Real-time Container Transport Planning with Decision Trees based on Offline Obtained Optimal Solutions

Bart van Riessen
vanriessen@ese.eur.nl
Econometric Institute
Erasmus School of Economics
Erasmus University Rotterdam

Rudy R. Negenborn
r.r.negenborn@tudelft.nl
Department of Marine and Transport Technology
Delft University of Technology

Rommert Dekker
rdekker@ese.eur.nl
Econometric Institute
Erasmus School of Economics
Erasmus University Rotterdam

EI 2016-14

March 2016

Abstract
Hinterland networks for container transportation require planning methods in order to increase efficiency and reliability of the inland road, rail and waterway connections. In this paper we aim to derive real-time decision rules for suitable allocations of containers to inland services by analysing the solution structure of a centralised optimisation method used offline on historic data. The decision tree can be used in a decision support system (DSS) for instantaneously allocating incoming orders to suitable services, without the need for continuous planning updates. Such a DSS is beneficial, as it is easy to implement in the current practice of container transportation. Earlier proposed centralised methods can find the optimal solution for the intermodal inland transportation problem in retrospect, but are not suitable when information becomes gradually available.
The main contributions are threefold: firstly, a structured method for creating decision trees from optimal solutions is proposed. Secondly, an innovative method is used for obtaining multiple equivalent optimal solutions to prevent overfitting of the decision tree. And finally, a structured analysis of three error types is presented for assessing the quality of an obtained tree. A case study illustrates the method's purpose by comparing the quality of the resulting plan with alternative methods.

Keywords: Intermodal planning, synchromodal planning, container transportation, decision support, decision trees.

Corresponding author
Bart van Riessen
T: (+31) 10 4081262
vanriessen@ese.eur.nl
Econometric Institute
Erasmus School of Economics
Erasmus University Rotterdam
P.O. Box 1738
3000 DR Rotterdam
The Netherlands

1. Introduction

Continuous growth of global container volumes puts increasing pressure on the inland road, water and rail connections, especially in developed countries with limited public support for infrastructural expansion. Simultaneously, shippers require more reliable inland connections because their supply chain demands for just-in-time delivery, and the environmental impact of the inland transportation is increasingly bound by restrictions from governments and from shippers themselves. In this study, we consider decision support for the planning of inland transportation. The problem is based on the situation of European Gateway Services (EGS), an inland transportation network providing transportation between the deepsea ports in Rotterdam and various *extended gates* in the European hinterland (Veenstra *et al.*, 2012, Van Riessen *et al*., 2014-b). Although the inland network has sufficient capacity in general, temporary congestion occurs frequently on all inland modes: road, water, and rail connections. Most inland transportation of containers is carried out by operators that are dedicated to specific modes. In the light of these developments, an integral approach for the routing and planning of all inland container transportation is vital. In this study we propose a real-time DSS for providing improved planning support. In particular, we propose to use decision trees as method in the DSS. Decision trees are a way to represent rules underlying data with hierarchical, sequential structures that recursively partition the data (Murthy, 2005). In our method, the routing decision is made per container, by applying the tree to the properties of the container transportation order (i.e. booking). Figure 1 gives an example of a decision tree supporting the routing decision for a container.

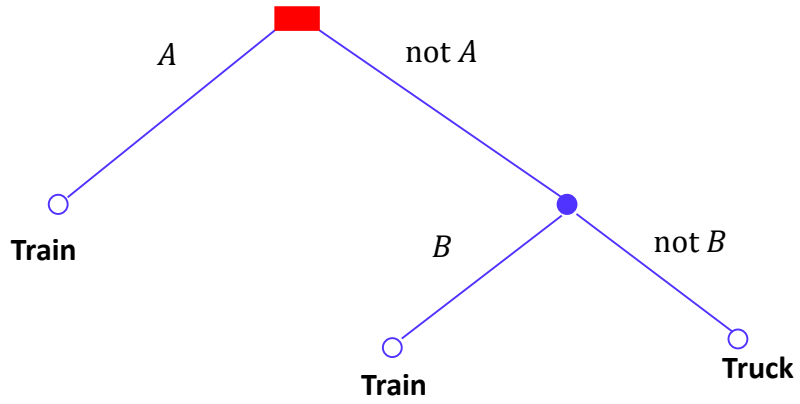


Figure 1 Example decision tree for deciding on the mode of transportation for a container.
The first decision is based on whether or not the container has property A. If not, a second decision follows, based on whether or not the container has property B.

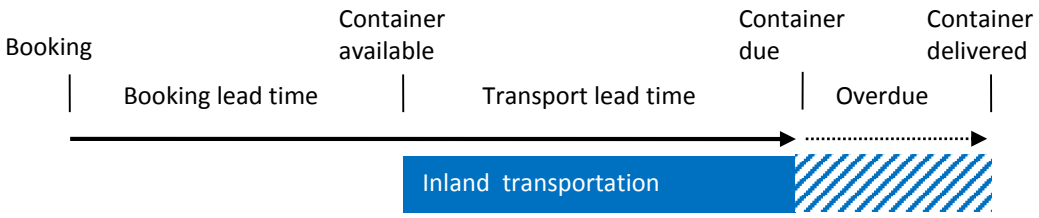


Figure 2 – Timeline of orders and inland transportation

1.1. Characteristic intermodal decision problem

For a particular corridor (i.e., the set of available transportation options between two locations), a set of inland services is available, characterized by the mode of transport (barge, rail or truck), cost per container, departure time, arrival time, and vehicle capacity (volume and weight). Naturally, the time between departure and arrival of a service depends on the mode's travel speed. Typically, speed and cost are high for trucking and low for barge transport. Volume capacity is high for barge transportation and low for a truck. The weight capacity for both barge and truck is mostly not restrictive. The mode train has intermediate levels for speed, cost and capacity, but typically has a restrictive weight capacity, especially in mountainous regions. In this setting, we consider scheduled barge and train services with fixed capacities, while trucks can be ordered at any time without limits. Generally, in a transportation setting, orders arrive sequentially at a transportation network planning department. Each order has several attributes, such as the client name, the number of containers for the order, the booking lead time, the transport lead time and the size and weight of each container in the order. The number of containers is measured in standard container sizes of Twenty feet Equivalent Units (TEU). The booking lead time is the time between the arrival of the order and the availability of the container; the transport lead time is the time between the availability of the containers and the due time at the destination (see Figure 2). The planner's goal is to transport the containers at the lowest possible total cost, ideally before the due time of the containers, but often it is allowed to deliver a little later, indicated by the overdue time. In practice, overdue delivery can sometimes be negotiated with customers, in our modelling overdue delivery is allowed at a penalty cost for the network operator. The characteristics of this decision problem do not change over time, giving rise to periodicity, e.g. weekly. Therefore, analytics on historic information can be used to find patterns and create a decision tree (DT). Subsequently, this decision tree can be used for decision support in future periods.

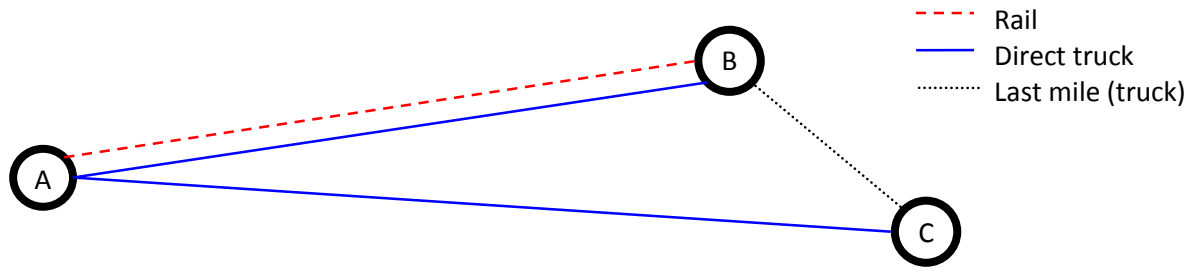


Figure 3 Characteristic decision problem

Figure 3 illustrates an example of the characteristic intermodal decision problem schematically. We consider 1 origin, A, and 2 destinations: B and C. Both destinations can be reached by using a truck (T) or a rail connection can be used (R), for transport from A to B or from A via B to C. In the case of using rail transportation for containers with destination C, last mile trucking from B to C is required. In general, rail is cheaper than truck, but has limited capacity. Trucking capacity is abundant, and considered unlimited for this study. The train has a limited capacity, denoted by K. The costs for the four transportation options are denoted as c_{dm}, where d denotes the destination and m denotes the used mode, $d \in \{\mathrm{B}, \mathrm{C}\}, m \in \{\mathrm{R}, \mathrm{T}\}$. E.g., c_{BR} denotes the costs for transporting one unit from A to destination B by rail. Typically, the goal is to maximise the utilisation rate of the lower priced rail mode for the highest yielding destination. Because of different transportation restrictions between container classes and differences in costs, this decision problem is not straightforward.

For the planner, who must make decisions instantaneously for incoming orders, the question is how many slots to reserve for each destination, i.e., use a booking limit of K_{B} slots for containers to destination B and K_{C} slots for destination C, adhering to $K_{\mathrm{B}} + K_{\mathrm{C}} = K$.

1.2. Real-time intermodal planning problem

Nowadays, in real cases (according to our experiences with EGS) often a greedy approach or first come, first serve (FCFS) approach is used for planning the container transportation. In case of a greedy approach, a

container transportation order is assigned to the cheapest feasible service at the time of order arrival, i.e. the cheapest service with free capacity that travels within the container's time restrictions. In an FCFS approach, a booking is assigned to the earliest available service. In both methods, an order is assigned instantaneously at the time of order arrival.

