

ERASMUS UNIVERSITY ROTTERDAM

ERASMUS SCHOOL OF ECONOMICS

MASTER'S THESIS ECONOMETRICS AND MANAGEMENT SCIENCE

SPECIALIZATION OPERATIONS RESEARCH AND QUANTITATIVE LOGISTICS

Solving a Two-Stage Assignment Problem in Rail Freight Transportation

Using a Greedy Construction Heuristic Framework and a Genetic Algorithm

Abstract

Freight transportation is an increasingly relevant element of our industrialized society. This paper aims to solve a specific two-stage assignment problem in rail freight transportation. In the first stage, wagons need to be assigned in such a way that the profit is maximized. The amount of profit partially depends on the way shipments are assigned to the wagons, which is the second stage of the problem. We propose a Greedy Construction Heuristic to solve the problem, which needs to be combined with an algorithm for the second stage of the problem. Two construction heuristics and an exact formulation are investigated as a second-stage algorithm. Furthermore, we propose a Genetic Algorithm, specifically implemented for this problem. All solution methods are evaluated based on a data set provided by a large European intermodal transportation company and a smaller data set. The computation time used by the Greedy Construction Heuristic depends on the second-stage algorithm. This also determines the quality of the solutions. Generally, better solutions can be found by using more time-consuming algorithms. The Genetic Algorithm outperforms the Greedy Construction Heuristic, but also takes more time.

Author:

Martijn BIJ DE VAATE (476061)

Supervisor:

dr. T.A.B. (Twan) DOLLEVOET

Date final version:

November 23, 2022

Second assessor:

dr. R. (Remy) SPLIET

Contents

1	Introduction	1
2	Problem Description	3
2.1	Network and train services	3
2.2	Wagons and compositions	3
2.3	Shipments	4
2.4	Shunting	5
2.5	Constraints	7
2.5.1	Compositions	7
2.5.2	Shipments	8
2.6	Two-stage problem	8
2.7	Data	9
3	Literature Review	11
4	Methodology	14
4.1	Greedy Construction Heuristic	14
4.1.1	Randomized Greedy Construction Heuristic	18
4.2	Shipment Loading	18
4.2.1	Greedy Shipment Loading	19
4.2.2	Sorted Shipment Loading	21
4.2.3	Exact Formulation	22
4.2.4	Relax & Fix	26
4.2.5	Ignore & Fix	27
4.3	Genetic Algorithm	27
4.3.1	Initial Population	28
4.3.2	Crossover	28
4.3.3	Mutation	32
4.3.4	Population Construction	32
5	Results	35
5.1	Greedy Construction Heuristic	35
5.1.1	Greedy Shipment Loading	35
5.1.2	Sorted Shipment Loading	36
5.1.3	Exact Formulation	38
5.1.4	Relax & Fix	39
5.1.5	Ignore & Fix	40
5.2	Genetic Algorithm	42
5.2.1	Initial Population	42

5.2.2	Crossover	43
5.2.3	Mutation	44
5.2.4	Genetic Algorithm	44
6	Conclusion	49
6.1	Main Findings	49
6.1.1	Greedy Construction Heuristic	49
6.1.2	Genetic Algorithm	50
6.2	Discussion and Further Research	50
6.3	Management Advice	52
7	Appendix	54
A	Extended Data Analysis	54
A.1	Regular instance	54
A.2	Small instance	55
B	Details Genetic Algorithm Crossover Operation	56

1 Introduction

A key element in modern-day industrialized society is freight transportation. Using ships, airplanes, trains and trucks, products can be delivered to satisfy demand of costumers, who are often far away from production sites. *Intermodal transportation* regards transportation using different modes of transport. Companies performing intermodal transport generally use standardized units, such as shipping containers, which can be easily handled by ships, trains and trucks. In this paper, we focus on a company in the field of transportation by rail. A set of wagons is available to the company. These wagons can be combined to form many trains. Furthermore, the company has to adhere to a timetable with specified times for all relevant connections between terminals. Customers demand to transport their goods between these terminals during a certain time-window. The problem such a company is facing is how to distribute the wagons over the network.

Not all shipping containers can be transported by all wagons, so the decision which is made regarding the distribution of wagons affects the amount of customer demand which can be satisfied. This is also directly relevant for the company itself, because of the revenue related to this. Furthermore, the costs are affected by this decision, because of the depreciation or renting costs of wagons and the cost of fuel.

In this case study, we define the above problem in more detail based on a large European intermodal transportation company. We develop several algorithms to find solutions in a reasonable amount of time. In particular, since we are dealing with long-term planning, we are interested in algorithms which are able to find a solution within eight hours.

We have been provided with a data set from this company, which comprises roughly one third of the wagons and network on which this company operates. The size of this data set will determine which methods are suitable for solving the problem in a reasonable amount of time. The data set will be used to evaluate the methods and compare their results. Furthermore, a smaller data set will be generated, to provide some insight into the scalability of the methods.

The methods which will be investigated in this paper are different versions of a Greedy Construction Heuristic, among others combined with Mixed Integer Linear Programming. A construction heuristic is useful, because the problem is an assignment problem and partial solutions are generally already feasible. This means that a feasible solution can be constructed by assigning wagons iteratively, while ensuring that all constraints are satisfied at all times. Furthermore, a Genetic Algorithm will be proposed. A randomized version of the Greedy Construction Heuristic will be used to generate the initial population.

The Greedy Construction Heuristic generally constructs solutions in a small amount of time. The exact amount of time and the quality of the solutions depends on the version and parameter settings of the heuristic. When applying the different versions of the heuristic to our data set, one of the versions which needed the least computation time, found a solution with a total profit of 13,621,014 in 523 seconds. Our best version of the Greedy Construction Heuristic could improve this to 13,684,274 using a computation time of almost 4 hours. The Genetic Algorithm outperformed the Greedy Construction Heuristic, leading to a solution with a total profit of 13,751,461 in a

computation time of almost 80 hours. However, the Genetic Algorithm is highly parallelizable, meaning the computation time can be reduced significantly.

The remainder of this paper is organized as follows. The problem is defined in detail in Section 2, followed by an overview of the relevant literature in Section 3. The methodology is proposed in Section 4 and the results of implementing this methodology and evaluating it on the data sets are described in Section 5. Finally, Section 6 provides the conclusion and discussion of this paper.

2 Problem Description

This section defines the problem we are dealing with in this paper. Given a company using a set of wagons of different types, the goal is to form compositions of wagons which can be used to perform a set of train services and transport a subset of the set of shipments, in such a way that the profit is maximized. Shipments can also be rejected, meaning that they are not transported. The profit is defined as the revenue obtained by transporting the shipments minus the cost of using the wagons. Additionally, there are three terms which are included in the total profit in order to define some soft constraints, which can be violated against a cost, or satisfied for a reward. These are an estimated shunting cost for changing the initial compositions, a penalty for having many rejected shipments, and a bonus for having long trains. More details on the characteristics of the problem and of these penalty and bonus terms will be given throughout the remainder of this section.

2.1 Network and train services

A *terminal* is a train station to which or from which the company can transport goods. A *connection* is a track between an arrival and departure terminal. In case there are other terminals in between these terminals, it will not be possible to load and unload there. On each of the connections, trains run on a regular basis. The frequency and departure and arrival times are determined beforehand, because the company makes use of the regular train tracks, and is therefore dependent on other rail operators. A *train service* is defined as a combination of a connection and a departure and arrival time. For example, the connection going from A to B could have two train services every week, both departing at 9 AM and arriving at 2 PM, one on Mondays and one on Thursdays. The earliest and latest train service define the planning horizon of the problem, which will generally be at least multiple weeks.

All connections are part of a *family*. The families are disjoint subsets of the set of connections, such that one can traverse all connections in a family without traversing other connections. For example, a family could consist of connections A to B, B to A, A to C and C to A. The assignment of connections to families is also determined beforehand, such that it is not part of the problem we consider in this paper.

2.2 Wagons and compositions

A *wagon* is the smallest part of a train, and it can be loaded with goods to be transported. It has a wagon type, used to determine in which families it can be used, and which goods can be transported on it. A *composition*, sometimes also referred to as a *train*, is a set of wagons, which have been connected to perform train services. Any set of wagons can be connected. A composition can perform train services on different connections, but only when they are in the same family. In other words, all train services defined on connections belonging to the same family, must be performed by the same set of compositions. These compositions may not perform other train services. All compositions performing train services for the same family should be identical in the

number and types of wagons, which has been taken into account when the families were created. This means that the expected shipments on different connections of the same family are somehow similarly distributed regarding the number of shipments per train service and regarding the types of shipments. If this would not be the case, the family could be split into several families, each with their own set of identical compositions.

For some families, there is a bonus for having compositions longer than a certain threshold, due to subsidies. This long train bonus is linear in the amount of length exceeding this threshold, and may be different for each family.

It is assumed that the *rotation planning* has been carried out before solving the problem regarded in this paper. Rotation planning concerns the problem of determining which train services are performed by the same composition and, consequently, how many compositions are needed for each family. For example, after going from B to A, should the composition be used for the train service going from A to B, or for the train service going from A to C? A good rotation planning uses a minimum number of compositions, such that the waiting time for compositions at terminals is minimized.

2.3 Shipments

A *shipment* is the smallest batch of goods which should be transported together. It is related to a connection, and it has a delivery time at the origin terminal and a due time before which it should be at the destination terminal. A shipment should be transported using a train service which runs between those times. It also has a shipment type, which includes the size of the shipment, and a weight. Together, the shipment type and weight determine on which wagons it can be transported and with which other shipments it can be combined on a wagon. There can be at most four shipments on a wagon. The rules for assigning a shipment to one of those four *positions* are defined using the *loading schemes*. Each wagon type has its own loading scheme, which is a set of shipment type and weight combinations or *loading combinations*. A loading combination specifies the allowed shipment types for a shipment on each of the positions on the wagon. Furthermore, depending on the shipment type, the position on the wagon and the weight capacity for the other shipments, there is a weight capacity for each of the shipments.

Figure 1 is a schematic representation of an example of a loading scheme, consisting of six loading combinations. When using the first loading combination, the shipment on position 1 should have shipment type 20, 21 or 22, the shipment on position 3 should have shipment type 30 or 31 and there should not be any shipments on positions 2 and 4. The weight capacity is 17,000 kg for position 1 and 34,000 kg for position 3. For another loading combination, the weight capacity is 23,000 kg for position 1 and 33,000 kg for position 3. Regarding the weight capacity, this pattern generally holds, because the weight should be somehow balanced on the wagon; A lighter shipment on one side of the wagon allows for a heavier shipment on the other side, but generally the total weight capacity is less when the weight is distributed less evenly.

A wagon should always be loaded according to one of the loading combinations in the loading

scheme of the wagon type, however, it is also allowed to leave positions or the entire wagon empty. The positions do not directly correspond to a physical position on the wagon. For example, when loading two shipments onto a wagon according to one of the loading combinations from Figure 1, they might be directly next to each other, while there can be another shipment on position 2 when using a wagon of another type.

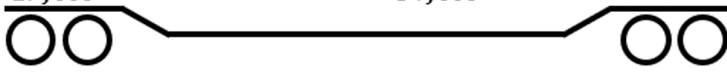

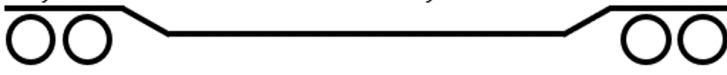
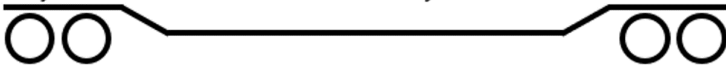
Loading Scheme LS1		Used for wagon types W1, W2, W4, W7			
Loading Combination		Position 1	Position 2	Position 3	Position 4
LC1	Shipment type: Weight capacity:	S20/S21/S22 17,000	- -	S30/S31 34,000	- -
					
LC2	Shipment type: Weight capacity:	S20/S21/S22 20,000	- -	S30/S31 33,500	- -
					
LC3	Shipment type: Weight capacity:	S20/S21/S22 23,000	- -	S30/S31 33,000	- -
					
		...			
LC6	Shipment type: Weight capacity:	S20/S21/S22 30,000	- -	S30/S31 30,000	- -
					

Figure 1: Example of a Loading Scheme.

2.4 Shunting

Since shunting costs are relatively high, we do not allow for compositions to change during the planning horizon in this problem. On the day of operation, the compositions should only be changed if there is some unforeseen reason and it is absolutely necessary, for example because of a broken wagon. There is one exception to this, which concerns the so-called *Y-shuttles*. A Y-shuttle is a pair of train services which partly run on the same tracks. A schematic example of two Y-shuttles is given in Figure 2. Here, the first Y-shuttle consists of the train service from A to B, departing at 9 AM and arriving at 2 PM, and the train service from A to C via B, departing at 9 AM and arriving at 5 PM. In this case, both train services go from A to B between 9 AM and 2 PM and their compositions will be coupled, such that they form one composition or train. At B, the train

is decoupled, and one part continues to C, while the other part stays at B. On the way back, this could be the other way around. The two parts would then be coupled again. We will refer to train services such as the train service from A to B as the *short leg*, as opposed to train services which are the *long leg*, such as the train service from A to C. In families without Y-shuttles, all train services will be regarded short-leg train services. All train services on the same connection are equal in regard to the Y-shuttles. That is, every connection contains only short-leg train services or only long-leg train services. We will therefore also refer to short-leg or long-leg connections.



Figure 2: Example of two Y-shuttles.

Instead of having identical compositions for all train services in a family, we now have two different compositions, one of which is part of the other one. The composition running on the short-leg connections is the *large composition*. The *small composition* is a subset of the wagons of the large composition, and runs on the long-leg connections. We assume that the remaining wagons, which stay at the terminal when the composition for the long leg is decoupled, cannot continue on their own. They need to be coupled with the small composition again before they can perform another (short-leg) train service. This does not need to be the small composition to which they were coupled before. It could also be another small composition from the same family, since these need to be identical. It is part of the rotation planning to find a way to let the large and small compositions perform the short-leg and long-leg train services of a family. Since this is all predetermined, and out of scope for the problem considered in this paper, the shunting costs associated with coupling and

decoupling the trains are not included when calculating the total profit for a certain set of wagon compositions.

Regarding the long train bonus, the sets of small and large compositions of one family behave as two different families. For example, the bonus may apply only to the large compositions, or the size of the bonus may be different for the small and large compositions.

Before the start of the planning horizon, most of the wagons are already used by the company, meaning they are somewhere in a composition at one of terminals. We will refer to these compositions as the *initial compositions*. They should be taken into account when finding new compositions for the wagons. Removing a wagon from a composition has a shunting cost, as well as adding a wagon to it. The complexity of these shunting operations will not be considered. Instead, there will be a fixed, but adjustable, shunting cost for removing and for adding one wagon. The initial compositions do not need to satisfy the constraints for the new compositions. For example, compositions may be too long for the terminals they visit, or contain wagons which are not allowed in countries they traverse. The wagon capacity does not even need to be satisfied, meaning that a change in the number of wagons at the start of the planning horizon is not problematic. This would simply mean there will be shunting costs, because the initial compositions need to be changed to create feasible compositions.

2.5 Constraints

2.5.1 Compositions

Wagons are assigned to compositions, or rather to families, which require a certain number of compositions. Regarding the assignment of wagons to compositions, we firstly have the capacity constraints. For each wagon type, the total number of wagons assigned to the compositions should not exceed the number of available wagons. Secondly, not all wagons are allowed on every train service. Because all compositions in a family should be identical, a wagon needs to be allowed on every train service to be assigned to a composition. Or, in case of a family with Y-shuttles, it should be allowed on all long-leg train services to be part of the small composition and it should be allowed on all short-leg train services to be part of the large composition. Reasons for a wagon to be not allowed on a train service could be certain regulations which are specific to the country or rail operator. Also, for some wagons and train services there are physical limitations, for example because of narrow tunnels. Lastly, there are constraints on the compositions as a whole. For each family, there is a maximum length for the large compositions and, if the family has Y-shuttles, for the small compositions. Furthermore, there is a weight capacity for the large and small compositions. The weight capacity does not only apply to the weight of the wagons, but also to the weight of the assigned shipments. This constraint should therefore be satisfied for each train service. For example, once some wagons and shipments have been assigned, it depends on the current weight of all relevant train services whether another wagon can be assigned to the family.

Some families have specific wagon requirements, independent of the shipments. This means certain families need at least a certain number of wagons of a certain type, for the small or large

compositions. These requirements mainly come from *slot agreements*, which are agreements with customers to reserve a spot on a train at all times. The customer has priority to use these spots, but does not always use them. For the long term planning of composing the trains, we need to make sure that the compositions are created in such a way that these spots are always available, independent from the shipments.

2.5.2 Shipments

Shipments are assigned to train services by assigning them to wagons which are used in the compositions performing the train services. Regarding the assignment of shipments to train services, we again have the constraint on the total weight of a train service, consisting of wagon weight and shipment weight.

