regression Statistics	
R Square	0.9270
Standard Error	0.2520
Observations	1000

	Coefficients	Standard Error	t Stat	P-value
Intercept	0.1222	0.0172	7.0992	$2.379 \cdot 10^{-12}$
X variable	0.0007	$5.781 \cdot 10^{-6}$	112.5933	0.0000

D.2 Number of nodes

For MIP1, the following regression statistics are found.

Regression Statistics	
R Square	0.4968
Standard Error	2.0395
Observations	1000

	Coefficients	Standard Error	t Stat	P-value
Intercept	1.9510	0.1037	18.8141	$8.043 \cdot 10^{-68}$
X variable	3.9572	0.1261	31.3897	$5.269 \cdot 10^{-151}$

For MIP2, the following regression statistics are found.

Regression Statistics	
R Square	0.5271
Standard Error	1.9771
Observations	1000

	Coefficients	Standard Error	t Stat	P-value
Intercept	3.4334	0.0702	48.8915	$3.786 \cdot 10^{-267}$
X variable	8.9757	0.2691	33.3553	$1.702 \cdot 10^{-164}$

D.3 Number of service tracks

For MIP1, the following regression statistics are found.

Regression Statistics	
R Square	0.8684
Standard Error	0.6207
Observations	600

	Coefficients	Standard Error	t Stat	P-value
Intercept	1.3147	0.0316	41.6109	$1.003 \cdot 10^{-178}$
X variable	4.1984	0.0668	62.8063	$1.749 \cdot 10^{-265}$

For MIP2, the following regression statistics are found.

Regression Statistics	
R Square	0.3163
Standard Error	1.4145
Observations	600

	Coefficients	**Standard Error**	**t Stat**	**P-value**
Intercept	2.9454	0.0637	46.2726	$8.978 \cdot 10^{-200}$
X variable	5.6961	0.3425	16.6328	$2.438 \cdot 10^{-51}$

E Programming Code

For clarity purposes, Github is referred to for the programming code of this thesis. One could obtain the code by requesting access of the project at https://github.com/KoenHendrikse/MAPFW_Pricer/tree/master. CPLEX version 12.10.0 was used in C++. Instances for the agents (train units), Services and Graphs are used. As various classes and headers were created, a short description of each file is provided here.

Agent.cpp and Agent.h

These files create the Agent class which represents a train unit. It contains a method that reads in the data. Agents have starting times and locations and a set of services that should be performed.

Service.cpp and Service.h

The Service class is created and read in here. It contains a set of tracks on which the service could be performed. These could be added to the service when the program reads in the graph.

Node.cpp and Node.h

The Node class is created by these files. Besides the location of the node, it encompasses whether it is a gateway track and whether waiting movements are allowed. The class contains the locations of the neighbours, from which the edges are formed. Furthermore, the node class contains incoming, outcoming and self-recurring edge sets which are important for solving the Pricing Problem. Nodes are created when reading in the graph.

Edge.cpp and Edge.h

The edge class is described by a 'to' node and a 'from' node, it is not time-dependent. Furthermore, each edge contains an identification.

Graph.cpp and Graph.h

The graph class, that is created here, encompasses the shunting yard infrastructure. It contains the number of time steps and all nodes and edges. The first stage of constructing the graph consists of reading in the data and constructing the nodes. Afterwards, the edges are constructed and placed in the correct sets of the corresponding nodes.

Pathsolver.cpp

This file contains the MIP methods to solve the shortest path with waypoints with durations. It contains four functions that are described below.

pathSolver

This function represents MIP1 as described in Section 6.3. The input of the function is the shunting yard graph, the map of services, the agent and an integer used for the random seed. This random seed is used to sample the reduced costs of edges from a Uniform distribution. A CPLEX model in accordance with MIP1 is created and solved. The file PathSolver.lp is created for clarity. The objective value of the optimal path is returned.

pathSolver2

Compared to the previous function, the only difference is that the CPLEX model of this function corresponds with MIP2 as described in Section 6.4.

pathSolverG

Here, the first function is extended by the gateway extension. This is done by disregarding gateway edges that could not be used from the model.

pathSolver2G

This function corresponds with the second function. However, it is extended by including the gateway extension as described above.

Main

In this file, all relevant functions are called in the correct order to solve the problem. Furthermore, the wall time of solving the PathSolver functions is measured. Based on the exact research settings, adaptions can be easily made here.