The problem addressed in this study is to allocate an incoming order immediately to the most suitable inland service, as part of the optimization of the entire corridor. Existing scientific methods for real-time decision making for planning of inland container transportation focus on finding cheapest or shortest paths per transport order (Ziliaskopoulos and Wardell, 2000; Ayed et al., 2011). More advanced methods for solving the online problem require real-time automated data processing and are less insightful to human planning operators (Nabais, 2013, Li *et al*., 2013). We propose a method for allocating orders to services based on optimal historical plans.

1.3. Proposed method for real-time decision support

In recent years, several studies have proposed optimisation methods for determining the optimal allocation of containers to all available inland transportation services, considering capacity, costs, lead times and emissions. The proposed methods are suitable for solving the *offline* planning problem, in which an optimal network plan is created for a batch of transportation orders collectively. In intermodal networks, such as the network of European Gateway Services, the implementation of a centralised offline approach is difficult for various reasons:

- Real-time decisions: The nature of the inland transport logistics requires a real-time approach, in which a customer can get immediate feedback on the selected mode, route, and most importantly, the estimated time of arrival. Consequently, updates in the planning of inland transportation have large influences on the subsequent production processes, possibly resulting in an undesired cascade of changes in earlier determined plans. An improved solution method must support real-time planning decisions, without continuous planning updates.
- Incomplete information: The operation of transportation systems is often not centralised, but depends on multiple cooperating decision makers, e.g. a logistics service provider and a transportation operator. The supply chain of container logistics lacks information integration (van der Horst and de Langen, 2008). Capacity and/or demand information for future periods is often not fully available. However, existing centralised optimisation methods depend strongly on complete information from integrated and automated processes, both for terminals, as for other parts of the supply chain. They lack the flexibility to deal with incomplete information. A method must therefore be able to provide decision support even with incomplete information.
- Human-aware decision making: In relation to the previous aspect, planning operators manually gather information ad hoc, such as real-time information on capacity and delays. Delays are common as the workloads in container terminals have a stochastic nature and are distributed unevenly in time (Murthy, 2005). For this, direct communication between manual operators is essential (Douma, 2008). A transparent approach is required that allows the human operators to include manually obtained information in the decision process. We consider this the white box poperty of the desired solution method.

In this paper, we propose a general method for a real-time DSS that addresses the aforementioned issues, by meeting three main requirements. Firstly, the proposed method allows real-time decision support for allocating incoming transport orders directly to available inland services, resulting in a stable solution and instant feedback to the customer without the necessity of continuous planning updates. Secondly, the method does not need complete information to provide decision support: The decision tree obtained by our method can be applied in real-time in daily practice without an automated decision system. Thirdly, the proposed method provides decision support in a white box system. The human planner responsible for a central network planning can check available capacity on a proposed service manually and include that in the decision process. In a case study, we will exemplify how the proposed method adheres to these three characteristics while resulting in a solution closer to the optimum than the current practice, although some misclassifications are

still to be expected. It is assumed that the problem is repetitive over time: the weekly service schedule and the distribution of demand are considered as constant in time.

1.4. Approach

The model we propose is based on an analysis of the solution space of an offline optimization model, and translates the offline model's optimal solutions to a decision tree: a white box representation of decision rules. It will therefore more easily be accepted for use in daily practice. The method must operate well under various circumstances and therefore multiple historic order arrival patterns are used. The proposed approach is valid for a decision process with repetitive patterns, for which offline optima of historic data can be determined. The quality of the real-time decision support will depend on the problem structure. We introduce a method to assess that a priori (i.e. before applying it in real-time). In the remainder of this paper, we focus on the problem of intermodal hinterland transportation. In Section 6 we will provide insight on how to apply the proposed method to other cases as well.

The approach is pictured schematically in Figure 4. First, the data of historic demands are assembled, i.e. the historic container transportation bookings. Secondly, the optimal transportation plan is determined using the recently proposed *linear container allocation problem with time restrictions* (LCAT) (Riessen *et al*., 2014-a). The resulting optimal solutions for historical demand periods provide the baseline for real-time decision support and are used in the third step to find properties of an effective planning of a container considering the uncertainty in the demand. The relations between container properties and the planned mode and route for that container in the optimal solution are determined. For this, we use a method of supervised learning for creating a decision tree based the allocations in optimal plans of the training sets. The supervised learning algorithm creates decision rules for the allocation of a container to a suitable service based on the container and order properties, such as the time of availability, the transportation lead time, and container mass. Subsequently, the set of rules can be used in a real-time setting as DSS: for each incoming order the DSS will provide a human planner with a set of suitable services in an understandable way.

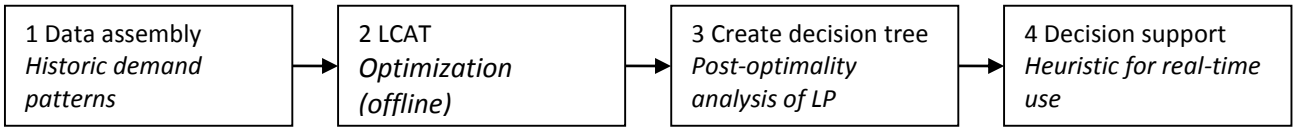


Figure 4 – Proposed methodology for real-time decision support based on optimal solutions obtained offline

It is expected that the performance of the proposed method is comparable to a low-level assignment strategy such as *first-come, first serve* for cases with orders that are entirely randomly distributed across the selected input features. If, however, historic information contains specific demand patterns, identifiable from the selected features, our method will capture those patterns without further detailed analysis. In this article we extend our previous work on real-time decision support based on offline optimisation (Van Riessen et al., 2014-c).

1.5. Contribution

The main contributions of the method are threefold. First, we propose a structured method for creating decision trees from optimal solutions. Secondly, we use a method for obtaining multiple equivalent optimal solutions to prevent overfitting of the decision tree. And finally, we develop a structured analysis of three error types for assessing the quality of an obtained tree and the expected error compared to the theoretical optimum. The remainder of this paper is organised as follows. Section 2 provides an overview of relevant literature on real-time decision support and decision trees. Subsequently, Section 3 gives a formal description of the proposed method for obtaining and using the decision tree. In this section, we also compare the results of the proposed method with an optimal strategy for the characteristic intermodal problem. In Section 4 methods for estimating the performance of the algorithm in general are described. In Section 5, the method is applied in a more general case study of an intermodal hinterland transportation corridor, i.e. in this case, multiple modes are available for transporting containers between two locations. This case study is motivated

by a practical case in the port of Rotterdam. Section 6 summarises the findings of the study and provides an outlook on future research.

2. Literature overview of real-time decision-making (using offline models)

2.1. Optimisation models for decision problems

In the literature, several methods have been proposed to deal with uncertainty in a combinatorial optimization problem. Gal and Davis (1979) describe parametric analysis in linear programming (LP); Jenkins (1990) describes parametric analysis in (mixed) integer linear programming (MIP). His purpose is to study the sensitivity of the problem's objective for certain parameters. Greenburg (1998) gives an extensive overview of the more general concept of post solution analysis for both LP and MIP. He mentions the concept of *stability analysis:* finding the set of parameters for which a given solution remains optimal. However, Wallace (1996) illustrates that sensitivity analysis is only appropriate for deterministic problems and not suitable to support decision making under uncertainty. An alternative for parametric analysis of a solution is to develop a robust planning, that is optimal considering the uncertainty of all parameters. Since 2000, a large number of studies have been published on Robust Optimization, see Bertsimas *et al.* (2011) for an overview. Both parametric analysis and robust optimization assume a fixed set of decision variables. However, in our case, the number of decisions to take depends on the number of transportation orders. Branley *et al.* (1997) describe the concept of post-evaluation analysis. The purpose of such an analysis is to describe the decision surface, representing the value of the objective of the problem for a set of decision variables. Such a surface gives insight in the effect of certain decisions, but creating such a surface is computationally very demanding as all optimal solutions must be found. This problem belongs to the class of #P-complete problems and is as least as difficult as NP-complete problems (Valiant, 1979). In post-optimization analysis, only a set of optimal solutions is studied, i.e., a subset of solutions with equal optimal objective value (Venkat *et al.*, 2003). Such post-optimization analysis is often applied in multi-objective decision-making.

The intermodal decision problem can be considered as a specific type of multi-knapsack problem. The multi-knapsack problem is a well-known problem in literature, e.g. Rinnooy Kan *et al.* (1993), Pak and Dekker (2004) and Van Hentenryck and Bent (2009). The former two do not address decision support for real-time decisions. Van Hentenryck and Bent (2009) address a category of problems for which the uncertainty does not depend on the decision-making. Typically, the time-critical nature of the decisions in such a problem require online decision making. Online anticipatory algorithms (OAA) are used to solve this type of problems, by combining online algorithms and optimisation models. For making a decision at a given point in time a distribution of future events is assumed. If a predictive model is not available, sampling of historic data can be used. With OAA, an optimal policy that prescribes the required action in any state is not necessary, instead, only the decision provided the current situation and the expected future events are considered using optimization models. An OAA can be used in an iterative manner, alternating between decision-making and incorporating new observations. However, in our proposed method we do not yet consider iterative learning. The distributions of inputs are sampled and learned independently from historic data of the underlying decision process (Van Hentenryck and Bent, 2009, Van Hentenryck *et al.*, 2010). Verwer *et al.* (2014) propose a method that can be seen as the inverse from our proposal: Their technique encodes optimisation models from learned decision trees, which can be used in auction settings. The method we propose uses a new approach for post-optimization analysis. As no predictive model is available, generally, we propose a learning algorithm that translates historic optimal solutions of the offline problem into real-time decision support.