Besides this, regarding the assignment of shipments to wagons, the set of shipments which are assigned to a wagon of a certain type should satisfy (at least) one of the loading combinations in the loading scheme of the wagon. This includes a weight capacity for the shipments on the different positions, meaning that we have two constraints related to weight. The difference between this constraint and the constraint on the total weight of a train service is that this constraint considers a specific wagon and the balance of shipments on a wagon, while the other constraint only considers the total weight of a train service without taking into account how the weight is distributed. Both constraints need to be satisfied for a feasible solution to the problem.

Lastly, we have a constraint defined on the connections. For certain connections, there will be a penalty when the accepted proportion of shipments stays below a certain limit. This proportion is measured in *shipping units*, so each shipment will consist of a certain number of shipping units. We will refer to this as the acceptance penalty.

2.6 Two-stage problem

Even though the assignment of shipments to wagons is relevant to evaluate the quality of the wagon compositions, it is not relevant itself. This is because the compositions are determined long before the day of operation, when the actual shipments are not known yet. On the day of operation, the actual shipments are known, which is when they are planned on wagons. For our problem, we assume the shipment data represents the actual shipments. For example, this could be an estimation of future shipments or historical data. This also enables us to use the model for a historical period to find out which compositions could have been formed for a certain amount of profit.

Because of this, the problem considered in this paper can be defined as a two-stage problem. In stage one, the *wagon assignment* is determined. Once this has been done, we know the number and types of wagons which are used for every train service. Since wagons of the same type are equivalent in our problem, this also allows us to choose wagons for a train service. The problem of stage two concerns the *shipment loading*, or the assignment of shipments to wagons and train services. In this problem, many shipments are completely independent of each other, in the sense that they do not affect the same constraints. This is true for any pair of shipments from different families,

but also for a pair of shipments from different connections, unless the train services running on these connections are part of the same Y-shuttle. Such connections will be referred to as *dependent connections*. We define a *minimum-size self-dependent set of connections* as a subset of the set of connections from the same family which cannot be divided into two or more non-empty sets such that every two dependent connections are in the same set. For all families without Y-shuttles, every connection forms such a minimum-size self-dependent set of connections, while for families with Y-shuttles, there will be at least one set with multiple connections.

The concept of minimum-size self-dependent sets of connections allows us to split the problem of shipment loading into many independent subproblems, which can be solved in parallel to solve the complete problem of shipment loading. Furthermore, when only part of the wagon assignment has changed, the shipment loading problem does not need to be fully re-solved. Instead, only some of the subproblems need to be re-solved.

2.7 Data

A data set has been made available by the company facing the problem described in this section. It consists of 19 families with between 2 and 12 connections each. The total number of connections is 88. The planning horizon consists of 50 days, during which there are a total of 2,496 train services. On the busiest connection, there are 13 train services per week on average, while on the least busy connection, there is only one train service during the planning horizon. Among all train services, there are 482 pairs which form a Y-shuttle. These come from 8 out of 19 families. There are 1,654 available wagons, which belong to 49 different types. Together, these wagon types have 5,516 different loading combinations. Initially, all wagons are assigned to one of the families, which will be used to determine the shunting costs. For 14 out of 19 families, wagon requirements are given, consisting of between 2 and 20 wagons per family. Together, 140 wagons are required. The initial assignment of wagons is such that the wagon requirements are already satisfied. Lastly, there are 58,262 shipments available for transportation during the planning horizon. Appendix A.1 contains some graphs summarizing the set of shipments based on their revenue, weight and size in terms of shipment units. We will refer to this data set as the *regular* data set or instance.

Furthermore, a *small* data set or instance has been constructed. The small data set is a subset of the regular data set, consisting of a total of 2 families and 21 train services. One of the families has Y-shuttles. The planning horizon spans 12 days instead of 50 days for the regular data set. There is a total of 210 wagons of 10 different wagon types. This number of wagons has been chosen in such a way that the small instance and the regular instance are of similar restrictiveness. A total of 24 wagons is necessary to be assigned to the families due to the wagon requirements. Lastly, there are 1,002 shipments to be transported. Appendix A.2 summarizes the set of shipments using graphs.

In Table 1, we introduce the sets which formally define the data of an instance of our problem. Next to these sets, all data to calculate the total profit and to evaluate the constraints needs to be defined. The wagon costs, long train bonus and shipment revenue will be given in euros. However,

since the shunting costs and the acceptance penalty are values which can be tuned to evaluate their effect on the solution, the total profit cannot be directly related to a monetary amount. We will therefore omit the euro sign in this paper.

Table 1: Sets of Introduced Terms.

Set	Description
A	set of all connections;
F	set of all families of connections;
L	set of loading combinations for shipments on wagons;
P	set of positions on a wagon;
S	set of all shipments available during the planning period;
T	set of all train services to be performed on connections A during the planning period;
V	set of all wagon types with at least one wagon available for train services;
W	set of all wagons available for train services T .

3 Literature Review

A lot of research has been done regarding freight transportation by rail. Some common topics are locomotive assignment, empty freight car distribution and freight car fleet sizing. Gorman et al. (2014) provide an overview of the application of Operations Research techniques in freight transportation, organized by mode of transport. Regarding load management or train makeup, which our shipment loading stage is a part of, they note that relatively little research has been done in this area, even though efficient use of the capacity of trains can potentially save a lot of money. Some of the papers regarding shipment loading will now be discussed. They all consider the case of one train running from a fixed origin to a fixed destination, with a fixed fleet of wagons to be used and a list of trailers or containers. Unless specified otherwise, all trailers or containers need to be transported.

Feo and González-Velarde (1995) assume each wagon should transport exactly two trailers. An artificial trailer type has been created to serve as the second trailer when a wagon can only transport one trailer in reality. Trailers have a type, which is used to determine which trailers can be combined on one wagon. All restrictions regarding the assignment of trailers to wagons can then be represented by a binary parameter for each combination of a pair of trailer types and a wagon type. Using this parameter, an Integer Linear Programming (ILP) problem is defined. Another solution technique which Feo and González-Velarde (1995) describe, is a Greedy Randomized Adaptive Search Procedure (GRASP). In the constructive phase, pairs of trailers are assigned to wagons iteratively by randomly selecting a combination of a trailer pair and a wagon from a list of candidates. The list of candidates consists of the most difficult combinations, where trailer pairs and wagons with few assignment possibilities are considered difficult. The weight of trailers and the weight capacity of wagons are not considered, however, this could be incorporated by including the weight or weight category of trailers in the trailer types.

Ernst and Pudney (2001) indeed model the weight restrictions by categorizing the containers into light, medium weight, heavy and very heavy containers. These categories are used to define which containers can be transported on the same wagon simultaneously. Using these categories, the total number of loading configurations stays relatively low, compared to defining loading configurations using all occurring weights or using more than four weight categories. To solve the problem of assigning containers to wagons, they propose an ILP problem in which the decision variables represent the number of wagons of a certain wagon type which are loaded according to a certain loading configuration. Each wagon can only be loaded according to one loading configuration. In this model, loading configurations which are not feasible for a certain wagon type can be easily handled by restricting the relevant decision variable to zero. The assignment of loading configurations to wagons needs to be chosen in such a way that exactly all containers are assigned. This can be done by ensuring that the number of containers of a certain type which is necessary given the loading configuration assignment is equal to the total number of containers of this type.

Corry and Kozan (2008) consider the same problem as Feo and González-Velarde (1995), but without the assumption of a maximum of two trailers or containers per wagon. They define a Binary

Programming (BP) problem with a decision variable which denotes whether a certain container is assigned to a certain slot on a certain pattern for a certain wagon. The choice of a pattern for a wagon is modelled without the need of additional variables by two constraints. Firstly, only one of the feasible patterns for a certain wagon may have a container assigned to its first slot. Secondly, for each pattern, the value of variables corresponding to all other slots than the first slot of a pattern must be equal to the value of the variable of the corresponding first slot. This means that for the chosen pattern, all slots must be used, while for all other patterns, no slots can be used. Corry and Kozan (2008) also define a version of the problem in which containers are not directly assigned to wagons. Instead, a pattern is chosen for each wagon, in such a way that the number of slots available for each type of container is equal to the total number of containers of that type. The containers can then be assigned to wagons by choosing any of the feasible slots. For this alternative problem definition, the container types need to be disjoint, that is, containers cannot have more than one type. Compared to their original problem definition, this version uses a much smaller amount of time to be solved. Other solution techniques considered by Corry and Kozan (2008) are a Local Search algorithm with two different neighborhoods and a Simulated Annealing algorithm.

Lai et al. (2008) claim to be the first to solve the problem of assigning containers to wagons with the objective of minimizing the fuel consumption through loading containers in such a way that the aerodynamic resistance of the train is as low as possible. However, they only consider wagons with one container or with two containers on top of each other.

Bruns and Knust (2012) solve a problem very similar to our shipment loading problem. They consider one train of wagons of different types, and a set of containers and trailers, which also have different types and weights. Not all containers and trailers need to be assigned to a wagon. The assignment of containers and trailers to wagons depends on the wagon type. For this, restrictions identical to our *loading schemes* are used. Wagons have an initial configuration, or *loading combination* using our terminology from Section 2. Setup costs arise for changing this configuration. Furthermore, costs are associated with transporting the containers and trailers to the wagons, which is mostly dependent on the distance from the storage place to the wagon. The objective is to maximize the utilization of the train and to minimize the setup costs and transportation costs.

Three different ILP problems are defined by Bruns and Knust (2012) to solve the problem. The first two directly use the loading schemes, while the third looks at the technical reasons behind the loading schemes. Not only should the total weight of containers on a wagon not exceed the weight capacity of the wagon, the total weight supported by each of the bogies should also not exceed a certain value. Moreover, the weight supported by one bogie should at most be three times the weight supported by the other bogie, meaning that the weight on the wagon should be somewhat balanced. A similar approach is taken by Anghinolfi et al. (2014), but with the objective of maximizing the (weighted) number of containers. A drawback of this approach is that there should be data about the positions of the containers and trailers with respect to the bogies to calculate the weight supported by each of the bogies. The advantage is, as Bruns and Knust (2012) remark, that the requirement of balanced weight can be easily modelled. For approaches which directly use the loading schemes,

this is not the case, since there is only an upper limit on the weight on each position. Another advantage is that the weight restrictions from the loading schemes are not discrete anymore, which makes the model more realistic.

Lastly, Upadhyay et al. (2017) propose a BP problem for assigning containers to wagons with the possibility for double-stacked containers. They only consider four types of containers and a small number of loading patterns. Not all available containers need to be transported. Instead, the goal is to load a selection of containers to maximize the profit or minimize the tardiness, which ensures that containers which have been available for a relatively large amount of time are prioritized.

All papers discussed so far consider stage two of our two-stage problem. The first stage, concerning wagon assignment, is a more general allocation problem, for which many different solution techniques exist. Groups of wagons of different types need to be assigned to families, with specific requirements. For example, a GRASP can be used for this, which has first been described by Feo and Resende (1989).

Given a randomized construction heuristic, such as the one used for the GRASP, it is also possible to define a Genetic Algorithm. The construction heuristic is used to create an initial population, and the fitness of the solutions is determined by the total profit corresponding to the wagon assignment and shipment loading. A simple version of the Genetic Algorithm has been introduced by Holland (1992). As shown by Katoch et al. (2021), variants of the Genetic Algorithm have been successfully used to solve many different problems, among others in the field of operation management.

4 Methodology

In this section, the methodology for solving the problem considered in this paper will be introduced. Section 4.1 contains a general Greedy Construction Heuristic for the first stage of the problem, the wagon assignment. It needs to be combined with a solution method for the second stage of the problem, the shipment loading. Several ways to solve the shipment loading problem will be defined in Section 4.2. Lastly, a Genetic Algorithm will be defined for the problem as a whole in Section 4.3.

4.1 Greedy Construction Heuristic

A Greedy Construction Heuristic for the problem considered in this paper can be defined as follows. Groups of wagons of the same type are iteratively assigned to families. In case of a family with Y-shuttles, they are either assigned to the small compositions or to the large compositions. The number of wagons which is assigned to a family per iteration is equal to the number of compositions in the family.

We will refer to such a group of wagons, which is assigned to the small or large compositions of a family at once, as a *wagon group*. Even when using another solution method, the number of wagons of a certain wagon type which have been assigned to the small or large compositions of a certain family, will always be a multiple of the number of compositions in the family. We can therefore always split the set of wagons into wagon groups. It is useful to consider the wagons through their wagon group, rather than directly, because in the context of this problem, it is unknown which wagon is going to be used for which train service. Shipments are therefore assigned to a wagon group, rather than to a specific wagon.

In each iteration of the Greedy Construction Heuristic, all pairs of wagon types and families are considered, for the small and large compositions. These are all our candidate wagon groups, out of which one has to be selected for assignment. The wagon group which will be selected is the wagon group which increases the total profit of the solution the most without creating an infeasible solution. Each of the wagon groups is evaluated in two steps.

The first step is to evaluate whether the wagon group is feasible for assignment. In case of a family without Y-shuttles, we only need to consider the wagon group for the large compositions of the family, because such a family does not have small compositions. In case of a family with Y-shuttles, both wagon groups need to be considered separately, because the feasibility of the wagon groups for the small and large compositions may differ. A wagon group is feasible if (1) there are still enough wagons of the wagon type available to assign one wagon to each composition in the family, (2) the wagon type is allowed to be used for the family in small or large compositions, depending on the wagon group, (3) the length limit for the small or large compositions will not be exceeded after adding a wagon to each of them and (4) the weight limit is not exceeded after adding a wagon to each of them for any of the train services using these compositions.

The second step consists of determining the increase in profit which can be obtained by assigning the wagon group. This step only needs to be performed for feasible wagon groups. The increase in

profit consists of wagon costs, shunting costs, long train bonus, shipment revenue and acceptance penalty. For the last two terms, the shipment loading needs to be performed for each of the minimum-size self-dependent connection sets in the family. For families with Y-shuttles, when considering a wagon group for the small (large) compositions, the shipment loading only needs to be performed for connection sets with at least one long-leg (short-leg) connection, because all other connection sets are not affected by the wagon assignment. Several ways to perform the shipment loading will be defined in Section 4.2.

Once the increase in profit has been calculated for every wagon group, the wagon group with the highest increase in profit is assigned, including the shipments which lead to this increase in profit. In the next iteration, for many wagon groups, the feasibility and the profit increase does not have to be evaluated again. Only wagon groups using the same wagon type as the wagon group which was assigned in the previous iteration, or which consider the family to which wagons were assigned in the previous iteration need to be reconsidered. Concerning equal wagon types, the feasibility of the wagon group should be reevaluated, because the number of available wagons has changed. Concerning equal families, the increase in profit should be recalculated, because the unassigned shipments have changed. Furthermore, the feasibility should be reevaluated, because the length of the compositions and the weight of the train services has been increased in the previous iteration.

Instead of considering the profit increase for each wagon group, the profit increase divided by the number of wagons in the wagon group can be considered. When using the profit increase, large wagon groups will often be preferred to small wagon groups. However, small wagon groups may actually be more profitable when considering the profit increase per wagon. The results of these two different evaluation methods will be discussed in Section 5.

The Greedy Construction Heuristic is terminated when there are no more feasible wagon groups which can increase the total profit, for example when all wagons have been assigned, when all shipments have been assigned, or when the revenue which can be obtained when adding wagons does not outweigh the costs.

When performing the Greedy Construction Heuristic as described so far, the wagon requirements, which are constraints on the number of wagons of a certain type to be assigned to the small or large compositions of a certain family, may not be satisfied. To avoid this, wagon groups should be assigned to satisfy the wagon requirements before all possible wagon groups are evaluated. The shipment loading should also be performed. The shipment loading problem can be solved per wagon group, but depending on the shipment loading algorithm, it might also be possible to perform the shipment loading for multiple wagon groups from the same family at once. This will likely result in a higher increase in profit.

Algorithm 1 provides an overview of this Greedy Construction Heuristic. It uses Algorithm 2, Algorithm 3 and Algorithm 4, which are the algorithms for creating a partial solution which satisfies the wagon requirements, for determining whether a wagon group is feasible, and for calculating the profit increase of a wagon group, respectively.

Algorithm 1: GreedyConstructionHeuristic(WAGON_TYPES, FAMILIES)

Result: Feasible solution

SOLUTION = **Algorithm 2**(FAMILIES); A solution satisfying the wagon requirements;

CONTINUE = true;

while CONTINUE **do**

 CANDIDATES = empty list of tuples of a wagon group and the corresponding profit increase;
 Note: In many cases, the candidate list of the previous iteration can be largely copied;

for each tuple of a WAGON_TYPE in WAGON_TYPES, a FAMILY in FAMILIES and a binary variable LARGE_COMPOSITIONS indicating whether the large (TRUE) or small (FALSE) compositions of FAMILY should be considered **do**

 WAGON_GROUP = a wagon group corresponding to WAGON_TYPE, FAMILY and LARGE_COMPOSITIONS;

 FEASIBLE = **Algorithm 3**(WAGON_GROUP, SOLUTION); Evaluate the feasibility of this wagon group;

if FEASIBLE **then**

 Assign WAGON_GROUP to SOLUTION;

 PROFIT_INCREASE = **Algorithm 4**(WAGON_GROUP); Calculate the profit increase obtained when performing shipment loading on this wagon group;

 PROFIT_INCREASE = PROFIT_INCREASE + long train bonus - wagon costs - shunting costs; Corresponding to the assignment of this wagon group;

 Undo the assignment of wagons and shipments;

if PROFIT_INCREASE > 0 **then**

 Add the tuple {WAGON_GROUP, PROFIT_INCREASE} to CANDIDATES;

end

end

end

if CANDIDATES is empty **then**

 CONTINUE = false;

else

 {BEST_WAGON_GROUP, HIGHEST_PROFIT_INCREASE} = tuple in CANDIDATES with highest PROFIT_INCREASE;

 Assign BEST_WAGON_GROUP to SOLUTION;

 Perform shipment loading on every minimum-size self-dependent connection set with at least one long-leg or short-leg (depending on BEST_WAGON_GROUP) train service in the family to which BEST_WAGON_GROUP is assigned;

end

end

return SOLUTION

Algorithm 2: SatisfyWagonRequirements(FAMILIES)

Result: Partial solution in which the wagon requirements are satisfied

SOLUTION = empty solution;

REQUIRED_WAGON_GROUPS = wagon groups corresponding to the wagon requirements;

for *each* WAGON_GROUP in REQUIRED_WAGON_GROUPS **do**

 Assign WAGON_GROUP to SOLUTION;

 FAMILY = family in FAMILIES to which WAGON_GROUP is assigned;

 Perform shipment loading on every minimum-size self-dependent connection set with at least one long-leg or short-leg (depending on WAGON_GROUP) train service in FAMILY;

end

return SOLUTION

Algorithm 3: EvaluateWagonGroupFeasibility(WAGON_GROUP, SOLUTION)

Result: Binary variable indicating whether WAGON_GROUP can be assigned to SOLUTION without violating the constraints

WAGON_TYPE = wagon type of the wagons in WAGON_GROUP;

FAMILY = family to which WAGON_GROUP is assigned;

AVAILABLE_NUMBER_OF_WAGONS = number of unassigned wagons of WAGON_TYPE;

FEASIBLE = true;

if AVAILABLE_NUMBER_OF_WAGONS < *number of compositions in FAMILY* **then**

 FEASIBLE = false;

else if *wagons of WAGON_TYPE are not allowed for the small or large (depending on WAGON_GROUP) compositions of FAMILY* **then**

 FEASIBLE = false;

else if *the small or large (depending on WAGON_GROUP) compositions of FAMILY exceed their maximum length after adding a wagon of WAGON_TYPE to them* **then**

 FEASIBLE = false;

else if *there exists a train service using the small or large (depending on WAGON_GROUP) compositions of FAMILY for which its weight exceeds its weight capacity after adding a wagon of WAGON_TYPE to it* **then**

 FEASIBLE = false;

end

return FEASIBLE

Algorithm 4: CalculateWagonGroupProfitIncrease(WAGON_GROUP)

Result: Profit increase which would result from assigning shipments to WAGON_GROUP

FAMILY = family to which WAGON_GROUP is assigned;

ACCEPTANCE_PENALTY_BEFORE = combined acceptance penalty of connections in FAMILY;

Perform shipment loading on every minimum-size self-dependent connection set with at least one long-leg or short-leg (depending on WAGON_GROUP) train service in FAMILY, for example using **Algorithm 5** or by solving **Mixed Integer Linear Programming problem (1)-(13)**;

REVENUE_INCREASE = combined revenue of the shipments assigned to WAGON_GROUP;

ACCEPTANCE_PENALTY_AFTER = combined acceptance penalty of connections in FAMILY;

ACCEPTANCE_PENALTY_DECREASE = ACCEPTANCE_PENALTY_BEFORE -

ACCEPTANCE_PENALTY_AFTER;

PROFIT_INCREASE = REVENUE_INCREASE + ACCEPTANCE_PENALTY_DECREASE;

return PROFIT_INCREASE

4.1.1 Randomized Greedy Construction Heuristic

Applying the Greedy Construction Heuristic generally does not result in the optimal solution, because of the greedy nature of the algorithm. For example, selecting the second-best assignment possibility in the first iteration, may leave better possibilities during later iterations, such that the final solution has a higher total profit. A randomized version of the Greedy Construction Heuristic can be defined as follows. After evaluating the increase in profit (or increase in profit per wagon) for all wagon groups, determine the highest increase in profit among these wagon groups. Randomly select one of the wagon groups which has an increase in profit of at least a certain percentage of this highest increase in profit, e.g. 90%. We will refer to this percentage as the threshold parameter α .

This Randomized Greedy Construction Heuristic can be performed many times, which will likely result in many different solutions, with different amounts of total profit. If this heuristic can be performed in a relatively small amount of time, a solution strategy could be to perform the heuristic many times until the available amount of time runs out, and use the best solution as the final solution.

4.2 Shipment Loading

The problem of shipment loading is the second stage in the problem of this paper. As explained in Section 2.6, it can be divided into many subproblems by regarding only one or a few connections at a time. We will denote a minimum-size self-dependent set of connections by $\tilde{A} \subseteq A$. This division into subproblems has two advantages. Firstly, the subproblems are smaller than the original problem, which will generally mean they are easier to solve. Secondly, since the subproblems are independent from each other, the total computation time can be reduced by solving them in parallel.

In this section, we will propose two different construction heuristics for the problem of shipment loading. Furthermore, we will define a mathematical model for the problem of assigning shipments to wagons, given the assignment of wagons to the train services on the connections of such a set \tilde{A} . Here, we assume that the wagons have been assigned to train services in such a way that all the relevant constraints are satisfied. Lastly, two heuristic techniques will be defined for this specific mathematical model.

In this paper, we usually solve the shipment loading problem for one wagon group at a time. The solution methods proposed in this section have been explained for this case. However, the construction heuristics can be easily generalized to solve the problem with multiple wagon groups. The mathematical model already has been defined in a general way, such that it can be used to solve such problems without any change.

4.2.1 Greedy Shipment Loading

Similarly to the Greedy Construction Heuristic for wagon assignment, the shipment loading problem can also be solved in a greedy way, by considering all shipments which need to be transported on one of the connections of the connection set, but which have not been assigned to a pair of a train service and a wagon group yet. We define this Greedy Shipment Loading algorithm as follows. In each iteration, the shipment with the highest profit increase which is allowed on the wagon group we currently consider for some feasible train service should be added to it. This profit increase consists of the shipment revenue and the acceptance penalty decrease related to assigning the shipment. Given a train service running on the relevant connection, we can evaluate whether the shipment is allowed to be assigned to it by checking two things. Firstly, the total weight of the train service should not exceed the maximum weight after adding the shipment. In case of a long-leg train service, this should also be checked for the corresponding short-leg train service, since its total weight will also be increased by the shipment. Secondly, there should be a feasible loading combination for the relevant wagon type which can transport all shipments which have already been assigned to the combination of this train service and wagon group, including the candidate shipment.

When checking the feasibility of a shipment, it is sufficient to find one train service which can transport the shipment. Starting from the second iteration, it is not necessary to find such a feasible train service again for all shipments. The list of shipments from the previous iteration can be reused, after removing the shipment which has been assigned. Only for those shipments for which the feasible train service was equal to the train service to which a shipment was assigned in the previous iteration, we need to find a feasible train service again. In some cases, there might be another feasible train service, while in other cases, there will not be a possibility anymore to transport the shipment using the wagon group we currently consider.

This is repeated until no more shipments can be added to the wagon group using any of the train services. After terminating the algorithm, the total increase in revenue and decrease in acceptance penalty can be determined, which will be used in the Greedy Construction Heuristic to select the most profitable wagon group. An overview of Greedy Shipment Loading is given in Algorithm 5.

Algorithm 5: GreedyShipmentLoading(WAGON_GROUP, CONNECTION_SET, LARGE_COMPOSITIONS)

Result: Assignment of shipments to a pair of a wagon group and a train service

UNPLANNED_SHIPMENTS = list of shipments which are available for transportation on the connections in CONNECTION_SET and which have not been assigned to a train service yet;

CONTINUE = true;

while CONTINUE **do**

 CANDIDATES = empty list of tuples of a shipment and the corresponding profit increase;

Note: In many cases, the candidate list of the previous iteration can be largely copied;

for each SHIPMENT in UNPLANNED_SHIPMENTS **do**

if LARGE_COMPOSITIONS and [CONNECTION_SET has a train service which needs large compositions and which can transport SHIPMENT using WAGON_GROUP] **then**

 PROFIT_INCREASE = revenue of SHIPMENT + decrease in acceptance penalty

 resulting from assigning SHIPMENT;

 Add {SHIPMENT, PROFIT_INCREASE} to CANDIDATES;

end

if not LARGE_COMPOSITIONS and [CONNECTION_SET has a train service which needs small compositions and which can transport SHIPMENT using WAGON_GROUP] **then**

 PROFIT_INCREASE = revenue of SHIPMENT + decrease in acceptance penalty

 resulting from assigning SHIPMENT;

 Add {SHIPMENT, PROFIT_INCREASE} to CANDIDATES;

end

end

if CANDIDATES is empty **then**

 CONTINUE = false;

else

 BEST_CANDIDATE = tuple in CANDIDATES with highest PROFIT_INCREASE;

 Assign BEST_CANDIDATE to a train service which can transport it using

 WAGON_GROUP;

 Remove BEST_CANDIDATE from UNPLANNED_SHIPMENTS;

end

end

return Assignment of shipments to a pair of a wagon group and a train service

Instead of evaluating shipments by the increase in profit, we can also consider the increase in profit divided by the weight of the shipment. This would be a measure for the profitability, while taking into account how much of the capacity is used. However, the capacity of a wagon is not only determined by the weight, but also by the number and types of shipments. Furthermore, some shipments consist of an empty container, and even though their revenue is generally smaller, the

revenue to weight ratio will be much larger than full containers of the same size. Without a more sophisticated or exact approach, we cannot fully incorporate the wagon capacity, which is a logical result of a greedy heuristic. Both evaluation methods will be discussed in Section 5.

4.2.2 Sorted Shipment Loading

Another construction heuristic which can be used for shipment loading is based on Feo and González-Velarde (1995), who propose a Greedy Randomized Adaptive Search Procedure for a similar problem, in which the construction heuristic sorts combinations of a type of shipment and a type of wagon based on the frequency of the shipment and the wagon. Rare shipments are planned first, and rare wagons are filled first. The idea behind this is that the most difficult planning is done first, such that the more versatile shipments and wagons can still be planned when there are fewer options available. We will use this idea in our Sorted Shipment Loading, in which we consider the difficulty to plan a shipment on the one hand, and the difficulty to plan shipments on a combination of a train service and a wagon on the other hand. With $\hat{S} \subseteq S$, $\hat{T} \subseteq T$ and $\hat{W} \subseteq W$ being subsets of shipments, train services and wagons, respectively, on which we apply the Sorted Shipment Loading, we define $\beta_s, s \in \hat{S}$ and $\gamma_{tw}, t \in \hat{T}, w \in \hat{W}_t$. These are the loading difficulty parameters for shipments and for combinations of a train service and a wagon. Since we do not know which wagon of a wagon group will be used for which train services of the relevant family, we only consider one of the wagons of a wagon group, and assume all train services are performed by this wagon. When applying Sorted Shipment Loading on one wagon group, this results in only one wagon, so parameter γ_{tw} simply reflects the difficulty to plan shipments on train service t .

For shipments, β_s is calculated by counting the number of combinations of a train service and a wagon which can transport shipment s in time, for which the following is true. Shipment s can be assigned to the train service and wagon, such that one of the loading combinations of the wagon is satisfied, without violating the weight capacity of the train service. This is then divided by the total number of combinations of a train service and a wagon which can transport shipment s in time. If this total number is equal to 0, β_s will also be set to 0, indicating that this shipment cannot be planned anymore.

For combinations of a train service and a wagon, γ_{tw} is calculated by counting the number of shipments which can be transported in time using train service t and wagon w , for which the following is true. The shipment can be assigned to train service t and wagon w , such that one of the loading combinations of the wagon is satisfied, without violating the weight capacity of train service t . This is then divided by the total number of shipments which can be transported in time using train service t and wagon w . If this total number is equal to 0, γ_{tw} will also be set to 0, indicating that no more shipments can be planned on this combination of a train service and a wagon.

All combinations $\{s, t, w\}$ of a shipment, a train service, and a wagon used for that train service for which the product of β_s and γ_{tw} is non-zero are considered, starting from the combination for which this product is the smallest. When such a combination exists, the shipment is planned on the train service and wagon. Next, the values of β_s and γ_{tw} are updated. By keeping track of the

numerator and denominator of the division used to calculate β_s and γ_{tw} , most of the calculations do not have to be performed again. For example, for a combination of a train service and a wagon which was not involved in the planning decision, the planned shipment needs to be removed by subtracting 1 from the numerator if the shipment could be transported given the loading combinations and the weight capacity, while the denominator stays the same. γ_{tw} can then be calculated again by dividing the numerator by the denominator.

Shipment s with $\beta_s = 0$ cannot be planned on any of the remaining combinations of a train service and a wagon. Also, the combination of train service t and wagon w with $\gamma_{wt} = 0$ cannot transport any more shipments. Such shipments and combinations of a train service and a wagon should not be considered anymore. The process is repeated by planning another shipment until all shipments have been planned or no more combinations of a train service and a wagon can transport a remaining shipment.

In the Sorted Shipment Loading algorithm as described here, the revenue of shipments is not considered. The objective of the algorithm is not to find a way to load shipments with a large corresponding profit increase, but rather to find a tight shipment loading, in which there are no or only some positions left which could transport another shipment, but for which there are no feasible shipments left. A way to incorporate the revenue would be to redefine β_s by dividing the original value by the revenue or the revenue-weight ratio. This would prefer a shipment with high revenue or revenue-weight ratio to a shipment with lower revenue or revenue-weight ratio for the same loading difficulty.

4.2.3 Exact Formulation

We will now model the shipment loading problem as a Mixed Integer Linear Programming (MILP) problem.

Sets

Firstly, Table 2 defines the sets which are used in the MILP problem and which have not yet been defined in Table 1. Furthermore, we define a function to match train services which are part of the same Y-shuttle as follows: let $t^{long}(t^{short}) \in \tilde{T} \setminus \tilde{T}^{short}$ be the long-leg train service corresponding to short-leg train service $t^{short} \in \tilde{T}^{short}$.

Table 2: Sets for the Exact Formulation of Shipment Loading.

Set	Description
\tilde{A}	minimum-size self-dependent set of connections;
\tilde{S}	set of shipments which are available for transportation on connections \tilde{A} ;
\tilde{T}	set of train services to be performed on connections \tilde{A} ;
\tilde{W}	set of all wagons used for train services \tilde{T} ;
$L_w \subseteq L$	set of loading combinations which are allowed for wagon $w \in \tilde{W}$;
$\tilde{S}_a \subseteq \tilde{S}$	set of shipments which are available for transportation on connection $a \in \tilde{A}$;
$\tilde{S}_t \subseteq \tilde{S}$	set of shipments which can be transported by train service $t \in \tilde{T}$;
$\tilde{T}_s \subseteq \tilde{T}$	set of train services which can transport shipment $s \in \tilde{S}$ regarding time restrictions;
$\tilde{T}^{short} \subset \tilde{T}$	set of train services which are the short-leg train service of a Y-shuttle;
$\tilde{W}_t \subseteq \tilde{W}$	set of wagons which are used for train service $t \in \tilde{T}$.

Parameters

Secondly, Table 3 defines the parameters which are used in the MILP problem. Regarding the weight capacity, we already know the weight of the wagons of a train service. The shipment weight capacity $h_t, t \in \tilde{T}$ can therefore be calculated as the original weight capacity minus the total wagon weight. For train services which are the short leg in a Y-shuttle, parameter h_t is the weight capacity on the shipments for the short-leg train service and for the corresponding long-leg train service, because both kinds of shipments are transported on the route of the short-leg train service.

Table 3: Parameters for the Exact Formulation of Shipment Loading.

Parameter	Description
$d_a \in [0, 1]$	desired lower limit on the proportion of transported shipment units for connection $a \in \tilde{A}$;
q_a	penalty per 100 percentage points below the desired lower limit of transported shipment units for connection $a \in \tilde{A}$;
r_s	revenue of shipment $s \in \tilde{S}$;
u_s	size of shipment $s \in \tilde{S}$ in shipment units;
g_s	weight of shipment $s \in \tilde{S}$;
h_t	total shipment weight capacity for train service $t \in \tilde{T}$;
b_{lstp}	$\begin{cases} 1 & \text{if shipment } s \in \tilde{S} \text{ is allowed on position } p \in P \text{ for loading combination } l \in L \\ & \text{when running on train service } t \in \tilde{T}_s, \\ 0 & \text{otherwise.} \end{cases}$

Variables

Lastly, Table 4 defines the variables which are used in the MILP problem.

Table 4: Variables for the Exact Formulation of Shipment Loading.