2.2. Decision Support Systems in container transportation

Steiger (1998) states that the purpose of a decision support system is to provide understanding to the decision maker. Apart from the solution to a model, this also requires insight in the model and model outcome. Giboney (2015) shows that the representation of a knowledge systems is crucial for user acceptance. In relation to container logistics, existing literature mostly focuses on modelling and finding solutions. Several DSSs have been developed that focus on the operational problem of container transhipment in container terminals (e.g. Murty *et al.*, 2005; Ngai *et al.*, 2007; Ursavas, 2014). Ursavas (2014) also mentions some works

in literature that have proposed DSSs for container transportation problems. Some studies consider that demand is known in advance, such as Shen and Khoong (1995), who developed a mathematical model for planning the distribution of empty containers, suggesting specific options to the decision maker. Bandeira *et al.* (2009) propose a computer-based heuristic for solving a network flow model of both empty and full containers. After each time step, newly arrived future demand is considered. Janssen *et al.* (2004) describe an automated planning program for container trucking. Several studies considered a stochastic model using the expected future demand, e.g. Cheung and Chen (1998). However, because of the problem complexity and consequential computation time, this type of problems is impractical for application in real-time, as well as that no theoretical distribution of the future demand is available. More advanced methods for solving the online problem include Nabais (2013), who uses model predictive control to achieve a required modal split, and Li *et al*. (2013), who developed a sequential linear programming approach. All methods use computerised systems for providing proposed solutions to the decision maker.

Table 1 provides an overview of mentioned literature for decision support systems in real-time container transportation planning. For each of them is indicated to what extent they meet the 3 requirements for our problem as introduced in Section 1.3. From Table 1 we can see that all methods support decisions in real time to various extent. Several can deal with incomplete demand information, e.g. by using distributions of future demand. Most techniques require complete information on capacity, except for Ziliaskopoulos and Wardell's method, in which capacity is not considered. Also, none of the existing methods have a white box representation of the decision process. Therefore, our approach is compared with two simple heuristics that do meet all 3 requirements: a greedy approach and a FCFS approach.

Table 1 – Overview of decision support techniques in literature with respect to 3 requirements

Technique	Real-time decision support	Level of information required		White box
		Demand	*Capacity*	
Shen and Khoong (1995), Network flow	Periodically	Complete	Complete	N
Cheung and Chen (1998), Stoch. Progr.	Time restrictive*	Distribution	Complete	N
Ziliaskop. and Wardell (2000), Shortest p.	Instantly	Current order	Ignored	N
Janssen *et al* (2004), Multi-step heuristic	Periodically (15m)	Active orders	Complete	N
Bandeira *et al* (2009), Network flow	Periodically	Distribution	Complete	N
Nabais (2013), Model predictive control	Instantly	Distribution	Complete	N
Li *et al* (2013), Sequential LP	Periodically	Complete	Complete	N
Cormen *et al.* (1990), Greedy	Instantly	Current order	On request^{+}	Y
Ishfaq and Sox (2012), First Come First Serve	Instantly	Current order	On request^{+}	Y
Proposed method, Decision Tree	Instantly	Historic orders	Historic^{+}	Y

* The proposed stochastic programming approach can be applied periodically, but may take very long to solve

$^{+}$ With Greedy and FCFS the operator can request up-to-date capacity info per incoming order; this is also possible in our proposed method

2.3. Decision trees

We select decision trees as the classification approach for our study for several reasons. Firstly, decision trees provide direct insight into which rules and criteria lead to a decision. This is important for practical acceptance by the manual planning operators and is defined as a *white box* property. Secondly, decision trees can be trained using offline data. Subsequently, a decision tree can be used to distinguish between more than two classes, i.e., different inland services. Finally, the learning method must be suitable for using input parameters with categorical data, for instance, the customer type (Kotsiantis, 2007). Huysmans *et al.* (2011) describe several rule-base decision support methods: decision trees, decision tables and textual rule set descriptions. They performed empirical tests of users using these methods on various problems. For classification type problems, as in our case, decision tables result in slightly faster and more accurate results. However, as no direct methods for obtaining decision tables exist, decision trees are a suitable alternative.

The challenge is to create an accurate classifier. The accuracy only be determined after the learning process, i.e. by splitting the data in a training and test set, or by cross-validation techniques (Kotsiantis, 2007). In our method we use a test set and a training set to validate the performance of the classifier.
Decision tree classifiers are used as a method to structure complex decision-making. The decision is split up in multiple stages of simpler sub decisions. A decision tree can be represented by an acyclic directed graph, where a decision rule is associated with each node (Safavian et al, 1991). To make a decision using a decision tree, the sub decisions per node are applied recursively to the parameters of the case. The decision rule at each node level defines what the next node of the decision process will be. This is called the child-node. Nodes without children are called leaf nodes and are associated with the final decision outcomes of the tree.
The generalization error of a tree is defined as the misclassification rate over the input distribution (Rokach *et al.*, 2005). Typically, the goal for creating a decision tree is to find an optimal tree that minimizes the generalization error. Finding an optimal tree is an NP hard task, which is only feasible for small problem sizes (Rokach *et al.*, 2005). The topology of a decision tree and the decision rule at each node can be estimated empirically, using real-world data for which the intended outcome is known, i.e., supervised learning (Arentze and Timmermans, 2004).
Estimating a decision tree involves three aspects: design of the tree structure, the inference method decision rule at each node, and the selection of the feature set containing the input parameters (Safavian et al, 1991). The tree structure and decision rules are determined using a learning heuristic, or *inference method*. Rokach *et al.* (2005) provide an overview of inference methods: Most often a top-down heuristic is used as inference method, although bottom-up inference methods also exist. The inference of the tree usually involves a growth phase followed by a pruning phase. In the growth phase, branches are added starting from the root of the tree while considering a splitting criterion. In the pruning phase, branch nodes are turned into leave nodes and the leaf nodes of that node are removed (Rokach *et al.*, 2005). According to Arentze and Timmermans (2004), the following are the most widely used learning heuristics: C4.5 (Quinlan, 1993), CART (Breiman, 1984) and CHAID (Kass, 1980). All methods consider a condition with a single input variable as splitting criterion and use a top down induction method. Chandra and Varghese (2009) mention two popular splitting criteria: the Gain ratio (Quinlan, 1993) and the Gini index (Breiman, 1984). Lastly, the set of input parameters must be determined. Often, this is carried out in a greedy way, by adding the input parameter that adds the most value to the classification accuracy iteratively, until no more improvement is made (Rokach *et al.*, 2005).
The induction of a decision tree via a learning heuristic requires a training set for the learning process and a test set to evaluate the quality of the induced tree in a cross-validation. If some observations are more important than others, it is possible to add observation weights to each observation.

3. Decision trees for real-time decision making in intermodal planning

3.1. Approach

This study focuses on developing a method that is suitable for operational usage in a real-time setting. The quality of the real-time method is compared to the quality of a theoretically optimal solution, obtained offline. In the method we propose in this study, we do not aim to formulate the online decision problem explicitly. Instead we aim to translate the results of an optimal model into rules for online application automatically. We do not pursue rule inference by interviewing operational planning experts, as it has two disadvantages. This approach typically results in a few rules per man day (Quinlan, 1986). An expert system that can provide decision support for container transportation planning may require a large amount of rules. As a result, the rule inference for an entire network may be time-consuming. Secondly, the quality of the transportation by the operational planning experts is unsure. For these reasons, we use a machine learning technique to infer the decision rules based on optimal planning solutions of the offline problem. The DSS is induced in a series of steps. The approach depicted in Figure 1 is formalised in Algorithm 1. Section 3.2 to 3.4 describe the steps 2 to 4 of Algorithm 1, respectively.

Algorithm 1 – Obtain real-time decision support

1. Assemble N demand sets for training
2. Determine P optimal solutions for each demand set using the CLCAT model.
3. Infer decision tree T based on $N \times P$ solutions
4. Use decision tree T in a real-time setting on a per-container planning heuristic.

3.2. Finding P optimal solutions using the CLCAT model

The model we use to determine the optimal solution for the transportation planning is based on the earlier proposed LCAT model (Van Riessen et al., 2014-a). That model delivers optimal solutions for the planning of an entire network. For the characteristic intermodal problem studied in this article, we introduce the simplified formulation of minimising costs on a single corridor, the Corridor Linear Container Allocation model with Time restrictions (CLCAT).