Variable	Description
x_{stwp}	$\begin{cases} 1 & \text{if shipment } s \in \tilde{S} \text{ is assigned to train service } t \in \tilde{T}_s \text{ and wagon } w \in \tilde{W}_t, \\ & \text{on position } p \in P, \\ 0 & \text{otherwise;} \end{cases}$
y_{twl}	$\begin{cases} 1 & \text{if for train service } t \in \tilde{T}, \text{ wagon } w \in \tilde{W}_t \text{ uses loading combination } l \in L_w, \\ 0 & \text{otherwise;} \end{cases}$
k_a	the proportion of transported shipment units on connection $a \in \tilde{A}$;
z_a	$\begin{cases} d_a - k_a & \text{if } k_a < d_a, \text{ so the desired proportion of transported shipment units} \\ & \text{is not reached for connection } a \in \tilde{A}, \\ 0 & \text{otherwise.} \end{cases}$

MILP Formulation

Using the sets, parameters and variables defined in this section, we can formulate the problem as a Mixed Integer Linear Programming (MILP) problem.

MILP FORMULATION

$$\max \sum_{s \in \tilde{S}} r_s \sum_{t \in \tilde{T}_s} \sum_{w \in \tilde{W}_t} \sum_{p \in P} x_{stwp} - \sum_{a \in \tilde{A}} q_a z_a \quad (1)$$

$$\text{s.t.} \quad \sum_{l \in L_w} y_{twl} \leq 1, \quad t \in \tilde{T}, w \in \tilde{W}_t \quad (2)$$

$$\sum_{s \in \tilde{S}_t} x_{stwp} \leq 1, \quad t \in \tilde{T}, w \in \tilde{W}_t, p \in P \quad (3)$$

$$\sum_{t \in \tilde{T}_s} \sum_{w \in \tilde{W}_t} \sum_{p \in P} x_{stwp} \leq 1, \quad s \in \tilde{S} \quad (4)$$

$$x_{stwp} \leq \sum_{l \in L_w} y_{twl} b_{lpst}, \quad s \in \tilde{S}, t \in \tilde{T}_s, w \in \tilde{W}_t, p \in P \quad (5)$$

$$\sum_{s \in \tilde{S}_t} g_s \sum_{w \in \tilde{W}_t} \sum_{p \in P} x_{stwp} \leq h_t, \quad t \in \tilde{T} \setminus \tilde{T}^{short} \quad (6)$$

$$\sum_{s \in \tilde{S}_t} g_s \sum_{p \in P} \left(\sum_{w \in \tilde{W}_t} x_{stwp} + \sum_{w \in \tilde{W}_t^{long(t)}} x_{st^{long(t)}wp} \right) \leq h_t, \quad t \in \tilde{T}^{short} \quad (7)$$

$$k_a = \frac{1}{\sum_{s \in \tilde{S}_a} u_s} \sum_{s \in \tilde{S}_a} u_s \sum_{t \in \tilde{T}_s} \sum_{w \in \tilde{W}_t} \sum_{p \in P} x_{stwp}, \quad a \in \tilde{A} \quad (8)$$

$$z_a \geq d_a - k_a, \quad a \in \tilde{A} \quad (9)$$

$$x_{stwp} \in \mathbb{B}, \quad s \in \tilde{S}, t \in \tilde{T}_s, w \in \tilde{W}_t, p \in P \quad (10)$$

$$y_{twl} \in \mathbb{B}, \quad t \in \tilde{T}, w \in \tilde{W}_t \quad (11)$$

$$0 \leq k_a \leq 1, \quad a \in \tilde{A} \quad (12)$$

$$0 \leq z_a \leq d_a, \quad a \in \tilde{A} \quad (13)$$

Objective (1) ensures that the total revenue obtained by transporting shipments minus the corresponding acceptance penalty is maximized.

Constraints (2)-(5) are the assignment constraints. Firstly, Constraints (2) ensure that at most one loading combination is assigned to a combination of a wagon and a train service. Since y_{twl} is only defined for loading combinations which are allowed for wagon w , the assigned loading combination will always be feasible. These constraints leave room for assigning no loading combination at all to a combination of a wagon and a train service, however, this should only happen when there are no shipments assigned to it. Constraints (3) and (4) ensure that each position contains at most one shipment and that each shipment is assigned to at most one position, respectively. Lastly, Constraints (5) are used to make sure that shipments are only assigned to positions which are allowed by the assigned loading combination.

Constraints (6) and (7) regard the shipment weight capacity for train services, where Constraints (7) are used for the short-leg train services of Y-shuttles, so they include the weight of shipments assigned to the corresponding long-leg train service.

Constraints (8) and (9) define the proportion of transported shipment units on a connection and the difference between this proportion and the desired lower limit on this proportion. If the proportion is above the desired lower limit, the difference will be set to 0 because of Constraints (13), such that there will not be a reward for being above the limit. Variable $k_a, a \in \tilde{A}$ can be eliminated by substituting (8) into (9), however, we present the model in this way for the sake of readability.

Finally, Constraints (10)-(13) define the domains of the variables used in this MILP problem.

Implementation

The exact formulation of the shipment loading problem has been defined for a general shipment loading problem on one set of connections. It can also be used for a subset of the shipments, for example the unassigned shipments at some point while performing a construction heuristic. Furthermore, it can be used for a subset of the wagons, which is how we will use the MILP problem in the Greedy Construction Heuristic when evaluating the profit increase of a certain wagon group. In any case, only one wagon should be used per wagon group, for the reason given in Section 4.2.2.

When implementing the MILP problem with subsets of the sets given in Table 2, some slight changes may need to be made such that the acceptance penalty is ensured in the correct way. For example, if connection $a \in \tilde{A}$ already has some shipments assigned to it, their total size should be added to k_a . It is also possible to apply some simple linear transformations to d_a and $q_a, a \in \tilde{A}$ to be able to use the original constraints. Furthermore, the shipment weight capacity $h_t, t \in \tilde{T}$ may need to be calculated differently, for example by subtracting the weight of shipments which have already been assigned to the relevant train services in addition to subtracting the total wagon weight from the original weight capacity.

4.2.4 Relax & Fix

Using a commercial solver for the MILP problem for shipment loading will provide us with the optimal solution. However, depending on the size of the problem, the amount of time required to find this solution may be too large. Alternatively, we can iteratively relax part of the binary variables by allowing them to take on values between 0 and 1 to find a solution in less time. A common technique which can be applied to this problem is a construction heuristic known as the Relax & Fix algorithm. In the first iteration, most of the binary variables are relaxed, but some are restricted to 0 and 1 like in the original MILP problem. After solving the resulting problem, part of the restricted variables are fixed to their value in the solution. In the next iteration, some of the variables which were relaxed are now also restricted to 0 and 1, and another batch of restricted variables will be fixed after solving the altered problem, and so on. This algorithm is especially useful when the variables have a natural ordering, for example when they can be ordered chronologically. The details of the Relax & Fix algorithm are given by Pochet and Wolsey (2006).

A way to relax and fix the variables in the problem of shipment loading would be to sort them by the arrival time of the train services, such that the shipment loading can be determined chronologically, starting from the first train services. For example, in the first iteration, all variables regarding train services which arrive after the end of the first day of the planning horizon are relaxed. In other words, only those variables which regard train services which arrive on the first day of the planning horizon are restricted to 0 and 1. Because of the way some of the train services are related through Y-shuttles, we will sort long-leg train services by the arrival time of their related short-leg train services. In our example, consider a Y-shuttle leaving at noon on the first day of the planning horizon, of which the small composition is decoupled before midnight. If the long-leg train service, which uses this small composition, arrives at its destination after midnight, we restrict the variables corresponding to this train service instead of relaxing them. This allows for the complete Y-shuttle to be planned in one iteration.

We introduce two parameters to formally define the Relax & Fix algorithm for the problem of shipment loading. Parameter δ represents the duration of the restriction period in hours and parameter $\epsilon \leq \delta$ represents the duration of the fixation period in hours. Furthermore, let the arrival time of train service $t \in \tilde{T}$ in hours after the start of the planning horizon be given by m_t . In case t is a long-leg train service, m_t will be replaced by the arrival time of the corresponding short-leg train service. The sets of variables to be restricted and fixed in iteration i , depending on parameters δ and ϵ can now be defined as follows:

$$X^{restrict}(\delta, \epsilon, i) = \left\{ x_{stwp}, s \in \tilde{S}, t \in \tilde{T}_s, w \in \tilde{W}_t, p \in P : (i-1)\epsilon < m_t \leq (i-1)\epsilon + \delta \right\} \quad (14)$$

$$Y^{restrict}(\delta, \epsilon, i) = \left\{ y_{twl}, t \in \tilde{T}, w \in \tilde{W}_t, l \in L_w : (i-1)\epsilon < m_t \leq (i-1)\epsilon + \delta \right\} \quad (15)$$

$$X^{fix}(\epsilon, i) = \left\{ x_{stwp}, s \in \tilde{S}, t \in \tilde{T}_s, w \in \tilde{W}_t, p \in P : (i-1)\epsilon < m_t \leq i\epsilon \right\} \quad (16)$$

$$Y^{fix}(\epsilon, i) = \left\{ y_{twl}, t \in \tilde{T}, w \in \tilde{W}_t, l \in L_w : (i-1)\epsilon < m_t \leq i\epsilon \right\} \quad (17)$$

The last iteration of the Relax & Fix algorithm is the first iteration in which there are no more

relaxed variables. The solution found by this final problem is the final solution found by the algorithm.

4.2.5 Ignore & Fix

An alternative to relaxing some of the binary variables is to completely ignore them, by excluding them from the problem or by restricting them to zero. Compared to the Relax & Fix algorithm, this will lead to smaller problems to solve, which will likely save time. However, since some of the variables are completely ignored, instead of being relaxed, the solution quality may deteriorate. The extent of this effect is expected to be dependent on the duration of the fixation period. For a small fixation period (in terms of the restriction period), there is a large amount of time between the variables to be fixed and the variables which are ignored, or rather between the periods to which these variables correspond. In that case, the solution quality of this alternative algorithm will likely not deteriorate as much compared to the Relax & Fix algorithm as when using a larger fixation period. We will refer to this alternative solution method as the Ignore & Fix algorithm. The same parameters δ and ϵ and the same sets of variables (14)-(17) will be used to formally define the algorithm.

4.3 Genetic Algorithm

Given a set of solutions, a Genetic Algorithm can be used to find a better solution in the following manner. The set of solutions, referred to as the *population*, is manipulated by combining pairs of solutions through a so-called *crossover* operation, and by changing or *mutating* solutions. The idea is analogous to evolution of a population of a species in biology. Each solution represents the chromosome of an individual. Crossover represents reproduction, because two *parent* solutions are combined to form a *child* solution with characteristics of the parent solutions. Mutation regards slight changes in the solutions. Each iteration of the algorithm produces one new *generation* of solutions. This is done in such a way that each next generation is likely to contain a better solution than the best solution of the previous generation. Solutions are evaluated by their *fitness value*, which will be the objective value or total profit in our case.

The Genetic Algorithm is a general framework, which can be implemented for many different problems. The success of the algorithm depends on the different choices to be made. Firstly, an initial population needs to be constructed. The solutions in the initial population do not need to be very good, although better solutions might reduce the computation time for the remainder of the Genetic Algorithm. A randomized construction heuristic generally is a good option for generating the initial population, since it is fast and results in many different solutions. Secondly, the crossover operation needs to be defined. Two solutions need to be combined such that the resulting solution is feasible and is of reasonable quality. The child solution may be worse than the parent solutions, such that the population gets more diverse. The solution could then improve through mutation. However, if the child solutions are much worse, the crossover operation may not be successful in the context of the Genetic Algorithm. Thirdly, a mutation operation has to be defined. Now, one

solution needs to be changed, such that the result is a different, feasible solution. The mutated solution may be worse, but in general, a mutation operation which often leads to a better solution will perform better for the Genetic Algorithm. Lastly, there should be a way to create a new population given the current population. Here, there are many options. It is common to define the *elite population* as the set of solutions among the current population with the highest fitness value. Typically, the elite population is copied to the new population. Furthermore, solutions are generated by selecting solutions from the current population and applying crossover and mutation operations to them.

4.3.1 Initial Population

We will construct the initial population using the Randomized Greedy Construction Heuristic, which has been defined in Section 4.1.1. As our solution method for shipment loading, we will use Greedy Shipment Loading. We expect this to result in relatively good solutions in a reasonable amount of time. The size of the initial population will be denoted by ζ .

4.3.2 Crossover

There is no straight-forward way to define a crossover operation to combine two solutions, because of the characteristics of the problem. For example, taking some of the wagon groups from one solution, and some of the wagon groups of another solution, is likely infeasible because of the wagon capacity. Even if we could find such sets of wagon groups without violating the wagon capacity of the child solution, the shipment loading would not be taken care of. Some shipments might be assigned to two wagon groups, one from each parent solution. An option would be to completely perform the shipment loading from scratch. However, this is undesirable for two reasons. Firstly, it would cause the crossover operation to take a relatively large amount of time. Secondly, the shipment loading of the parents determines the quality of the parent solutions, which is something we would like to be reflected in the child solution.

We define a crossover operation which overcomes these problems by selecting wagon groups in a greedy way and by checking the feasibility in between. The shipment loading is largely taken from the parent solutions, but is also extended after the wagon groups have been selected.

To start, two parent solutions should be selected. For a regular Genetic Algorithm, these are selected at random from the entire population. For a Biased Genetic Algorithm, as described by Gonçalves and Resende (2011), one of them is selected from the elite population, while the other is selected from the remaining part of the population.

Consider the list of all wagon groups from both parent solutions. These wagon groups are assigned to a family of connections in the parent solution. For each of them, determine the profit increase which would result from assigning the wagon group to the same family of connections in the child. This profit increase consists of the revenue increase from the shipments which have been assigned to the wagon group in the parent solution, the decrease in acceptance penalty, the

(possibly negative) decrease in shunting costs, the increase in long train bonus and the increase in wagon costs.

Out of all wagon groups, the wagon group with the highest profit increase is selected to be assigned to the child solution. All shipments which were assigned to this wagon group in the parent solution are also assigned. The wagon group is removed from the list of wagon groups.

Until the list of wagon groups is empty, we keep repeating the selection and assignment of wagon groups. From the second iteration onward, the profit increase does not need to be recalculated for most of the wagon groups. This is only necessary for wagon groups which are assigned to the same family as the last assigned wagon group. In particular, the revenue increase may have changed for two reasons. Firstly, some shipments have been assigned, so these should not be considered anymore when calculating the revenue increase. Secondly, for some train services, the shipment loading may not be feasible anymore due to weight restrictions. In such cases, we sort the shipments by increasing revenue and remove as much shipments as needed to satisfy the weight capacity.

In the same way as during the first iteration, the best wagon group is selected. In addition to checking the shipment loading, we should now also check the wagon group itself for feasibility. The wagon group is feasible if (1) there are enough wagons of the correct type still available in the child solution and (2) the length limit for compositions will not be exceeded after assigning the wagon group. There is no need to check if the wagon type is allowed for the family. This should already be the case, since the wagon group has been used in a feasible parent solution.

If the best wagon group is infeasible, it is removed from the list of wagon groups, and the next best wagon group is selected, until there is a feasible wagon group. If its profit increase is positive, the wagon group is assigned to the child solution. Shipments which have not been assigned before are assigned as long as they do not violate the weight restrictions, as explained before.

At some point, there will be no wagon groups left, or all remaining wagon groups will have negative profit increase, and the selection and assignment of wagon groups should be terminated. It is not guaranteed that adding a wagon group with negative profit increase does not lead to an increase in profit over multiple iterations. In fact, adding a wagon group with negative profit increase might cause the profit increase of another wagon group to become positive again because of the long train bonus. However, this will likely not occur very often, and because of the greedy nature of this crossover operation, we will not be adding any wagon groups with negative profit increase.

Among many of the wagon groups, there may be room for more shipments, given the fact that shipments which had already been assigned have been removed from the wagon groups. Therefore, we perform shipment loading on the assigned wagon groups, given the current shipment loading, for example using the Greedy Shipment Loading algorithm as defined in Algorithm 5. This is expected to increase the total profit of the child solution to a value similar to the total profit of the parent solutions.

Depending on the problem instance and the number of wagon requirements, there is a possibility that the child solution will not satisfy the wagon requirements. To avoid this, we first assign some

wagon groups to satisfy the wagon requirements before assigning wagon groups as described before. This is done in almost the same way as for regular wagon groups, so by adding the wagon groups according to their profit increase, with the following exceptions. Firstly, the wagon groups from both parents which are considered are only those which would help satisfy the wagon requirements when assigned to the child. Secondly, we do not only remove wagon groups which have already been assigned to the child or which have become infeasible after the last assignment, but also wagon groups which would not help satisfy the wagon requirements anymore. For example, if a family needs one wagon group of a certain wagon type for the family's small compositions, all wagon groups of the same wagon type for small compositions can be removed from the list of wagon groups once such a wagon group has been assigned. Lastly, we do not check whether the profit increase of a wagon group is positive, because even if all remaining wagon groups have negative profit increase, we still need to satisfy the wagon requirements. Once the list of wagon groups is empty, the wagon requirements have been satisfied, and the remainder of the algorithm can be performed as described before, using all wagon groups, except for those which were already assigned to the child to satisfy the wagon requirements.

Algorithm 6 provides an overview of the crossover operation. It uses Algorithm 9, in which a partial child solution is created which satisfies the wagon requirements. Since it is so similar to the main part of the crossover operation, this algorithm can be found in Appendix B.

Algorithm 6: GeneticAlgorithmCrossover(PARENT_1, PARENT_2)

Result: Child solution

CHILD = **Algorithm 9**(PARENT_1, PARENT_2); A child solution satisfying the wagon requirements;

WAGON_GROUPS = set of assigned wagon groups from PARENT_1 and PARENT_2 which have not been assigned to CHILD yet;

while WAGON_GROUPS is not empty **do**

for each WAGON_GROUP in WAGON_GROUPS **do**

 Remove shipments which have already been assigned to CHILD;

 For each train service for which the weight capacity would be exceeded, remove enough shipments to prevent this, starting from the shipment with the smallest revenue;

 Determine the profit increase (consisting of revenue increase, decrease in acceptance penalty, decrease in shunting costs, increase in long train bonus and increase in wagon costs) when assigning WAGON_GROUP and the corresponding shipments to CHILD;

Note: As soon as one wagon group has been assigned to CHILD, the profit increase only needs to be recalculated for wagon groups which are assigned to the same family as the last assigned wagon group;

end

BEST_WAGON_GROUP = the wagon group in WAGON_GROUPS with the highest profit increase;

if BEST_WAGON_GROUP is feasible for CHILD and BEST_WAGON_GROUP has a positive profit increase **then**

 Assign BEST_WAGON_GROUP to child;

 Assign shipments to BEST_WAGON_GROUP which were also assigned to BEST_WAGON_GROUP in the parent solution unless they have already been assigned to another wagon group in CHILD or they would violate the weight restrictions;

end

Remove BEST_WAGON_GROUP from WAGON_GROUPS;

end

FINAL_WAGON_GROUPS = set of assigned wagon groups from CHILD;

for each WAGON_GROUP in FINAL_WAGON_GROUPS **do**

 Assign (more) shipments to WAGON_GROUP (e.g. using **Algorithm 5**);

end

return CHILD

4.3.3 Mutation

A solution is mutated by randomly selecting one wagon group and assigning these wagons to a randomly selected other family. Shipments which were assigned to the wagons of the wagon group are now not transported anymore. On the other hand, the shipment loading can be performed for the newly assigned wagons, with shipments of the other family. When performing such a mutation, there are some details which need to be taken care of in order for the solution to remain feasible. Firstly, there should be enough wagons for all compositions of the other family. For this, not only the wagons from the randomly selected wagon group can be used, but also the wagons of the same type which were not assigned to any family. If there are not enough wagons, another wagon group and family should be selected. Secondly, the wagon requirements should remain satisfied after reassigning the wagon group. Thirdly, the assignment should be feasible for the small or large compositions of the family. Whether the wagons will be assigned to the large or to the small compositions of the family may be determined by random selection. Furthermore, to determine the feasibility of the assignment, (1) the wagon type should be allowed to be used for the family (in small or large compositions), (2) the length limit for the small or large compositions will not be exceeded after adding a wagon to each of them and (3) the weight limit is not exceeded after adding a wagon to each of them for any of the train services using these compositions. If the assignment is infeasible, another wagon group and family should be selected. Finally, when a feasible assignment has been found and performed, the shipment loading should be performed, which can be done in a greedy way as in Algorithm 5 or in some other way, as explained in Section 4.2. Algorithm 7 provides the details for mutating a solution.

When mutating a solution, the total profit will likely change, because the assigned shipments change. Furthermore, the wagon costs, acceptance penalty, shunting costs and long train bonus may change. Therefore, the total profit of the mutated solution may be higher or lower than the total profit of the original solution. Because of the randomness of the mutations, some mutations will likely lead to better solutions, unless the original solution is already optimal or locally optimal.

4.3.4 Population Construction

The first step in constructing the new population for the next iteration is to define an elite population. The elite individuals are our most promising solutions, which will all be taken to the next iteration. We will construct the elite population simply by taking the $\eta^{elite} \cdot \zeta$ best solutions from the current population, based on their total profit.

The other part of the new population is constructed by applying crossover and mutation operations on the current population. There will be $\eta^{crossover} \cdot \zeta$ crossover solutions and $\eta^{mutation} \cdot \zeta$ mutated solutions. To ensure that the population size stays at ζ , it should hold that $\eta^{elite} + \eta^{crossover} + \eta^{mutation} = 1$. We will use a Biased Genetic Algorithm, so parents for crossover operations are selected at random from the elite population for the first parent, and from the remaining part of the population for the second parent. For the mutations, solutions will be selected at random from the elite population.

Algorithm 7: GeneticAlgorithmMutation(SOLUTION)

Result: Mutated solution

Randomly select an assigned wagon group from the SOLUTION;
RANDOM_WAGON_GROUP = the selected wagon group;
RANDOM_FAMILY = family to which RANDOM_WAGON_GROUP is assigned;
RANDOM_WAGON_TYPE = wagon type of RANDOM_WAGON_GROUP;
NUMBER_OF_AVAILABLE_WAGONS = number of compositions of RANDOM_FAMILY +
number of unassigned wagons of type RANDOM_WAGON_TYPE;

Randomly select a family OTHER_FAMILY with at most NUMBER_OF_AVAILABLE_WAGONS
compositions. Restart **Algorithm 7** if such a family does not exist;
Randomly select a binary variable OTHER_LARGE_COMPOSITIONS denoting whether
RANDOM_WAGON_GROUP should be reassigned to large (TRUE) or small (FALSE)
compositions;

if *RANDOM_WAGON_GROUP* is feasible for *OTHER_FAMILY* **then**
 Undo the original assignment of RANDOM_WAGON_GROUP including shipments;
 Change the family of RANDOM_WAGON_GROUP to OTHER_FAMILY;
 Assign RANDOM_WAGON_GROUP to SOLUTION;

 if *the wagon requirements are not satisfied* **then**
 Restart **Algorithm 7**;
 end

 Assign shipments to RANDOM_WAGON_GROUP (e.g. using **Algorithm 5**);
else
 Restart **Algorithm 7**;
end

return SOLUTION

If the new population contains solutions with the same amount of total profit, we will replace these solutions by other solutions, which we will obtain by applying more crossover operations on the previous population. Solutions with the same amount of total profit are likely to be the exact same solution or almost the same solution. Having duplicate solutions in the population is undesirable, since this effectively decreases the population size, which could in turn have a negative effect on the performance of the Genetic Algorithm.

There are several ways to determine when to terminate a Genetic Algorithm. For example, a certain time limit or number of iterations can be used. Furthermore, the Genetic Algorithm can be terminated once the total profit of the best solution has not increased (significantly) for a certain number of iterations, in which case the algorithm has converged. We will be using a time limit θ , but also report the number of iterations and the degree of convergence when evaluating the genetic

algorithm. Upon termination, the best solution of the final population will be used as the final solution found by the Genetic Algorithm.

Algorithm 8 contains an overview of the framework of the Genetic Algorithm used in this report.

Algorithm 8: GeneticAlgorithm($\alpha, \zeta, \eta^{elite}, \eta^{crossover}, \eta^{mutation}, \theta$)

Result: Solution

POPULATION = set of ζ solutions constructed by Randomized Greedy Construction Heuristic with threshold parameter α , using Greedy Shipment Loading;

while *TIME_LIMIT* θ is not reached **do**

 ELITE_POPULATION = set of best $\eta^{elite} \cdot \zeta$ solutions from POPULATION;

 CROSSOVER_POPULATION = set of $\eta^{crossover} \cdot \zeta$ solutions constructed through a crossover between a randomly selected parent from ELITE_POPULATION and a randomly selected parent from (POPULATION \setminus ELITE_POPULATION) using **Algorithm 6**;

 MUTATION_POPULATION = set of $\eta^{mutation} \cdot \zeta$ solutions constructed through a mutation of a randomly selected solution from ELITE_POPULATION using **Algorithm 7**;

 NEW_POPULATION = union of ELITE_POPULATION, CROSSOVER_POPULATION and MUTATION_POPULATION;

while *NEW_POPULATION* contains two solutions with the same total profit **do**

 Remove one solution from NEW_POPULATION which has the same total profit as another solution in NEW_POPULATION;

 Add a solution to NEW_POPULATION constructed through a crossover between a randomly selected parent from ELITE_POPULATION and a randomly selected parent from (POPULATION \setminus ELITE_POPULATION) using **Algorithm 6**;

end

 POPULATION = NEW_POPULATION;

 BEST_SOLUTION = solution from POPULATION with the highest total profit;

end

return *BEST_SOLUTION*

5 Results

The solution methods as defined in Section 4 have been implemented in Java 17.0.2. In this section, the results of applying them to the small and regular data sets as described in Section 2.7 will be presented. The device which has been used has an Intel Core i7-5600U processor and 16GB RAM. MILP problems have been solved using CPLEX 12.10.

5.1 Greedy Construction Heuristic

The Greedy Construction Heuristic has been implemented using the different shipment loading algorithms from Section 4.2. In general, the Greedy Construction Heuristic always finds a feasible solution to the problem. The quality of the solution and the time used to find it depends on the shipment loading algorithm. The different shipment loading algorithms will be evaluated in the following sections.

5.1.1 Greedy Shipment Loading

Performing shipment loading in a greedy way is one of the fastest solution methods considered in this paper. For the small instance, a solution is found in 0.8 seconds. For the regular instance, a solution is found in 523 seconds. In the Greedy Construction Heuristic, the wagon groups have been evaluated by their corresponding profit increase, as well as by their profit increase per wagon. Similarly, in the Greedy Shipment Loading, the shipments have been evaluated by their corresponding profit increase, as well as by their profit increase divided by shipment weight. This results in four different settings. Table 5 shows the results of these four settings for the small and regular instance. Considering the small instance, it is better to evaluate the shipments by profit increase than by the profit increase divided by weight. Here, the evaluation method for the wagon groups seems to be of less importance. For the regular instance, we do not observe this effect. Instead, the best setting is to evaluate wagon groups by profit increase per wagon and shipments by profit increase divided by weight.

These different results for both instances indicate that the best settings are instance-specific. Since our regular instance is the most realistic one, we will use the settings which result in the best solution for the regular instance. However, we conclude that trying other settings might be worth-while when applying this solution method to other instances.

Using the profit increase per wagon to evaluate wagon groups, and the profit increase divided by shipment weight to evaluate shipments, the solution obtained for the small instance has a total profit of 163,398. For the regular instance, this is 13,621,014. Table 6 breaks down the total profit into different components. Furthermore, some characteristics of the solutions are given. The percentage of shipments which are transported is reported, measured by number, shipping units, weight and revenue. Also, the number of connections for which an acceptance penalty was given is reported, expressed as a percentage of the total number of connections for which a penalty could be given. For the small instance, 100% of these connections are penalized, while for the regular instance, this percentage is also very high at 89.8%. One reason for this could be that the penalty is not high

enough for the algorithm to find solutions in which the penalty is avoided for more connections. Indeed, the total acceptance penalty only decreases the total profit by 1.1% for the small instance and by 0.1% for the regular instance. Another reason could be that there are no feasible solutions with acceptance percentages high enough for the penalty to be avoided, or that these solutions can not be found by this version of the Greedy Construction Heuristic. In this case, a higher penalty might not decrease the percentage of penalized connections. Further research will have to show whether changing the value of the penalty affects the solutions and, if so, in what way and for what reason.

Table 5: Results Greedy Construction Heuristic with Greedy Shipment Loading for Different Evaluation Methods for the Wagon Assignment (WA) and Shipment Loading (SL) Stages.

WA eval.		Profit	Profit per wagon	WA eval.		Profit	Profit per wagon
SL eval.	Profit			SL eval.	Profit		
	Profit	172,351	174,738		Profit	13,339,899	13,405,898
	Profit to weight	163,398	163,398		Profit to weight	13,268,537	13,621,014

(a) Results Small Instance.

(b) Results Regular Instance.

Table 6: Results Greedy Construction Heuristic with Greedy Shipment Loading.

Instance	Small	Regular
Total profit	163,398	13,621,014
Computation time (s)	0.8	523
Revenue	261,762	17,090,137
Long train bonus	4,018	63,450
Wagon costs	100,596	3,040,950
Shunting costs	0	479,000
Acceptance penalty	1,786	12,623
Acceptance - number	47.6%	65.3%
Acceptance - shipment units	43.4%	61.8%
Acceptance - weight	38.0%	59.1%
Acceptance - revenue	44.1%	72.9%
Penalized connections	100.0%	89.8%

5.1.2 Sorted Shipment Loading

Using the Sorted Shipment Loading for the shipment loading stage of the problem also provides a solution in a small amount of time. For the small instance, a solution is found in 0.6 seconds, while this is 265 seconds for the regular instance. Compared to the Greedy Shipment Loading, the computation time has especially decreased for the regular instance. This indicates that the computation time of Sorted Shipment Loading increases less when the instance size increases. It would therefore be more useful to apply Sorted Shipment Loading to even larger instances instead

of Greedy Shipment Loading in case the computation time starts to become problematic.

For Sorted Shipment Loading, we consider two selection methods for the shipments. The first method is to use the difficulty parameter $\beta_s, s \in S$, which is a measure for the number of possibilities to assign the shipment. The second method is to replace β_s by $\beta_s/(r_s/w_s), s \in S$, in order to incorporate the revenue-weight ratio when selecting shipments for assignment. Combined with the two evaluation methods for the wagon groups, this gives us a total of four different settings.

Table 7 shows the results of applying the algorithm to the small and regular instance using these settings. For the small instance, there does not seem to be much difference among the results of the different settings. For the regular instance, it is better to use β_s than to divide it by the revenue-weight ratio. Furthermore, it is better to evaluate the wagon groups by profit increase per wagon than by profit increase, similar to the Greedy Shipment Loading results. With the same reasoning as before, we will base our choice of a setting on the regular instance, such that we choose to evaluate wagon groups by their profit increase per wagon and use the original difficulty parameter $\beta_s, s \in S$ for selecting the shipments.

Table 7: Results Greedy Construction Heuristic with Sorted Shipment Loading for Different Evaluation Methods for the Wagon Assignment (WA) and for Different Difficulty Parameters.

WA eval.			WA eval.		
Difficulty parameter	Profit	Profit per wagon	Difficulty parameter	Profit	Profit per wagon
β_s	164,784	165,865	β_s	12,033,782	12,224,063
$\beta_s/(r_s/w_s)$	165,175	165,737	$\beta_s/(r_s/w_s)$	11,794,922	11,808,910

(a) Results Small Instance. (b) Results Regular Instance.

Table 8: Results Greedy Construction Heuristic with Sorted Shipment Loading.

Instance	Small	Regular
Total profit	165,865	12,224,063
Computation time (s)	0.6	265
Revenue	264,192	15,476,913
Long train bonus	4,018	33,928
Wagon costs	100,596	2,718,250
Shunting costs	0	556,000
Acceptance penalty	1,748	12,528
Acceptance - number	47.9%	63.3%
Acceptance - shipment units	45.1%	61.5%
Acceptance - weight	42.8%	61.7%
Acceptance - revenue	44.5%	66.0%
Penalized connections	100.0%	89.8%

Table 8 contains the detailed results of this setting for both instances. For the small instance, the

total profit of 165,865 is not much different from the results of Greedy Shipment Loading, although there also are settings for which Greedy Shipment Loading yields solutions which are up to 5% better. For the regular instance, Sorted Shipment Loading is clearly worse than Greedy Shipment Loading with a decrease in total profit of 9% to 12,224,063. Again, we observe that the number of penalized connections is very high.

We conclude that the Greedy Construction Heuristic with Sorted Shipment Loading performs relatively well, especially given the short computation time and the fact that shipment revenue is not considered in the shipment loading stage. However, in most cases it would be more useful to apply Greedy Shipment Loading instead of Sorted Shipment Loading, since the results are often better.

5.1.3 Exact Formulation

By solving the MILP problems to perform the shipment loading, we always obtain the optimal selection of shipments to transport on a candidate wagon when assigning it to a candidate family, given the current shipment loading. This allows us to select the best candidate in the Greedy Construction Heuristic during each iteration. As expected, this also leads to a better total profit, compared to the Greedy Shipment Loading. For the small instance, the total profit increases by 10.7% to 180,864. For the regular instance, it takes over 24 hours to perform the solution methods. We therefore apply a time limit to the MILP problems, which is the maximum amount of time to spend on solving one MILP problem. Furthermore, we perform the Greedy Shipment Loading before solving the MILP problems, such that we can use it as a warm start for the solver. Using a time limit of 1 second leads to a total profit of 13,684,274, which corresponds to an increase of 0.5% compared to Greedy Shipment Loading.

Using the exact formulation provides us with better solutions. However, the computation time also significantly increases compared to the Greedy Shipment Loading, from 0.8 seconds to 209 seconds for the small instance and from about 0.15 hours to 3.83 hours for the regular instance.