The set of all cargo types that must be planned is denoted by demand set C and the set containing all services by S. The total number of TEU of cargo type c that must be transported is denoted as the demand d^c, this demand must be transported on one of the intermodal services or by direct truck. The set of services includes all available train and barge services, but not trucks, as trucks do not depart at predefined service schedules. Each service has a slot capacity u_s and a weight capacity of m_s, departs at time T_D^s and arrives at time T_A^s. Let x_s^c denote the number of TEU of cargo type $c \in C$ that is assigned to service $s \in S$. Each container of class c has a weight of W_c and must be transported in the time window from $t^c_{\text{available}}$ to t^c_{due}. The number of days that these containers are late is denoted by τ_s^c. The number of TEU of cargo class c assigned to a direct truck is denoted by v^c. No capacity or time restrictions are considered for trucking, as it is only used in exceptional cases. It is assumed that required trucking capacity is readily available. The transit costs of transporting one TEU on service s are denoted as c_s and the cost for direct trucking of a container of cargo class c is denoted as c_t^c. The objective of CLCAT is to minimise the total transportation costs of all containers, considering a penalty for overdue delivery of c_τ per day:

$$\min J_{\text{CLCAT}} = \sum_{s\in S}\sum_{c\in C}(c_s x_s^c - c_\tau \tau_s^c + c_t^c v^c), \tag{1}$$

subject to:

$$v^c + \sum_s x_s^c = d^c \qquad \forall c \in C \tag{2}$$

$$\sum_c x_s^c \le u_s \qquad \forall s \in S \tag{3}$$

$$\sum_c W_c x_s^c \le m_s \qquad \forall s \in S \tag{4}$$

$$x_s^c T_D^s \ge x_s^c t^c_{\text{available}} \qquad \forall c \in C, \forall s \in S \tag{5}$$

$$\sum_q x_s^c \,(T_A^s - t^c_{\text{due}}\,) \le \tau_s^c \qquad \forall c \in C, \forall s \in S \tag{6}$$

$$x_s^c, \tau_s^c, v^c \ge 0 \qquad \forall c \in C, \forall s \in S, \tag{7}$$

where the maximum capacity of service s is denoted by u_s (TEU capacity) and m_s (mass capacity). Constraints (2) ensure that all demand is met. By constraints (3) and (4), the total number of TEU on each service is restricted to the available capacity. Constraints (5) and (6) are the time constraints, where constraints (6) are for on-time delivery: τ_s^c measures the total number of days that containers of cargo class c on service s are late. Finally, constraints (7) are the nonnegativity constraints for the three sets of variables.

For most problems, multiple equivalent optimal solutions of $(1)-(7)$ exist, as each solution consists of specific assignments of cargo to a service. Often, some of these flows are fully exchangeable, e.g. because multiple services s are available with equal costs. In order to prevent overfitting on one optimal solution and to get decision rules that resemble all available optimal solutions as closely as possible, we need to determine a set of optimal solutions for demand set C. Finding all optimal solutions is a very difficult task (Valiant, 1979), so we use a new, innovative approach, aimed at finding a random subset of the set of optimal solutions. In order to generate such a random subset of optimal solutions, we solve the following problem P times:

$$\min \quad \sum \boldsymbol{rx} \tag{8}$$

subject to: $(2)-(7)$ (9)

$$\sum_{s\in S}\sum_{c\in C} c_s\ x_s^c - c_\tau \tau_s^c + c_t^c v^c = J_{\mathrm{CLCAT}} \tag{10}$$

where $\boldsymbol{x} = [x_s^c, \tau_s^c, v^c]^{\mathrm{T}}$, a vector of all decision variables, and $\boldsymbol{r}$ is a row vector of the same length of random numbers from the standard uniform distribution (ranging from 0 to 1). By (10), all feasible solutions of $(8)-(10)$ are optimal solutions of $(1)-(7)$. The random vector $\boldsymbol{r}$ is introduced to get random subset of solutions out of the set of all optimal solutions, i.e., solving $(8)-(10)$ P times corresponds to generating P random optimal solutions from the set of optimal solutions to $(1)-(7)$. By definition, all these solutions have equal objective value J_{CLCAT}. As mentioned in the introduction, a centralised offline optimisation method such as the CLCAT model is not useful in many practical situations for three reasons: it does not support real-time decisions; it would require extensive IT implementation to obtain all data form the transportation system; it requires complete information. In practice, real-time decisions require a heuristic provides decision support per incoming order anticipating unknown information, such as future demand. Current practical heuristics do not do that, but only provide solutions for available information. Below, we describe how the proposed DSS framework uses the optimal solutions on historic data in a learning algorithm to obtain a real-time DSS.

3.3. Decision tree inference

In order to obtain a decision tree that performs well under various circumstances and demand sets we use N demand sets for training. Historical data provides the distribution of attribute values; the sets can be demand sets from the last N weeks for instance. It is assumed that future demand will show a similar structure as the historic demand sets. Per demand set, P optimal solutions are determined using $(8)-(10)$. In this study, we use $P = 50$ optimal solutions. In total, we use P plans for N demand sets, i.e. PN optimal plans. Each observation in this set of solution denotes the allocation of x_s^c containers of cargo class c to service s. Each plan can contain $|C||S|$ observations, resulting in $m = NP|C||S|$ observations in total for the inference process. For creating the tree, the importance of each observation is proportional to the number of containers x_s^c that is assigned, hence x_s^c is used as the observation weight. We aim to create a decision tree that predicts the allocated service, based on selected input features from the transportation booking.

The inference of the decision tree is carried out using the CART method (Breiman *et al*, 1984), with Gini's split criterion: This method is suitable for discrete and nominal input features, aims for splitting on the most distinguishing feature and is also suitable for small data sets. The CART method uses recursive partitioning and selects in each node the input feature that gives the least impurity of the child nodes, according to the Gini's index. The Gini index is denoted by (Rokach, 2005):

$$G(y,Z) = 1 - \sum_{c_j \in \mathrm{dom}(y)} \frac{\left|\sigma_{y=c_j} Z\right|^2}{|Z|^2},$$

where y represents the target attribute, $|Z|$ represents the number of observations in the set and $|\sigma_{y=c_j} Z|$ the number of observations in the set with target value $y = c_j$. Hence, a pure node with just one class has only observations with $y = c_1$ and Gini index 0; otherwise the Gini index is positive. So the Gini index is a measure of node impurity. The recursive partitioning of nodes continues until in each node the stopping

criterion has been reached; we choose to stop splitting a node that corresponds to less than 20% of the average allocation to a service, i.e. 20% of the number of observations m divided by the number of services s. Hence, the minimum node size is $\epsilon = 0.2\frac{m}{s}$. Further detailing the allocation would likely result in overfitting. With each leaf node, a table is associated containing the class distribution in that leaf node for all observations in the training sets. If a leaf node is pure and the Gini-index equals zero, the classification table for that node has only one entry: all observations in the training set associated with that node were assigned to a single inland service. For leaf nodes with some impurity, the largest class indicates the label of that leaf node.
The obtained decision tree classifies incoming transport orders by recursively making sub decisions until a leaf node is reached. A classification table is association with each leaf node, indicating the distribution of inland services used for bookings that end in that leaf node. A human planning operator can use this list for determining the actual allocation, while he considers the remaining capacity and other practical considerations. In the approach we describe in this paper, we use a heuristic for the actual allocation, called the DT heuristic in the remainder of the paper.

3.4. Applying the decision tree in a per-container planning heuristic

We introduce the DT heuristic to use the inferred decision tree for assigning incoming orders to inland services. It can be used on a per-container basis. For each order, the splitting criterion in subsequent nodes are applied. Orders for transporting multiple containers can be considered as one, without loss of generality as the decision tree gives the same results for each container. In practice, this process can be carried out by a human operator using the decision tree. In order to demonstrate our approach in a systematic way we use an automated heuristic to generate the transportation plan for our case study. Using the decision tree, an incoming transportation order is associated with one of the leave nodes. The DT heuristic will use the classification table of that node containing the distribution of services associated with that leaf node in decreasing order. The DT heuristic is provided in Algorithm 2, considering a previously obtained decision tree T. Note that the greedy strategy of step 4 in Algorithm 2 will select a direct truck if none of the available inland services has capacity left. The allocation process is repeated at the arrival of every transportation order. To clarify the process of the decision tree inference and application, the next section shows an explanatory example for the characteristic intermodal problem.

Algorithm 2 – Decision tree heuristic using decision tree T

1. Consider incoming container transportation order
2. Apply decision tree T to obtain the classification table of the resulting leave node
3. Assign containers to the services that have capacity left proportionally to distribution in the classification table
4. If none of the indicated services has capacity available, the container is assigned to another service according to a greedy strategy, i.e., the feasible service with minimum cost is selected.

3.5. Complexity of pre-processing steps and real-time operations

For real-time decision support, the running time of any DSS is critical. Our method supports real-time decisions by using a pre-processing step based on historic data. Here we show that the complexity of each step is polynomial, and that the pre-processing steps result in a significant reduction of complexity for the real-time decision process.
Without pre-processing, we could apply LP $(1)-(7)$ to find the optimal plan. The average case complexity of solving LP $(1)-(7)$ is determined using the result of Borgwardt (1982) for the average running time of the simplex method:

$$O\left(n^3 p^{\frac{1}{n-1}}\right), \tag{11}$$

in which n denotes the number of columns, i.e. the number of decision variables, and p the number of rows, i.e. the number of constraints. In our case, the number of decision variables $n = 2(|C||S|) + |C|$, with $|C|$ and

$|S|$ denoting the number of cargo types and services, respectively. The number of constraints $p = 3|C||S| + |C| + 2|S|$. Substituting n and p in (11), and observing that $n - 1 \geq 2$, the average case complexity of LP $(1) - (7)$ is

$$O((|C||S|)^{3+\alpha}), \tag{12}$$

with $0 < \alpha \leq \frac{1}{2}$.

In the first pre-processing step, we consider LP (8) – (10), which also has average case complexity (11) as only 1 constraint is added in comparison to LP $(1) - (7)$. We solve this LP PN times (Section 3.2), i.e. for N historic periods and P repetition per period.