For the small instance, we can also impose a time limit. Using the same time limit of 1 second reduces the computation time by 50%, while the total profit does not change much. In fact, the total profit increases to 181,433. This can be explained by the observation that the time limit is only reached for 23 out of 488 model calls. In those cases, the shipment loading is likely to be not much worse than the optimal shipment loading. For the regular instance, it would make sense to try a higher time limit in order to find better results, especially since the time limit was reached for 8,360 out of 21,819 model calls (38.3%). However, using a time limit of 10 or 20 seconds only yields worse results with a total profit of 13,297,433 and 13,471,588, respectively, while the computation time increases to 15.5 and 23.1 hours, respectively. As expected, the proportion of model calls which were terminated due to reaching the time limit decreased to 3,749 out of 19,453 (19.3%) and 2585 out of 19,316 (13.4%), respectively. Even though this means that the optimal solution to the shipment loading problem was found in more cases, this did not lead to a better solution to the general problem. This can be explained by the randomness in the algorithm. When there is no

unique wagon group with the highest corresponding profit increase per wagon, one of the best wagon groups is selected at random. Choosing one or the other could make a difference for the possibilities later in the algorithm. This also shows why using a lower time limit could actually result in a better solution. During some iterations, the time limit causes a wagon group to be selected which is not the best option among all wagon groups, however, this leaves room for other wagon groups to be selected later in the algorithm, which all in all creates a better solution. This process is more or less unavoidable, given the greediness of the algorithm.

Table 9 provides the detailed results of this solution method. Regarding the penalized connections, we again observe high percentages. In Section 5.1.1, we suggested two possible reasons for this.

Table 9: Results Greedy Construction Heuristic with the Exact Formulation for Shipment Loading.

Instance	Small		Regular			
	Time limit (s)	1	<i>unlimited</i>	1	10	20
Total profit	181,433	180,864	13,684,274	13,297,443	13,471,588	
Computation time (s)	105	209	13,790	55,871	83,230	
Revenue	283,553	282,997	17,023,547	16,536,905	16,710,935	
Long train bonus	867	867	54,154	30,221	28,262	
Wagon costs	93,300	93,300	2,939,250	2,717,500	2,707,050	
Shunting costs	8,000	8,000	441,500	539,000	547,500	
Acceptance penalty	1,687	1,700	12,677	13,183	13,059	
Acceptance - number	50.3%	50.4%	64.1%	60.6%	61.3%	
Acceptance - shipment units	46.7%	46.7%	61.1%	58.7%	59.3%	
Acceptance - weight	43.6%	43.6%	60.2%	58.9%	60.0%	
Acceptance - revenue	47.7%	47.7%	72.6%	70.5%	71.2%	
Penalized connections	100.0%	100.0%	89.8%	89.8%	89.8%	

5.1.4 Relax & Fix

The Ignore & Fix algorithm as defined in Section 4.2.5 is based on the Relax & Fix algorithm. The goal of the Relax & Fix algorithm is to find a relatively good solution to the shipment loading problem in a small amount of time, by solving several easier MILP problems instead of one more difficult problem. However, relaxing part of the binary variables, such that they can take on values between 0 and 1 as well, in fact causes CPLEX to spend more time on solving the problem. For all problems, the same solution is found compared to the original problem, in which no binary variables are relaxed. So even though the binary variables are not restricted to 0 and 1, they do not take on other values than 0 and 1. Therefore, the Relax & Fix algorithm is not useful for solving the shipment loading problem. The same solution would be found in every iteration of the algorithm, and the total computation time to perform the algorithm would be much more than the computation time of simply solving the MILP problem.

5.1.5 Ignore & Fix

The alternative solution method as defined in Section 4.2.5 is the Ignore & Fix algorithm. Instead of relaxing variables, some variables are removed from the problem for some of the iterations of the algorithm. The Ignore & Fix algorithm has been implemented and evaluated on the small data set. Table 10 shows the results of this evaluation. Parameters δ and ϵ , representing the number of hours to optimize over and to fix, respectively, have been set to all possible combinations of a multiple of 24. The goal of the Ignore & Fix algorithm is to find a relatively good solution in a relatively small amount of time. Therefore, the total profit has been reported, as well as the computation time in parentheses when performing the Greedy Construction Heuristic with the Ignore & Fix algorithm as the solution method for shipment loading. The case with $\delta = \epsilon = 288$ is equivalent to the solving the original MILP problem. All cases with less computation time are given in bold. As expected, these are mainly cases for which the fixation period is of almost the same size as the optimization period, such that the number of iterations is kept low. Most of these cases have a total profit similar to or higher than the original problem. In fact, almost all parameter settings lead to a total profit which is higher than the total profit of the original problem. Furthermore, there does not seem to be a pattern in the total profit values. This indicates that the dependency between different time periods within the shipment loading problem may not be very high. However, we must be careful when drawing conclusions, because we are comparing the total profit of the Greedy Construction Heuristic, instead of the individual shipment loading problems. Since the parameter settings do not

Table 10: Analysis Ignore & Fix Algorithm for the Small Instance.

Total profit is reported for different parameter settings (computation time (s) in parentheses).

$\delta \backslash \epsilon$	24	48	72	96	120	144	168	192	216	240	264	288
24	180,520 (327)											
48	180,699 (312)	180,699 (179)										
72	181,169 (515)	180,994 (252)	180,827 (154)									
96	181,416 (483)	181,115 (265)	181,600 (184)	181,416 (131)								
120	181,158 (541)	179,387 (376)	178,831 (181)	181,122 (192)	181,122 (183)							
144	181,829 (719)	181,115 (397)	181,896 (248)	181,158 (201)	178,751 (215)	181,416 (156)						
168	179,862 (988)	181,295 (949)	181,125 (388)	180,857 (292)	181,158 (246)	181,463 (168)	181,416 (167)					
192	179,862 (833)	181,896 (477)	181,158 (360)	181,158 (293)	180,857 (300)	181,427 (242)	181,125 (185)	181,427 (179)				
216	181,474 (1037)	181,427 (822)	181,158 (522)	181,342 (667)	181,342 (517)	181,158 (283)	180,442 (380)	181,427 (279)	180,569 (390)			
240	181,205 (681)	181,158 (520)	180,736 (397)	179,581 (360)	181,158 (348)	179,862 (285)	179,860 (342)	181,004 (318)	181,360 (484)	180,962 (280)		
264	181,342 (700)	181,830 (835)	181,416 (425)	181,463 (449)	181,427 (323)	181,158 (402)	179,862 (225)	181,611 (304)	181,205 (407)	181,610 (259)	181,205 (230)	
288												180,864 (209)

seem to affect the total profit significantly, we will choose the parameter setting based on the computation time. The smallest amount of computation time was obtained by setting $\delta = \epsilon = 96$. This resulted in a decrease in computation time of 37% compared to solving the MILP problems at once.

For the regular instance, an analysis has been performed using equal values for δ and ϵ , since the analysis of the small instance shows that using $\epsilon < \delta$ is not necessary to obtain good solutions. The Greedy Construction Heuristic has been performed using the Ignore & Fix algorithm with $\delta = \epsilon = 24, 48, \dots, 1200$ and a time limit of 10 seconds per MILP problem. Each time, the heuristic was terminated after the same number of shipment loading problems, such that the computation times can be compared. The computation times are given in Figure 3 and show a clear trend. The smaller the optimization and fixation period, the more time is needed to solve the shipment loading problems. This is different from the small instance, in which many settings with δ and ϵ much smaller than their maximum value resulted in smaller computation times than for the original problem. However, this can be explained by the use of the time limit. If the time limit is reached in many cases, even for the smaller problems of the Ignore & Fix algorithm, applying Ignore & Fix will simply increase the number of MILP problems, without significantly reducing the computation time. This leads to higher computation times for smaller values of δ and ϵ , because of the larger number of iterations.

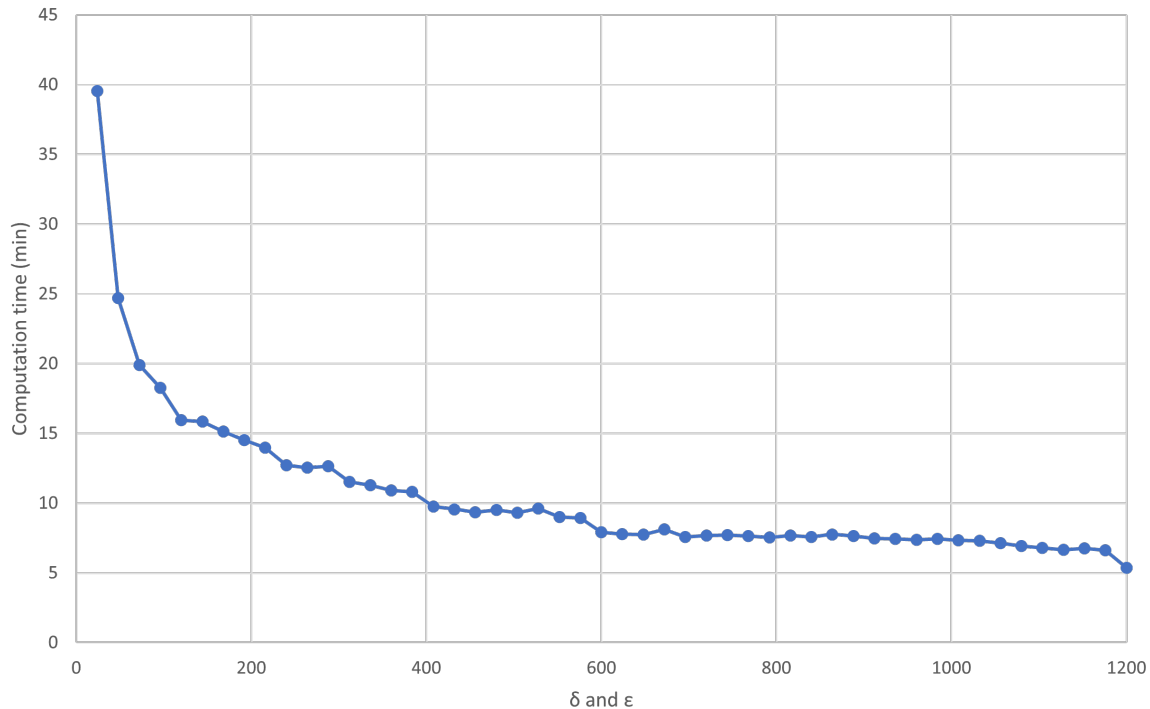


Figure 3: Computation Time of the First Part of the Greedy Construction Heuristic with the Ignore & Fix Algorithm for Different Values of δ and ϵ (Regular Instance).

To be able to find solutions using the Ignore & Fix algorithm in a smaller amount of time, we would need to imply an even lower time limit to the individual MILP problems. This will likely not be beneficial for the quality of the solutions. We conclude that the Ignore & Fix algorithm is successful in decreasing the computation time when not using any time limits. Unfortunately, this means we cannot use it in practice for instances of realistic size.

5.2 Genetic Algorithm

In this section, the results of the Genetic Algorithm will be evaluated. Firstly, we will describe the Randomized Greedy Construction Heuristic, which will be used to generate the initial populations for the Genetic Algorithm. Next, the crossover and mutation operations will be evaluated when performed on an initial population of 100 solutions. Lastly, the results of the Genetic Algorithm itself will be given.

5.2.1 Initial Population

The Randomized Greedy Construction Heuristic has been performed with Greedy Shipment Loading and different values for threshold parameter α . For each value of α between 60% and 100%, with increments of 2 percentage points, a population of size $\zeta = 100$ has been created. The results for the small instance are summarized by Figure 4. For each setting, the total profit of the worst and best solutions are given, as well as the average total profit and the 10% and 90% percentiles.

For values of α lower than 68%, the algorithm often finds solutions with a total profit below 155,000. Starting from $\alpha = 68\%$, the results of the algorithm seem to be stable, with almost all solutions having a total profit between 155,000 and 165,000. When increasing α from 68% to 100%, the average total profit increases from 159,390 to 163,409 and the difference between the worst and best solution decreases from 9,345 to 1,221. The fact that there are still different solutions when using $\alpha = 100\%$ can be explained as follows. When selecting the wagon group to be assigned, there sometimes is no unique best option. In that case, one of the best options is selected at random. This affects the next options, which causes the algorithm to find a different solution each time.

Starting from $\alpha = 92\%$, there is not much difference anymore in the total profit of the solutions. In fact, we observe many solutions with the same amount of total profit. This will likely not be beneficial for the performance of the Genetic Algorithm, since the initial population will not be very diverse. We conclude that a value between 68% and 90% is a good choice for α when applying the Random Greedy Construction Heuristic with Greedy Shipment Loading. It will provide us with a diverse initial population with relatively good solutions.

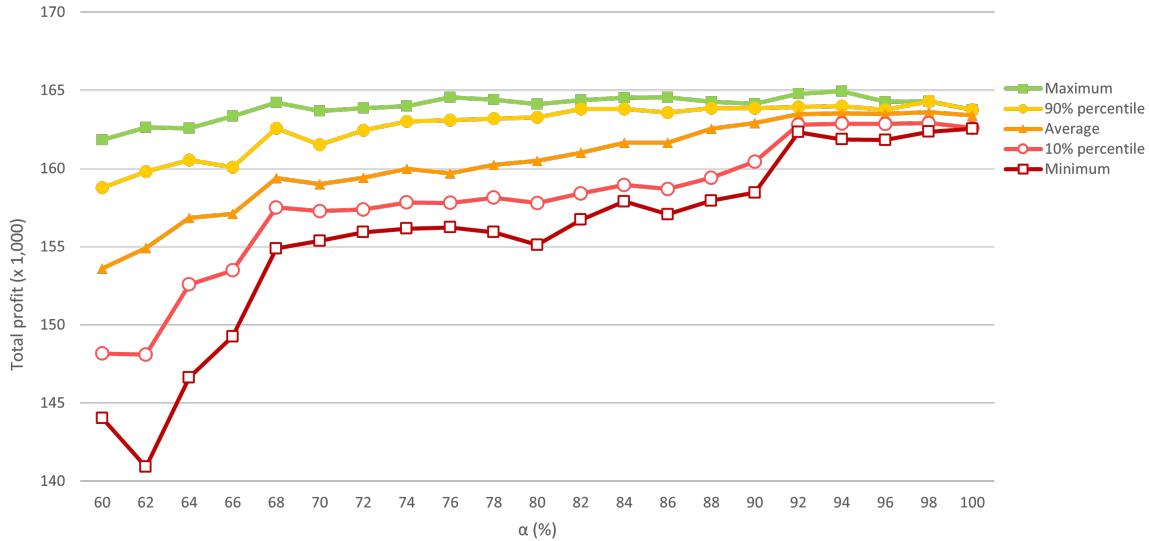


Figure 4: Results of 100 tries of the Randomized Greedy Construction Heuristic with Greedy Shipment Loading for different values of α .

Table 11 summarizes the initial populations obtained with threshold parameter $\alpha = 90\%$, both for the small and regular instance.

Table 11: Characteristics Initial Population Generated by Randomized Greedy Construction Heuristic with Greedy Shipment Loading.

	Instance	Small	Regular
Average total profit		163,010	13,488,581
Standard deviation		921	69,719
Lowest total profit		160,161	13,269,140
Highest total profit		164,274	13,598,980

5.2.2 Crossover

For the crossover operation, the Greedy Shipment Loading algorithm has been used to apply to the child solution once the wagon groups have been assigned. This keeps the computation time of a crossover operation relatively small. When performing 100 crossover operations on random pairs of solutions from the initial population, the total profit of the child solution is 100.5% of the average of the total profit of the parent solutions, on average, for the small instance. For the regular instance, this is 100.3%. This shows that this crossover operation produces solutions roughly of the same quality as the parent solution. Furthermore, the crossover operation provides solutions which are better than the best parent solution in 51 and 60 cases for the small and regular instance, respectively. The average computation time for one crossover operation is 0.13 seconds for the small instance and 7.52 seconds for the regular instance.

5.2.3 Mutation

For the mutation operation, there is not much shipment loading to be performed. We can therefore use the exact formulation to solve the shipment loading problems without spending too much time on the mutation operations. When performing a mutation operation on all 100 solution from the initial population, the average computation time is 0.28 seconds for the small instance and 2.72 seconds for the regular instance. This results in mutated solutions of which the total profit is 0.75% and 0.44% lower than before mutation, on average, for the small and regular instance, respectively. Despite this decrease in total profit, there are some cases in which the mutation is an improvement. This is the case for 46 solutions of the small instance and for 3 solutions of the regular instance.

It is expected that the proportion of mutations which lead to an improved solution will decrease as the total profit, or fitness of the solutions, increases, up to the point where we are at a local optimum with respect to this mutation operation. For a local optimum, there will be no more mutations with a higher total profit. However, the mutated solutions diversify the population. Combined with the crossover operations, they could still lead to better solutions, especially since the crossover operations often result in solutions which are better than the best parent solution.

5.2.4 Genetic Algorithm

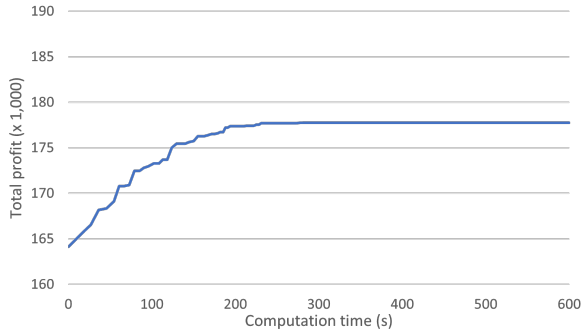
The Genetic Algorithm as defined in Section 4.3 has been performed on the small and regular instance. The parameter settings have been evaluated for the small instance, because it uses the least amount of time. When using the same settings for generating the initial population for different runs of the Genetic Algorithm, the same solutions have been used in order to rule out the effect of the quality of the initial population on the final solution.