The second pre-processing step is the learning algorithm for the DT (Section 3.3). The complexity of the learning algorithm can be determined using the rationale of Su and Zhang (2006). In each node, containing a subset of $|Z|$ observations, one of l attributes must be selected as splitting criterion by considering each possible split for each candidate attribute. This operation has complexity of $O(|Z|l)$. For each level in the tree, the union of the subsets comprises all m observations. Hence, the complexity for each sublevel is of $O(ml)$, with $m = NP|C||S|$. Considering the minimum node size $\epsilon = 0.2\frac{m}{|S|}$, the maximum depth of the tree (the maximum number of levels) equals $\frac{m}{\epsilon} = 5|S|$. During the inference process, a maximum of $5|S|$ levels must be considered. Hence, the complexity of the entire inference operation is:

$$O(|C||S|^2 l). \tag{13}$$

From (12) and (13) we can see that the average running time of the pre-processing steps to obtain the tree is polynomial in the number of cargo types and the number of services considered in the problem. The algorithm can also be applied to larger scale problems.

Finally, we consider the problem of the real time decision problem as well (Section 3.4). The decision tree is used for each incoming order. The complexity of applying the tree only depends on the number of levels in the tree, so it has a complexity:

$$O(|S|). \tag{14}$$

Comparing (14) with (12), we see that the making decisions with the decision tree is much less computationally complex than solving the LP for the optimal solution. This supports our aim to derive a decision support system that can be easily applied in real-time in practice, leveraging information of historic optima.

3.6. Validation of the DT heuristic using the characteristic intermodal decision problem

In this section, we make a comparison between the optimal strategy and using our DT heuristic in the characteristic intermodal problem. For this problem, as depicted in Figure 3, the question is how many slots to reserve for each destination. It is possible to determine an optimal strategy analytically; but even for such a simple problem it requires detailed analysis. Here we show that our DT heuristic is much more convenient to apply and results in results very close to the optimum. The purpose here is not to study a very realistic case, but merely a small example for which a comparison with the optimum is possible. In Section 4 we present a method for assessing the quality of the DT heuristic in more general cases, for which comparing to the optimum is not possible. This is also demonstrated in a more elaborate case study in Section 5.

In the characteristic intermodal decision problem, a planner has to decide between a lower priced, fixed capacity mode and a higher priced ample capacity mode. Two order types compete for the lower priced mode, in this case orders for two destinations, B and C. Containers for both destinations can go by a fixed capacity on rail, or using ample capacity of trucking, so we consider 4 transportation options: rail to B (BR), rail to C (BT), trucking to B (CR) and trucking to C (CT). For an optimal strategy, we need to determine the booking limits for rail slots toward B (K_{B}) and toward C (K_{C}), adhering to $K_{\mathrm{B}} + K_{\mathrm{C}} = K$, resulting in the lowest expected total cost. A detailed overview of determining these booking limits is presented in Appendix A, here we provide the main results.

3.6.1. Optimal slot reservation

We assume independent distributions for the number of containers for destination B and C during a planning period, denoted by N_{B} and N_{C}, respectively. For each destination d and mode m, we consider the number of used slots (i.e. planned containers), denoted by S^{dm} with a cost denoted by c_{dm}. Note that the costs of transporting one unit from A to C via rail, c_{CR}, also include the costs for last mile trucking between B and C. In line with the characteristic intermodal problem, we consider the set of problems for which the trucking costs exceed rail costs for all destinations x, i.e., $c_{d\mathrm{T}} \geq c_{d\mathrm{R}}$. The optimal booking limit on the train for each destination can be found by minimising the total costs for all transports, denoted by J. The expected costs are a sum of the costs for all four transportation options:

$$\mathbb{E}(J) = c_{\mathrm{BR}}\mathbb{E}\left(S^{\mathrm{BR}}\right) + c_{\mathrm{BT}}\mathbb{E}\left(S^{\mathrm{BT}}\right) + c_{\mathrm{CR}}\mathbb{E}\left(S^{\mathrm{CR}}\right) + c_{\mathrm{CT}}\mathbb{E}\left(S^{\mathrm{CT}}\right), \tag{15}$$

where $\mathbb{E}$ is the expectancy operator. For a known booking limit K_{B} (and K_{C}), we can determine the expected utilisation on each mode and each destination. For this, we use a similar approach as in Van Riessen *et al.* (2015), in which we used the expected utilisation to determine an optimal product mix. First, we determine the expected slots used for transportation to destination B. The expected number of train slots used for destination B is the sum of the conditional expectation if $N_{\mathrm{B}} \leq K_{\mathrm{B}}$ and of the case that $N_{\mathrm{B}} > K_{\mathrm{B}}$:

$$\mathbb{E}\left(S^{\mathrm{BR}}\right) = \mathbb{E}\left(N_{\mathrm{B}} \,|N_{\mathrm{B}} \leq K_{\mathrm{B}}\right)\mathbb{P}\left(N_{\mathrm{B}} \leq K_{\mathrm{B}}\right) + K_{\mathrm{B}}\mathbb{P}\left(N_{\mathrm{B}} > K_{\mathrm{B}}\right), \tag{16}$$

where $\mathbb{P}(\cdot)$ denotes the probability of event $\cdot$, and

$$\mathbb{E}\left(N_B \,\middle|N_B \leq K_B\right) = \sum_{n\in\{1,2,\dots\}} n\frac{\mathbb{P}\left(N_B = n \cap n \leq K_B\right)}{\mathbb{P}\left(N_B \leq K_B\right)} = \sum_{n\in\{1,2,\dots,K_B\}} n\frac{\mathbb{P}\left(N_B = n\right)}{\mathbb{P}\left(N_B \leq K_B\right)}. \tag{17}$$

By substituting (17) in (16), we get the expected number of rail slots used for destination B:

$$\mathbb{E}\left(S^{\mathrm{BR}}\right) = \sum_{n\in\{1,2,\dots,K_B\}} n\mathbb{P}\left(N_B = n\right) + K_B\mathbb{P}\left(N_B > K_B\right) \tag{18}$$

The results for the other transportation options are shown in Appendix A. Substituting (4) and $(\mathrm{A7}) - (\mathrm{A9})$ in (15) gives the expected costs for a given value of K_{B} (and $K_{\mathrm{C}} = K - K_{\mathrm{B}}$). The optimal booking limit for rail slots used for destination B, i.e., the optimal value of K_{B} is then determined by solving

$$K_{\mathrm{B}}^{*} = \underset{K_{\mathrm{B}}}{\operatorname{argmin}}\, \mathbb{E}(c_T).$$

This is valid for any independent distributions of N_{B} and N_{C}. Here, we'll determine the optimal value K_{B}^{*} for a uniform distribution of the demand to destinations B and C. E.g., for uniform distributions $N_{\mathrm{B}} \sim U(b_1, b_2)$, $N_{\mathrm{C}} \sim U(c_1, c_2)$, that allow a total demand exceeding the train capacity, i.e., $b_2 + c_2 > K$, the expected number of rail slots used for destination B can be estimated using (18) as

$$\mathbb{E}\left(S^{\mathrm{BR}}\right) = (K_B - b_1)\frac{b_1 + K_B + 1}{2(b_2 - b_1)} + K_B\frac{b_2 - K_B}{b_2 - b_1}$$

$$\mathbb{E}\left(S^{\mathrm{BR}}\right) = \frac{{K_{\mathrm{B}}}^2 + {b_1}^2 - 2K_{\mathrm{B}}b_2 + b_1 - K_{\mathrm{B}}}{2(b_1 - b_2)}. \tag{19}$$

The results for the other transportation options are presented in Appendix A. By using $K_{\mathrm{C}} = K - K_{\mathrm{B}}$ and finding the minimum of (15), we can find the analytical optimal value for $K_{\mathbf{B}}$:

$$K_{\mathrm{B}}^{*} = \frac{\left(c_2 - K + \frac{1}{2}\right)(c_{\mathrm{CR}} - c_{\mathrm{CT}})(b_1 - b_2) + \left(b_2 + \frac{1}{2}\right)(c_{\mathrm{BR}} - c_{\mathrm{BT}})(c_1 - c_2)}{(\mathrm{b}_1 - \mathrm{b}_2)(\mathrm{c}_{\mathrm{CR}} - \mathrm{c}_{\mathrm{CT}}) + (\mathrm{c}_1 - \mathrm{c}_2)(\mathrm{c}_{\mathrm{BR}} - \mathrm{c}_{\mathrm{BT}})} \tag{20}$$

The optimal value K_B^{*} is the upper limit for slots that can be used for transporting containers to destination B by rail. Next, we will compare the result of the proposed decision tree method with the analytical optimum.

3.6.2. Comparing DT heuristic to optimal solution for the characteristic intermodal problem

Now, we will use some specific instances to compare the performance of the DT heuristic with the analytical optimum for this problem. We assume a capacity of $K = 40$ for the train and we consider three demand scenarios. Table 2 shows the hypothetical costs and demand scenarios. The cost for the more expensive mode, trucking, is equal in both cases. The rail connection that is available travels directly to destination B, omitting the need for a last-mile trucking leg. Hence, this is the lowest priced transport. Using the rail connection to

destination C requires a last-mile truck delivery. For this transport an intermediate cost is assumed. The three demand scenarios all result in a conflict between planning containers for destination B or C on the train. We determine the optimal booking limits (i.e. slot reservations) using (20). The resulting limits for destination B and C are denoted in Table 3. For all three instances, we applied Algorithm 1 to obtain decision trees. As the first step, we assemble data patterns for training: we generate 20 sets of demand volumes for destination B and C using N_B and N_C, with a specific arrival sequence. For the real-time application of the DT heuristic, the sequence of bookings is relevant. Secondly, we find the optimal solution for each of these 20 demand sets using the CLCAT model. Thirdly, we use the results to train the decision tree on, with the following features for a container transportation order j: the destination D_j and the amount of containers with destination C that has already been ordered d_C^j before order i.