As our benchmark setting, we will use a population size $\zeta = 100$ and an elite population with a size of 50% of the population ($\eta^{elite} = 0.5$). For each new population, 25% will be created using crossover operations between randomly selected parents from the full population ($\eta^{crossover} = 0.25$) and the remaining 25% will be created using mutations of the elite population ($\eta^{mutation} = 0.25$). The algorithm will be terminated after $\theta = 600$ seconds. Figure 5a shows the results of this setting. A total of 155 iterations could be performed before the time limit was reached. The total profit obtained by the Genetic Algorithm is 177,735. This result is obtained after 277 seconds, meaning that the algorithm has converged. A better solution could potentially still be found when letting the algorithm run for even more than 600 seconds, however, given the trend of the best solution during the algorithm, this seems unlikely.

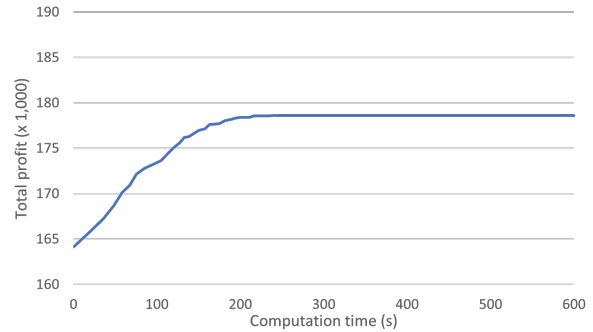
Smaller Elite Population

When using a smaller elite population, the best solutions will still be taken to the next iteration, but there will be more crossover and mutation operations per iteration. Furthermore, the mutation operations will be performed on better solutions, on average. Using $\eta^{elite} = 0.2$, $\eta^{crossover} = 0.4$ and $\eta^{mutation} = 0.4$, a solution with a total profit of 178,624 is found after 240 seconds. This is an

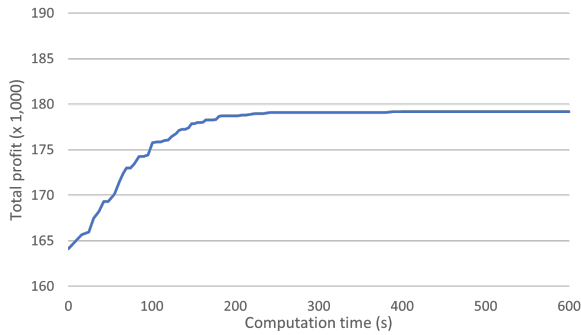
improvement of 0.5% compared to the benchmark setting. Figure 5b shows the trend of the best solution while executing the algorithm.



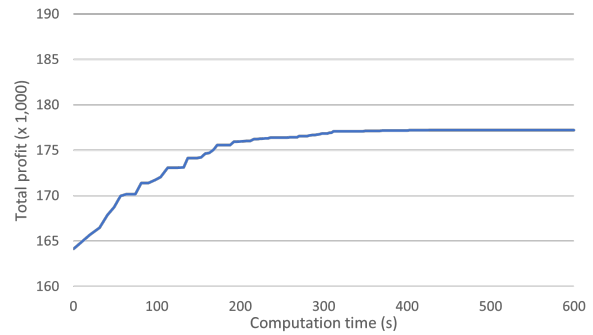
(a) $\eta^{elite} = 0.5, \eta^{crossover} = 0.25, \eta^{mutation} = 0.25$.



(b) $\eta^{elite} = 0.2, \eta^{crossover} = 0.4, \eta^{mutation} = 0.4$.



(c) $\eta^{elite} = 0.5, \eta^{crossover} = 0.4, \eta^{mutation} = 0.1$.



(d) $\eta^{elite} = 0.5, \eta^{crossover} = 0.1, \eta^{mutation} = 0.4$.

Figure 5: Results of Genetic Algorithm for Different Settings (Small Instance).

More Crossover Operations

When increasing $\eta^{crossover}$ to 0.4, while decreasing $\eta^{mutation}$ to 0.1, we obtain a solution with a total profit of 179,174, which corresponds to an improvement of 0.8%. This solution is found after 388 seconds. However, as can be seen in Figure 5c, the trend of the best solution is very similar to the case of the benchmark setting. Most progress is made during the first 250 seconds.

More Mutation Operations

When increasing the number of mutation operations, while decreasing the number of crossover operations, using $\eta^{crossover} = 0.1$ and $\eta^{mutation} = 0.4$, the result is worse than for the benchmark setting. Not only is the final solution slightly worse with a total profit of 177,223 (-0.3%), the algorithm also converges more slowly. Figure 5d shows this.

Biased Genetic Algorithm

In a Biased Genetic Algorithm, the parents on which the crossover operations are performed are not selected completely at random. Instead, the first parent is selected at random from the elite

population, while the second parent is selected at random from the remaining part of the solution. With this change, we obtain a solution with a total profit of 179,352 (+0.9%). Figure 6a shows the trend of the best solution, which is similar to the case of the benchmark setting.

Larger Population

With a population twice the size ($\zeta = 200$), we obtain the best solution found so far using the Genetic Algorithm. After 600 seconds, the total profit of the final solution equals 180,508 (+1.6%). The algorithm does not seem to have converged yet. Indeed, a solution with a total profit of 180,643 is reached after 2,407 seconds. After that, no better solution is found for at least another 1,000 seconds. It seems that using a larger population results in a better solution, while also taking more time. Furthermore, it takes another 80 seconds to generate more initial solutions, which has not been included in Figure 6b.

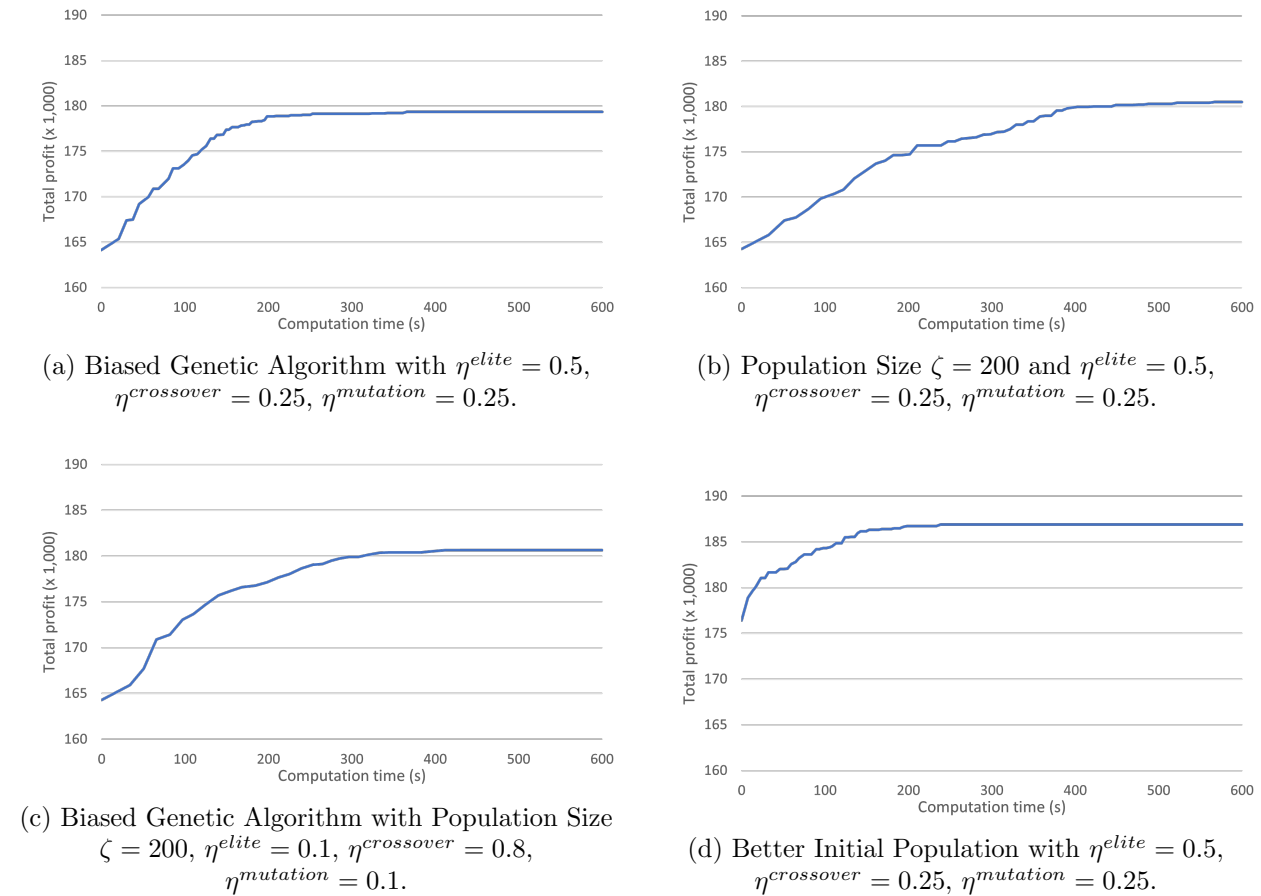


Figure 6: Results of Genetic Algorithm for Different Settings (Small Instance) - Continued.

Combined Strategy

Combining the conclusions of the parameter analysis, we propose the following strategy for the Genetic Algorithm. The biased variant of the Genetic Algorithm should be performed on a population

of $\zeta = 200$ solutions, using mainly crossover operations ($\eta^{crossover} = 0.8$) and some mutation operations ($\eta^{mutation} = 0.1$). The elite population consists of the best 10% of the population ($\eta^{elite} = 0.1$). This results in a final solution with a total profit of 180,640. This solution is reached in 412 seconds and no improvement is found for at least another 3,000 seconds, meaning that the algorithm has converged relatively quickly.

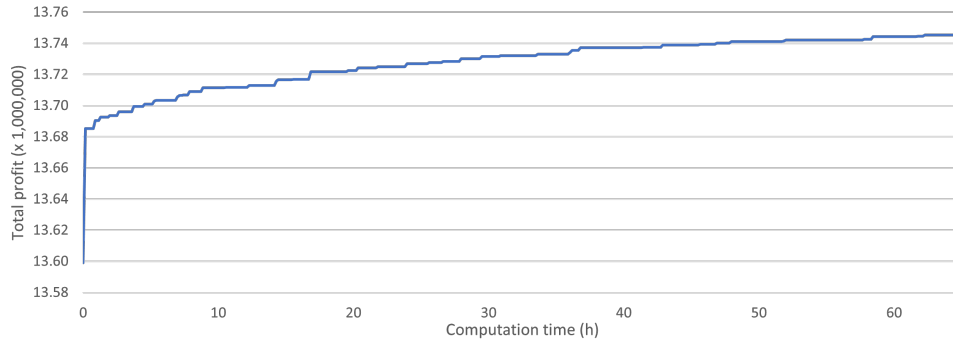
When comparing the Genetic Algorithm to the Greedy Construction Heuristic with the exact formulation for shipment loading, we observe that the first does not outperform the latter, although the results only differ by 0.1%. Furthermore, the Greedy Construction Heuristic could be performed in only 105 seconds, while the Genetic Algorithm needs 160 seconds to generate the initial populations and another 412 seconds for the remainder of the algorithm.

Better Initial Population

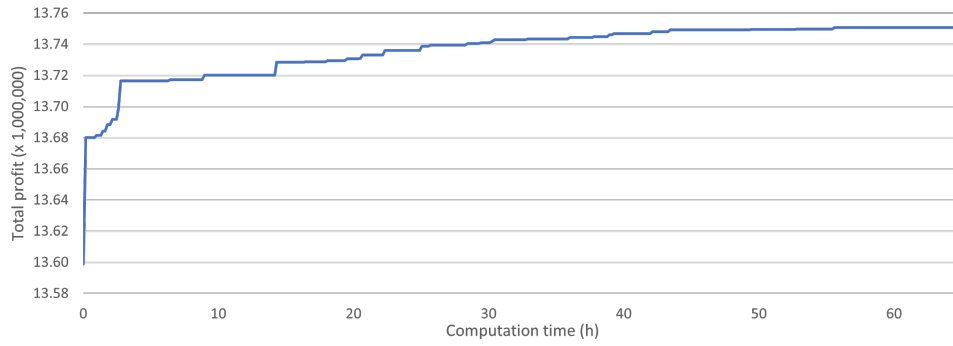
So far, the initial population has been created using the profit increase divided by shipment weight to evaluate the shipments during the Greedy Shipment Loading algorithm. However, we have a better option for the small instance, which has been shown in Section 5.1.1. When simply using the profit increase to evaluate the shipments, our initial population consists of solutions with a total profit between 170,731 and 176,419. Performing the Genetic Algorithm on this population with the benchmark settings results in a final solution with a total profit of 186,876. This shows the importance of a good initial population. In this case, the improvement in total profit with respect to the best solution among the initial solutions is 10,457, while this was 13,597 for the benchmark case. So even though the initial solutions were better, the improvement made by the Genetic Algorithm is still relatively large. This shows an advantage and a disadvantage of the Genetic Algorithm in the context of this problem. On the one hand, it does not succeed in finding near-optimal solutions when the initial population is of low quality. On the other hand, it still manages to find better solutions when the initial population is of higher quality.

Regular Instance

For the regular instance, the Genetic Algorithm also has been performed using the benchmark setting and a time limit of 65 hours ($\theta = 234,000$). Figure 7a shows the total profit of the best solution during the execution of the algorithm. The computation time to create the initial population, which equals 14.5 hours, has not been included in the graph. The algorithm does not converge within the time limit. The total profit of the best solution of the initial population equals 13,598,980. During the first iteration, this already gets improved by 0.6% to 13,685,233. After that, the total profit of the best solution increases to 13,745,279 in 65 hours, which corresponds to an improvement of the initial best solution of 1.1%. During this time, 419 iterations could be performed with an average of 557 seconds per iteration.



(a) $\eta^{elite} = 0.5, \eta^{crossover} = 0.25, \eta^{mutation} = 0.25$.



(b) Biased Genetic Algorithm with $\eta^{elite} = 0.1, \eta^{crossover} = 0.8, \eta^{mutation} = 0.1$.

Figure 7: Results of Genetic Algorithm for Different Settings (Regular Instance).

The regular instance has also been solved using the combined strategy, which worked well for the small instance. The only exception is the population size, which has been kept at $\zeta = 100$, because of the large amount of computation time needed to generate the initial population. As expected, this setting performs better than the benchmark setting. The total profit of the final solution of the benchmark setting is already reached after 38.9 hours. After 65 hours, the best solution has a total profit of 13,751,461, which is an improvement of 1.1% compared to the initial best solution. This solution is also 0.5% better than the solution found by the Greedy Construction Heuristic using the exact formulation for shipment loading.

6 Conclusion

6.1 Main Findings

The aim of this paper is to solve a specific two-stage assignment problem in rail freight transportation, which has been described in detail in Section 2. In short, the problem consists of assigning wagons to certain parts of the network, referred to as families, in the first stage. In the second stage, shipments need to be assigned to the wagons. The goal of this second stage is to determine the total profit which can be obtained by the wagon assignment.

Several solution methods have been proposed to solve this problem. To evaluate these solution methods, they have been implemented and applied to two data sets or problem instances, a small instance and an instance of regular size.

6.1.1 Greedy Construction Heuristic

The first approach to solve this problem is a Greedy Construction Heuristic. In itself, this is a straight-forward approach. However, the characteristics of the problem demand careful implementation. By defining the concept of a wagon group, wagons could be assigned in groups, such that the partial solution stays feasible during the execution of the heuristic. Furthermore, many calculations are repeated during multiple iterations. By keeping track of the changes resulting from assigning wagon groups, a large part of these calculations could be avoided, which reduces the computation time.

The Greedy Construction Heuristic needs to be extended with an algorithm for performing the shipment loading, which is the second stage of our two-stage problem, in order to determine the profit increase resulting from assigning a wagon group. Multiple approaches have been proposed for this. Firstly, two construction heuristics have been considered. The first is a greedy construction heuristic, referred to as Greedy Shipment Loading. The second is a construction heuristic based on Feo and González-Velarde (1995), in which shipments are assigned by decreasing difficulty to assign them, such that we can assign as many shipments as possible. We refer to this heuristic as Sorted Shipment Loading. Both heuristics cause the Greedy Construction Heuristic to find a solution to the overall problem in a relatively small amount of time of less than 1 second for the small instance and a couple of minutes for the regular instance. Sorted Shipment Loading is faster than Greedy Shipment Loading, but also generally leads to worse solutions, although this also depends on the parameter settings of both heuristics. Using the best parameter setting for Greedy Shipment Loading, we find a solution with a total profit of 13,621,014 for the regular instance.