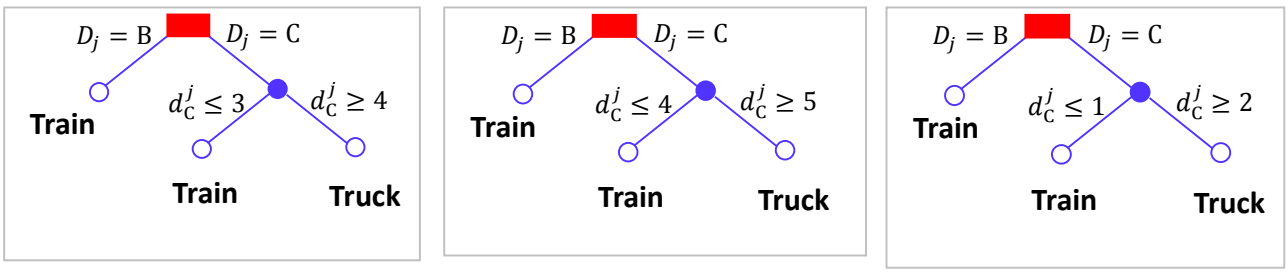
As the decision tree inference method is based on the demand set, we repeated the process 3 times, creating 3 decision trees per demand scenario, using independent demand sets. Figure 5 shows the resulting decision trees. If we inspect the decision trees for scenario 1, we observe that the trees shown in Figure 5a and 5b result in a maximum of 3, resp. 4, train slots for destination C. This is remarkably close to the optimum of 4. However, Figure 5c shows a tree that only uses 1 rail slot for destination C, which is too conservative, compared to the optimum. For demand scenario 2, the decision trees shown in Figure 5d and 5e result in a maximum of 10, resp. 11, train slots for destination C, which is close to the optimum of 12. Figure 5f shows a very simple decision tree, that allocates all incoming orders to the train. This results in the optimum in most cases (e.g. if $N_B + N_C \leq 40$), but in some extreme case may allow too many slots be used for destination C (e.g. if a large number of orders for destination C arrives early). Finally, for demand scenario 3, we obtain three very similar trees, in which 16 – 18 train slots are used for destination C (Figure 5g-i). This is again close to the optimum of 18 slots for destination C. The results for all scenarios show that the proposed method can identify the structure of the problem from the historic demand and provides a decision tree close to the optimal solution in most cases for this characteristic intermodal problem. In the next section, the DT heuristic will be applied to several variants of the characteristic intermodal problem with a range of parameters to validate the method for the more general situation.

Table 2 - Cost matrix and three demand instances of the characteristic decision problem

Destination	Train	Truck	Demand (1)	Demand (2)	Demand (3)
B	$c_{BR} = 2$	$c_{BT} = 4$	$N_B \sim U(30,40)$	$N_B \sim U(20,30)$	$N_B \sim U(15,25)$
C	$c_{CR} = 3$	$c_{CT} = 4$	$N_C \sim U(0,20)$	$N_C \sim U(0,20)$	$N_C \sim U(15,25)$

Table 3 – Optimal slot reservations for characteristic decision problem instances

Destination	K_B^*	K_C^*
Demand (1)	36	4
Demand (2)	28	12
Demand (3)	22	18



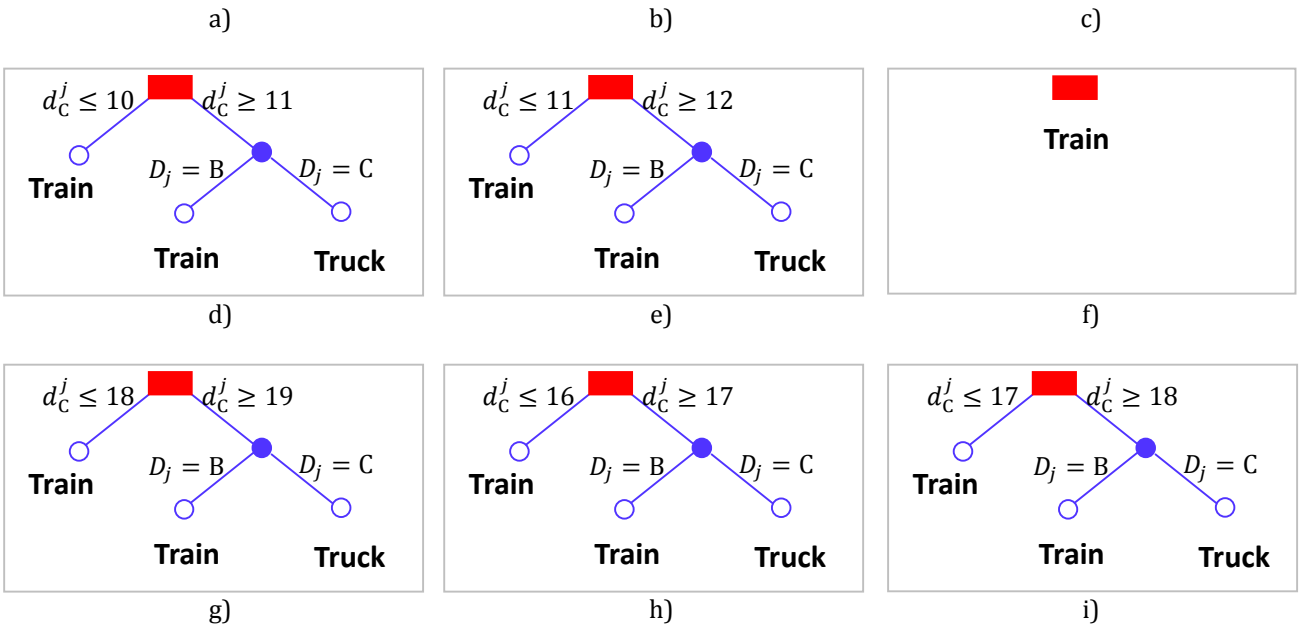


Figure 5 Decision trees obtained from Algorithm 1 for scenarios 1 (a-c), 2 (d-f) and 3 (g-i)

3.6.3. Evaluation of the DT heuristic applied to variants of the characteristic intermodal decision problem

The previous section looked in detail into several examples of generated trees. To validate the approach more generally, we apply it to a larger set of scenarios of the characteristic intermodal problem. For this, we determine multiple input values for the cost parameters and demand interval. Each combination of input parameters is considered as a separate instance. The demand intervals are defined as

$$N_B \sim U(b_1, b_1 + w_b);\ N_C \sim U(c_1, c_1 + w_c)$$

The parameters used are indicated in Table 4. Taking all combinations results in a total of 2187 scenarios. For each scenario, plans are created for 50 test instances using 4 methods: the theoretical optimum (obtained offline using the CLCAT model), the DT heuristic, the Greedy approach and the FCFS approach. For comparison, we use the competitive ratio of the objective costs of the heuristics and the optimal objective value. The competitive ratio is determined as the ratio of the optimal objective value and the objective value of the heuristic's results (Borodin and El-Yaniv, 1998). The optimal objective value can be obtained offline using the CLCAT model. By this definition, the competitive ratio of the CLCAT solution is always equal to 1, and for the heuristics always ≤ 1. A high competitive ratio indicates a good plan.

Table 4 - Values of input parameters

Parameter	Values	Parameter	Values
c_{BR}	0	b_1, c_1	0, 10, 20
c_{BT}	1, 2, 3	w_b, w_c	20, 30, 40
c_{CR}	2, 3, 4	K	40
c_{CT}	2, 3, 4		

For each scenario and for each method, the mean and standard deviation of the competitive ratio is determined over 50 test instances. The averages over all scenarios are shown in Table 5. Further detail can be seen in the boxplots of Figure 6. This shows that the DT heuristic performs generally better than the alternatives. Furthermore, we look in more detail into the differences in competitive ratio per scenario. Figure 7 shows the competitive ratio of the DT heuristic over the alternatives, in ascending order. The competitive

ratio $\eta_{\mathrm{G,DT}}$ of the DT heuristic over the Greedy algorithm is defined as the ratio of the total costs of using these methods, denoted as J_{DT} and J_{G}, respectively:, and similarly with the costs of FCFS, denoted by J_{FCFS}:

$$\eta_{\mathrm{G,DT}} = \frac{J_{\mathrm{G}}}{J_{\mathrm{DT}}};\ \eta_{\mathrm{FCFS,DT}} = \frac{J_{\mathrm{FCFS}}}{J_{\mathrm{DT}}}.$$

A number higher than 1 indicates that the DT heuristic outperforms the other method. The DT heuristic outperforms the Greedy algorithm in roughly half of the scenarios and the FCFS in most. More precisely, the competitive ratio of the DT heuristic is as least as good as the Greedy algorithm in 1746 (80%) scenarios and as least as good as the FCFS approach in 1838 (84%) scenarios.

Table 5 - Resulting mean and standard deviation of competitive ratios over 2187 instances

	DT	Greedy	FCFS
Mean	98.5%	94.7%	82.1%
Standard deviation	2.0%	3.6%	5.1%

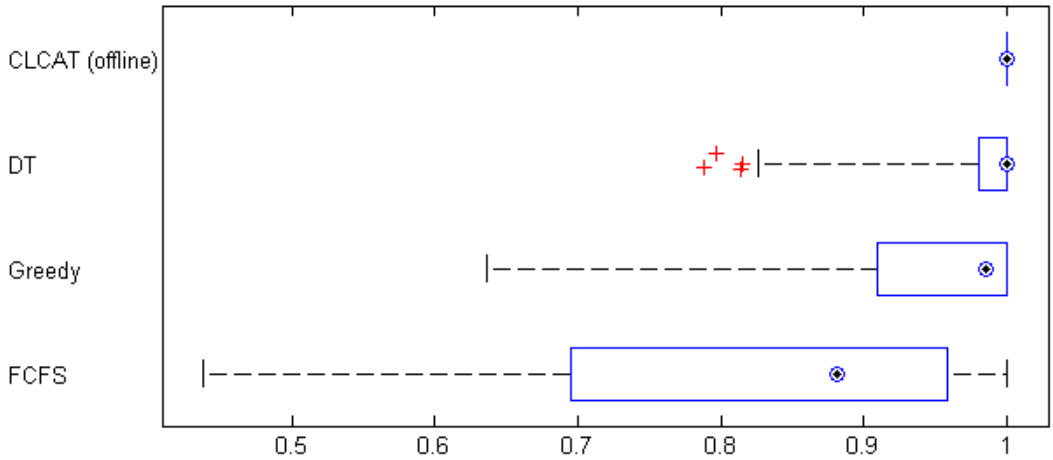


Figure 6 Average competitive ratios over 50 test sets for 2187 instances

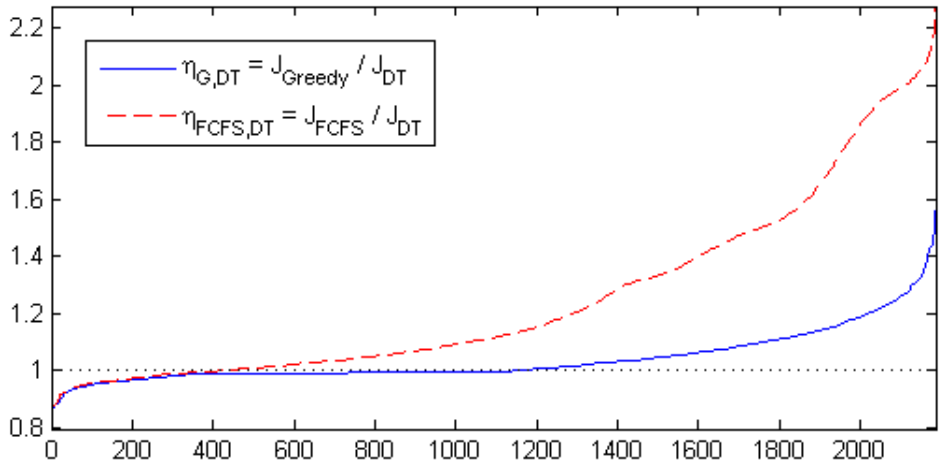


Figure 7 Competitive ratio of DT compared to Greedy / FCFS

The results of this section and the previous one show that in most cases the DT heuristic is preferable compared to the alternative heuristics, Greedy and FCFS. However, in some cases, the DT heuristic underperforms. The intuition for this is as follows. The DT heuristic uses a classification mechanism of optimal assignments in historic data sets and applies that to incoming orders. Optimal assignments are classified based on the order properties used in learning the tree. For those problem variants in which the optimal allocations follow a specific structure with a repetitive nature, the DT is capable of finding and exploiting that structure. As expected, due to stochasticity in the demand, some assignments might be misclassified. Generally, the benefit of following the DT assignment outweighs the cost of occasional misclassification. However, in problem variants with limited structure, the benefit of the DT heuristic diminishes, and the DT assignments may result in a higher costs than the alternative algorithms, due to misclassification. Considering the 20% of variants in which the DT underperforms the Greedy algorithm (as seen in Figure 7), we found this occurs predominantly if the rail corridor was not beneficial for destination C ($c_{\mathrm{CR}} > c_{\mathrm{CT}}$) or if the corridor is less beneficial for destination C than for destination B($c_{\mathrm{CR}} > c_{\mathrm{CT}}$ and $c_{\mathrm{CT}} - c_{\mathrm{CR}} < c_{\mathrm{BT}} - c_{\mathrm{BR}}$). In these cases, the using structure of the problem provided too little benefit to compensate for some misclassifications.
Therefore, it is vital to measure the quality of an obtained tree in any scenario *before* applying it in practice. The next section will provide an analysis method of the algorithm in order to determine the quality of a tree in advance, by assessing the misclassification error and two other sources of error.

4. Algorithm analysis

As shown in the previous section, the proposed DT heuristic provides an improvement over existing heuristics in most cases, but not in all. Therefore, in the following, we will address the performance of the DT heuristic by estimating the gap between the solution of the DT heuristic and the optimal solution. This algorithm analysis allows to address the expected quality of the DT directly after obtaining it, before applying it in practice. If the algorithm analysis results in acceptably low error, the DT can be used in practice. If not, we can reiterate the learning process, for instance with a different feature set.
The purpose of the decision tree heuristic is to find solutions as close as possible to the optimum. We use some notation to distinguish between several sets of solutions. We denote the set of all possible demand patterns by Δ and the set of all optimal solutions for all demands sets by Ω. During the decision tree inference, we use a subset of optimal solutions $Z, Z \subset \Omega$, the solutions to the training set of historic data. Also we have a set of solutions to an independent test set of historic data, denoted by $Y, Y \subset \Omega$. To find an estimate of the optimality gap of the DT heuristic, we consider each service in the service schedule separately. Let us consider service $b \in \{1,2, \dots\}$. A solution $s_i \in \Omega$ consists of a list of allocations of containers to services $b \in \{1,2, \dots\}$. For each service b, we denote the number of available slots as K_b. Let n_b^i denote the number of slots of service b that are in use in optimal solution s_i. The average number of slots in use on service b over all optimal solutions $s_i \in \Omega$, is denoted by $\overline{n_b^{\Omega}}$. The average number of slots in use on service b over the subset of optimal solutions $s_i \in Z$ is denoted by $\overline{n_b^{Z}}$. If decision tree T is applied to a subset of demand patterns, we obtain a set of solutions R'. We denote a solution in this set as $r'_k \in R'$. For service b the number of used slots in this solution is denoted by n'^k_b and the distribution of number of used slots is denoted as $n_b^{R'}$, with mean $\overline{n_b^{R'}}$. We use the accent to indicate the results of applying the tree directly, without the heuristic of Algorithm 2. Note that applying the decision tree directly may result in overbooking on a service, as no capacity limits are considered. Hence, finally, we consider the set of solutions R, obtained by applying Algorithm 2, the DT heuristic. A solution $r_k \in R$ denotes the heuristic's solution to demand set k. For service b the number of used slots in solution r_k is denoted by n_b^k and the distribution of used slots is denoted as n_b^R, with mean $\overline{n_b^R}$.
The cost on a slot of service b is denoted by c_b, and the total cost for all slots of service b in a solution s_i is $c_b n_b^i$. The DT heuristic aims to find the optimal number of used slots for each service b, but an error in finding the optimal number results in a different total cost for the found solution. For each service b, we distinguish between 3 error types:

Error type α. *Historic data error.* The decision support system is trained on historic data set Z. The target value of allocations to a service is based on the expected number of used slots of service b

from the historic data. The error in determining this target value from the historic data is denoted as error type α. This error type represents data inaccuracy, rather than an error of the algorithm.

Error type β. *Misclassification error.* Secondly, the decision tree must represent a classification that results in assigning the correct target number of containers to service b. The misclassification of containers in service b is denoted as error type β.

Error type γ. *Capacity-restriction error*. Finally, error type γ consists of misclassification due to the capacity limits of service b. Due to the stochastic nature of new demand patterns, the actual number of containers allocated to service b by the decision support system is variable. Because of the maximum capacity of service b, the expected value of containers allocated to this service is lower than the number would be in a case without capacity limits.

Below, we elaborate on how the size of all three error types can be estimated during the inference process, before the decision tree is used for real-time decision support.

4.1.1. Error type α, historic data error

First we look at error type α, the *historic data error*. This error measures the accuracy of the average number of used slots for service b in the historic data. We compare the set of available optimal solutions Z with the set of all possible optimal solutions Ω. In practice, not all demand patterns Δ are known, and not all optimal solutions Ω can be computed in advance. Instead, we can use only a subset of all optimal solutions $Z \subset \Omega$. The average number of used slots of service b is estimated using subset Z, using $\overline{n_b^Z}$ and variance ${\sigma_{b,\alpha}}^2$:

$$\overline{n_b^Z} = \frac{\sum_{i \in Z} n_b^i}{|Z|}, \qquad {\sigma_{b,\alpha}}^2 = \frac{\sum_{i \in Z} \left(n_b^i - \overline{n_b^Z}\right)^2}{|Z|} \tag{21}$$

Error type α denotes the difference in the allocations between subset Z and complete set Ω:

$$e_{b,\alpha} = \mathbb{E}\left(\overline{n_b^Z} - \overline{n_b^\Omega}\right).$$