Another approach for shipment loading is an exact formulation of the problem. For this, a Mixed Integer Linear Programming (MILP) problem has been defined, which can be solved to find optimal solutions to the shipment loading problem. However, this does not mean the Greedy Construction Heuristic will find the optimal solution to the overall problem when combined with the exact formulation for shipment loading. Using this approach causes the Greedy Construction Heuristic to spend 209 seconds finding a solution for the small instance and at least 23 hours for the

regular instance. This computation time could be reduced by setting a time limit for each MILP problem. Using a time limit of 1 second reduced the computation time by 50% to 105 seconds for the small instance, while the total profit of the solution did not become worse. For the regular instance, using a time limit of 1 second resulted in a better solution compared to using a time limit of 10 or 20 seconds. The solution has a total profit of 13,684,274, which is slightly better than the total profit of the solution found using Greedy Shipment Loading. However, the computation time of almost 4 hours is much larger.

In order to reduce the computation time, we proposed to use the exact formulation and apply a Relax & Fix algorithm to it. However, since the relaxed case of the problem did not result in solutions with non-integer values for the relaxed variables, and the time spent to find this solution was in fact higher than the time spent to solve the original problem, this approach could not be used to reduce the computation time. An alternative to the Relax & Fix algorithm has been proposed, referred to as the Ignore & Fix algorithm. In this alternative solution method, variables were simply ignored instead of relaxed. Since the dependency between different periods within the shipment loading problems seemed to be low, using the Ignore & Fix algorithm led to finding solutions with similar total profit as the exact formulation. For the small instance, a reduction in computation time could be achieved, however, this was not the case for the regular instance due to the use of a time limit. We concluded that the Ignore & Fix algorithm is successful when not using any time limits, but also that it cannot be used in practice for realistically sized instances.

6.1.2 Genetic Algorithm

The second approach which has been implemented and described in this paper is a Genetic Algorithm. To generate an initial population, the Greedy Construction Heuristic has been extended to a Randomized Greedy Construction Heuristic, such that the initial population consists of a diverse set of solutions. This randomized heuristic has a parameter which can be used to affect the diversity of the population.

The Genetic Algorithm has been defined for the problem considered in this paper, among others by defining the crossover and mutation operations. Both operations occasionally lead to improved solutions, which in turn causes the Genetic Algorithm to find better solutions than those of the initial population. The quality of the best solution found by the Genetic Algorithm and the rate of convergence depends on the settings of the algorithm. Using a strategy which combines some successful parameter settings, a solution with a total profit of 13,751,461 could be found, which is an improvement of 0.5% compared to the Greedy Construction Heuristic with the exact formulation for shipment loading. However, the time used to find this solution is 14.5 hours to generate the initial solution and an additional 65 hours of performing the remainder of the Genetic Algorithm.

6.2 Discussion and Further Research

For the small instance, the final solution of the Genetic Algorithm strongly depended on the quality of the initial population. We expect this might be the case for the regular instance as well. This

indicates a weak point in our Genetic Algorithm. It seems the Genetic Algorithm gets stuck in a local optimum with a total profit relatively close to the total profit of the best solution of the initial population. To improve our solution methods, new crossover and mutation operations could be defined, which diversify the solutions more than our current operations. Alternatively, the Greedy Construction Heuristic could be improved to start with a better initial population.

Regarding the size of the initial population of the Genetic Algorithm, there is a trade-off between computation time and solution quality. Increasing the population size leads to a better solution, but the algorithm also takes more time to converge. Furthermore, generating the initial population takes more time. It depends on the amount of available time which population size would be optimal. An idea for further research could be to investigate this trade-off.

The computation time is a limiting factor for our solution methods. Disregarding the computation time, the Genetic Algorithm is our best performing solution method. Even after 14.5 hours of generating an initial population and 65 hours of performing crossover and mutation operations to find better solutions, the algorithm still has not converged yet. Even though this gives us a total computation time of much more than 8 hours, the Genetic Algorithm still is a promising solution method, because it is highly parallelizable. Firstly, the generation of the initial population can be fully parallelized. Secondly, the remainder consists of many crossover and mutation operations. Crossover and mutation operations during the same iteration of the Genetic Algorithm can be performed independently from each other. When the number of operations is much larger than the number of cores, the total computation time can be reduced by a factor of almost the number of cores.

Even for the Greedy Construction Heuristic, computation time could become problematic for instances larger than our regular instance when using the exact formulation. Here, we can also speed up the algorithm by parallelizing it. During each iteration of the Greedy Construction Heuristic, many different candidates are evaluated, which can be done independently from each other. Furthermore, when performing the shipment loading, it can be solved for each of the minimum-size self-dependent connection sets by a different core. Many iterations may only have a few candidates which need to be re-evaluated, which means it may not be possible to use all cores at all times. However, we still expect parallelizing the Greedy Construction Heuristic will lead to a significant reduction in computation time for instances which otherwise take a lot of computation time.

In our two-stage problem, the second stage is used to determine the increase in profit which can be obtained by the wagon assignment. In reality, the results of the second stage will not be used for scheduling, because the actual wagon availability and shipment demand is only known just before or even on the day of operation. This leads to the idea of relaxing the second stage of our problem in order to speed up the shipment loading problem. For example, the loading scheme could be relaxed by not considering the weight capacity per position, but only for the full wagon. Alternatively, the shipment loading problem could be even more relaxed by only checking if a wagon has a position which could transport a certain shipment, without checking if this position is still available. To avoid assigning too many shipments to a wagon, a constraint could be added on the total number

of shipments or the total shipment weight per wagon.

Relaxing the shipment loading problem might very well lead to infeasible shipment loading. However, the corresponding profit increase could still be an indication of the actual profit increase which could be obtained by the wagon assignment. If the relaxed shipment loading still succeeds to a certain degree in ordering the wagon assignment candidates by profitability, the Greedy Construction Heuristic could still find good solutions. For further research, we suggest to investigate this idea and compare it to our solution methods regarding computation time and solution quality.

6.3 Management Advice

For the company facing the problem described in this paper, we suggest the following. Firstly, it is useful to implement the (Randomized) Greedy Construction Heuristic and Genetic Algorithm and parallelize them. This will make it possible to find a good solution in a relatively short amount of time using the Greedy Construction Heuristic. The Genetic Algorithm can be used for as long as desirable to find even better solutions.

Furthermore, we suggest the company investigates the optimal size of the population in the Genetic Algorithm for typical problem instances. Other parameters of the Genetic Algorithm may also be investigated, such that the Genetic Algorithm is tuned specifically for the company. This will save time, or yield a better solution within a certain time limit.

References

- Anghinolfi, D., Caballini, C., & Sacone, S. (2014). Optimizing train loading operations in innovative and automated container terminals. *IFAC Proceedings Volumes*, 47(3), 9581–9586.
- Bruns, F., & Knust, S. (2012). Optimized load planning of trains in intermodal transportation. *OR Spectrum*, 34(3), 511–533.
- Corry, P., & Kozan, E. (2008). Optimised loading patterns for intermodal trains. *OR Spectrum*, 30(4), 721–750.
- Ernst, A., & Pudney, P. (2001). Efficient loading of intermodal container trains. *Proceedings of the Mathematics-in-Industry Study Group*, 124–146.
- Feo, T. A., & González-Velarde, J. L. (1995). The intermodal trailer assignment problem. *Transportation Science*, 29(4), 330–341.
- Feo, T. A., & Resende, M. G. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2), 67–71.
- Gonçalves, J. F., & Resende, M. G. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5), 487–525.
- Gorman, M. F., Clarke, J.-P., Gharehgozli, A. H., Hewitt, M., de Koster, R., & Roy, D. (2014). State of the practice: A review of the application of or/ms in freight transportation. *Interfaces*, 44(6), 535–554.
- Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT Press.
- Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications*, 80(5), 8091–8126.
- Lai, Y.-C., Ouyang, Y., & Barkan, C. P. (2008). A rolling horizon model to optimize aerodynamic efficiency of intermodal freight trains with uncertainty. *Transportation Science*, 42(4), 466–477.
- Pochet, Y., & Wolsey, L. A. (2006). *Production planning by mixed integer programming* (Vol. 149). Springer.
- Upadhyay, A., Gu, W., & Bolia, N. (2017). Optimal loading of double-stack container trains. *Transportation Research Part E: Logistics and Transportation Review*, 107, 1–22.

7 Appendix

Appendix A.1 and Appendix A.2 contain some additional graphs to show the data set characteristics for the regular and small instance, respectively. Appendix B contains the details of the part of the crossover operation for the Genetic Algorithm in which the wagon requirements are satisfied.

A Extended Data Analysis

A.1 Regular instance

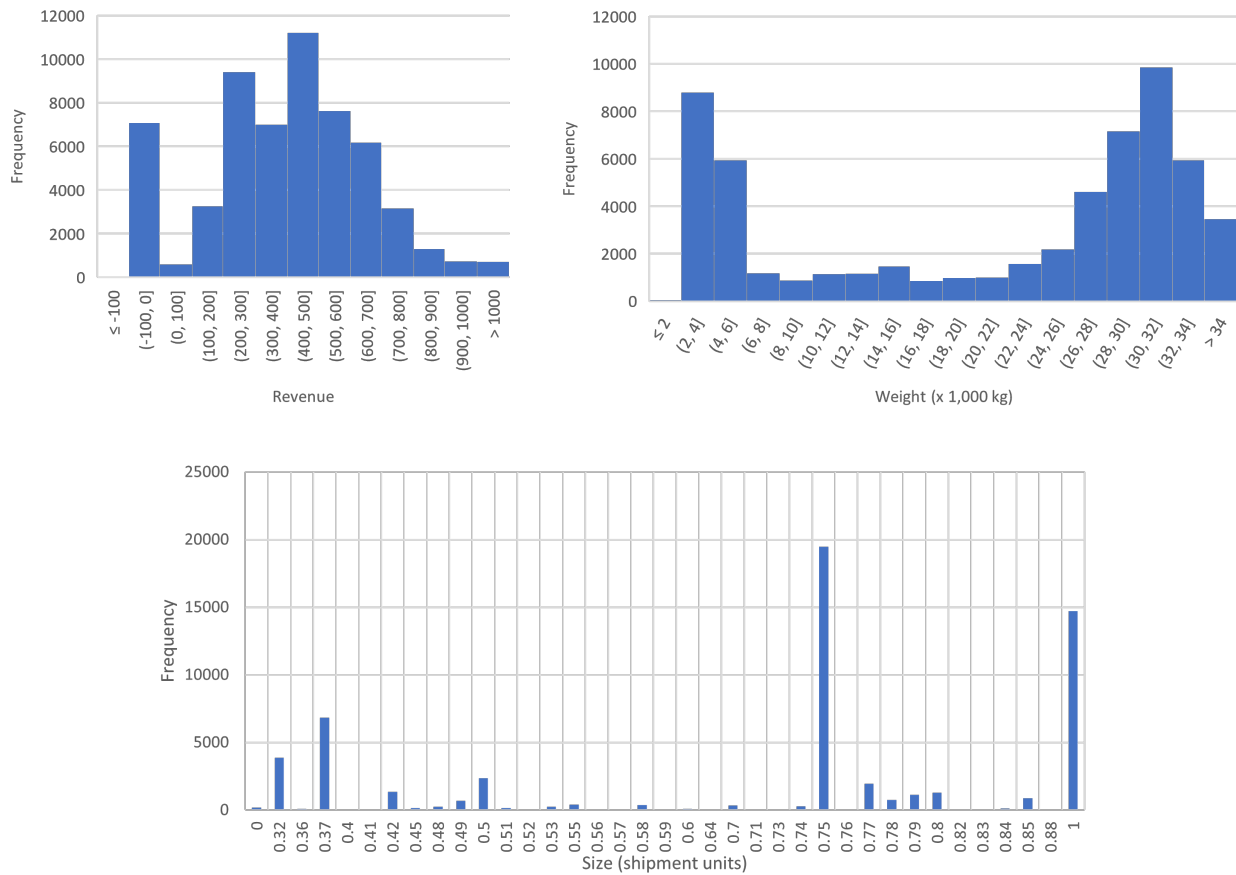


Figure 8: Histograms and Bar Chart of the Revenue, Weight and Size of Shipments (Regular Instance).

A.2 Small instance

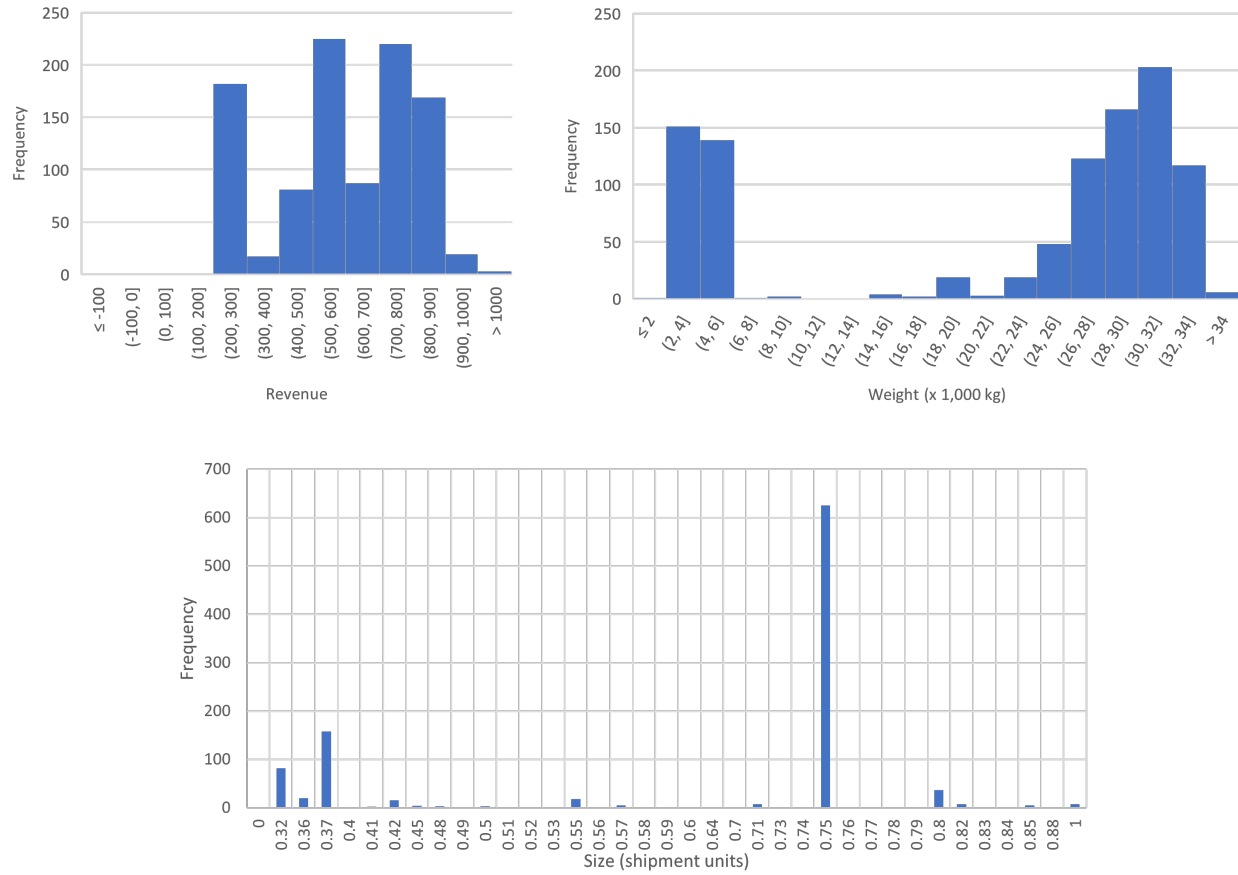


Figure 9: Histograms and Bar Chart of the Revenue, Weight and Size of Shipments (Small Instance).

B Details Genetic Algorithm Crossover Operation

Algorithm 9: GeneticAlgorithmCrossoverSatisfyWagonRequirements(PARENT_1, PARENT_2)

Result: Partial child solution in which the wagon requirements are satisfied

CHILD = empty solution;

WAGON_GROUPS = set of assigned wagon groups from PARENT_1 and PARENT_2 which would help satisfying the wagon requirements when assigning it to CHILD;

while *WAGON_GROUPS is not empty do*

for *each WAGON_GROUP in WAGON_GROUPS do*

 Determine the profit increase (consisting of revenue increase, decrease in acceptance penalty, decrease in shunting costs, increase in long train bonus and increase in wagon costs) when assigning WAGON_GROUP and the corresponding shipments to CHILD;

Note: As soon as a wagon group has been assigned to CHILD, the profit increase only needs to be recalculated for wagon groups which are assigned to the same family as the last assigned wagon group;

end

BEST_WAGON_GROUP = the wagon group in WAGON_GROUPS with the highest profit increase;

if *BEST_WAGON_GROUP is feasible for CHILD then*

 Assign BEST_WAGON_GROUP to child;

 Assign shipments to BEST_WAGON_GROUP which were also assigned to BEST_WAGON_GROUP in the parent solution unless they have already been assigned to another wagon group in CHILD or they would violate the weight restrictions;

end

Remove BEST_WAGON_GROUP from WAGON_GROUPS;

for *each WAGON_GROUP in WAGON_GROUPS do*

if *WAGON_GROUP does not help satisfying the wagon requirements when assigning it to CHILD then*

 Remove WAGON_GROUP from WAGON_GROUPS;

end

end

end

return *CHILD*