A positive value $e_{b,\alpha}$ denotes an overestimation of the number of containers assigned to service b. According to the Central Limit Theorem, the difference $\overline{n_b^Z} - \overline{n_b^\Omega}$ is normally distributed with mean 0 and standard deviation $\sigma_{b.\alpha}/\sqrt{n}$. This gives $\mathbb{E}\left[e_{b,\alpha}\right] = 0$ and a 95% confidence interval for $e_{b,0}$ of

$$0 \pm 2\frac{\sigma_{b.\alpha}}{\sqrt{|Z|}}. \tag{22}$$

4.1.2. Error type β, misclassification error

Secondly, we determine the size of error type β, the *misclassification error* of the tree. Let us consider decision tree T that is able to provide a perfect classification, i.e. with a node purity of 100%. The probability of a container allocation to service b is a property of this decision tree, denoted by Pr_b. For this perfect decision tree, it holds that $Pr_b\overline{n^Z} = \overline{n_b^Z}$, with $\overline{n^Z}$ the average total number of containers in the solutions of the training set. However, in practice, the decision tree will not be perfect and $Pr_b\overline{n^Z} \neq \overline{n_b^Z}$, due to some misclassification errors. This is error type β. For estimating the size of error type β, we use the empirical distribution $n_b^{R\prime}$, denoting the number of slots of service b used in the tree's solutions ${r'}_k$. This empirical distribution n_b has a mean of

$$\overline{n_b^Z} = Pr_b\overline{n^Z} \tag{23}$$

and a variance of

$$\sigma_{b,\beta}^2 = \sum_{i \in Z} \frac{\left(n_b^i - \overline{n_b^Z} \right)^2}{|Z| - 1}. \tag{24}$$

For a given tree, the size of error type β can be determined directly after creation, by identifying:

$$e_{b,\beta} = \overline{n_b^Z} - \overline{n_b^Z},$$

with a variance equal to (24) and a 95% confidence interval of:

$$e_{b,\beta} \pm 2 \frac{\sigma_{b.0}}{\sqrt{|Z|}}. \tag{25}$$

The value of $e_{b,\beta}$ denotes the average number of containers that the tree assigns to service b in excess of the optimum. A positive value of $e_{b,\beta}$ denotes that more containers are booked on service b than in the optimum; a negative value denotes that less containers are booked on service b than in the optimum

4.1.3. Error type γ, capacity-restriction error

Finally, we consider error type γ, the *capacity-restriction error*. For assessing the quality of the online solution with this decision tree, we consider a set of new demand instances, test set D. If we apply the tree directly, we obtain a set of solutions R. We compare these with the set of solutions H, obtained by applying Algorithm 2.

The distribution of used slots on service b in solution $r_k \in R$ is denoted by $n_b^{r_k}$. We denote the distribution of $n_b^{r_k}$ by n_b^R. The number of containers classified in service b may be higher than its capacity K_b. On the other hand, we have $n_b^{h_k}$ the number of containers assigned to service b in solution $h_k \in H$, which is obtained by the DT heuristic. Hence, the capacity restrictions are adhered to. We denote the distribution of $n_b^{h_k}$ by n_b^H. Error type 2 estimates the difference between the solutions r_k and h_k. The theoretical distributions of n_b^R and n_b^H are not known, as they depend on the structure of tree T. In order to assess the size of this capacity-restriction error, we can use an empirical distribution of containers assigned to b, based on the distribution of n_b^R and n_b^H in the solutions for test set D. Now, the size of error type 2 for service b is determined by the expected difference in number of containers assigned to service b:

$$e_{b,2} = \mathbb{E}(n_b^H - n_b^R). \tag{26}$$

We obtain the variance of error γ also by using the empirical distributions:

$$\sigma_{b,\gamma}^2 = Var(n_b^H - n_b^R) = \sum_{k \in D} \frac{\left(\left(\overline{n_b^R} - \overline{n_b^H} \right) - \left(n_b^{r_k} - n_b^{r_k} \right) \right)^2}{|R| - 1}. \tag{27}$$

This gives a 95% confidence interval for error type γ of:

$$e_{b,\gamma} \pm 2 \frac{\sigma_{b,2}}{\sqrt{|R|}}. \tag{28}$$

4.1.4. Total error estimate

Finally, to estimate the total error in cost, we take the sum of all errors multiplied by the respective slot costs.

$$e = \sum_b c_b \left(e_{b,\alpha} + e_{b,\beta} + e_{b,\gamma} \right). \tag{29}$$

Note that a positive error value denotes additional costs compared to the optimal solution. For individual services, an individual error component can be negative, for instance, when Algorithm 2 allocates less containers to a specific service than in the optimal solution. Assuming that the error types α, β and γ for each service b are independent, the variance for the total error is:

$$\sigma^2 = \sum_b c_b \left(\sigma_{b,\alpha}^2 + \sigma_{b,\beta}^2 + \sigma_{b,\gamma}^2 \right). \tag{30}$$

Assuming independence is conservative. In practice, the errors may have dependencies, resulting in a lower variance because of cross-correlations between the error terms. Determining these cross-correlations in general will be difficult and we will use (30) to determine the error variance.

5. Case study

In Section 3.5 we compared the proposed method with the optimal strategy in a small example. In this section, we will apply the DT heuristic to a more elaborate case study, and we will use the analysis method of Section 4 to determine a confidence interval for the obtained results. In our experiments, we will compare three online heuristics and the optimal offline plan in a series of simulations. As before, we use the competitive ratio to compare the objective costs of the heuristics and the optimal objective value, obtained offline using the CLCAT model. A high competitive ratio indicates a good plan.

5.1. Scenarios

We compare two scenarios with identical service schedules, but different demand patterns. In Scenario 1 the demand is distributed randomly across the input features and in Scenario 2 a specific demand pattern is considered. First we consider Scenario 1. Figure 8 shows a service schedule for a week with the Estimated Time of Departure (ETD) and Estimated Time of Arrival (ETA) of three services from A towards B. Trucks can always be used, and are therefore not shown. As our model is aimed at operators of intermodal networks, trucking is only used as a last resort, e.g. in case of disruptions or peak demands. Hence, the incidental demand for trucking is generally easily met by available capacity. The trucking capacity does not have to be modelled. For generating the tree, we use the container's time of availability ($t^c_{\text{available}}$), the transport lead time ($t_L = t^c_{\text{due}} - t^c_{\text{available}}$) and the container weight W_c.

Also, five transport demand flows are indicated. These flows have some variability; for obtaining the demand, we use a normal distribution with a mean of 50 TEU and a standard deviation of 12.5 TEU, which we round to integers. We assume that the weight per TEU is 10 tonnes. The $t^c_{\text{available}}$ and t^c_{due} of these flows are indicated by the shaded bands in Figure 8. E.g., flow 1 is available at A at some point during Monday and is expected two days later at B. The only service suitable for this flow is Train 1, as it departs after the containers are available and arrives at B before the due time of flow 1. Alternatively, trucks could be selected, but at much higher costs. Table 6 provides details on the services.

For this scenario, the optimal solution can be easily found by reasoning: Flow 1 and 2 must be fully assigned to Train 1 and Barge 1, respectively. Flow 3 is fully transported by trucks, as no suitable service is available. Then finally, flow 4 and 5 share service Train 2 up to its full capacity, however, the expected number of containers in flow 4 and 5 is 100 TEU, which is larger than the TEU-capacity for Train 2 (90 TEU), so some trucks must be used. The same solution is found by applying the learning method proposed in the previous section. The trees inference process was implemented using the CART implementation of the Statistics toolbox of MATLAB R2012a and the experiments were carried out using an AMD Athlon II X2 3.0 Ghz processor. The resulting decision tree for scenario 1 is shown in Figure 9: the optimal solution is mapped entirely by the tree. Figure 9 also exemplifies three important characteristics of the method. Firstly, the tree can be used as decision support per incoming container transportation order. Hence, immediate feedback can be provided to the customer after processing the order. Secondly, the decision tree is presented in a comprehensible manner to human planners. If, because of some disruption, Barge 1 is late, the planner can make a manual decision on an alternative routing for affected containers. Thirdly, apart from the order information the human planner does not need anything else to apply the decision tree. Subsequently, he can manually check if the proposed allocation is suitable. No real-time information exchange is required.

Table 6. Inland services for the case study

	Capacity		Costs
	[TEU]	[tonnes]	[per TEU]
Train 1 and 2	90	1000	100

Barge 1	200	10.000	50
Trucking	-	-	285

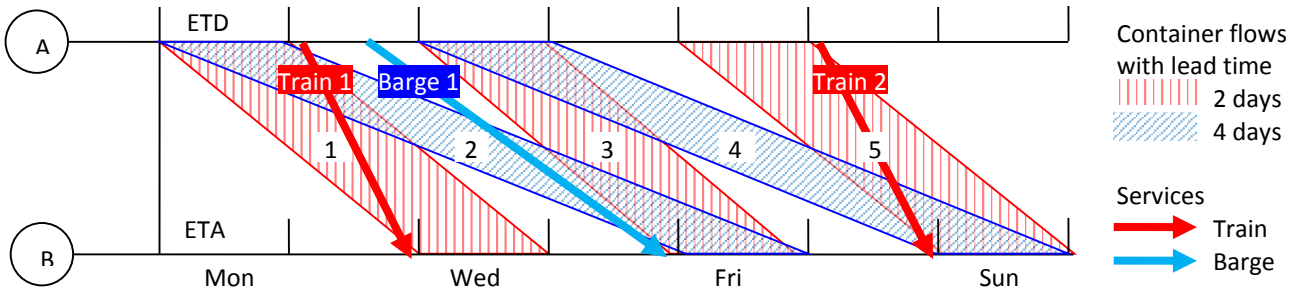


Figure 8. Transport orders and service schedule for the case study

Splitting rules per node
Node 1: $t^c_{\text{available}}$< Tue 12PM
Go to node 2, else node 3
Node 2: Lead time < 3 days
Select Barge 1, else Rail 1
Node 3: $t^c_{\text{available}}$ < Thu 12AM
Go to node 6, else select Rail 2
Node 6: Lead time < 3 days
Select trucking, else Rail 2

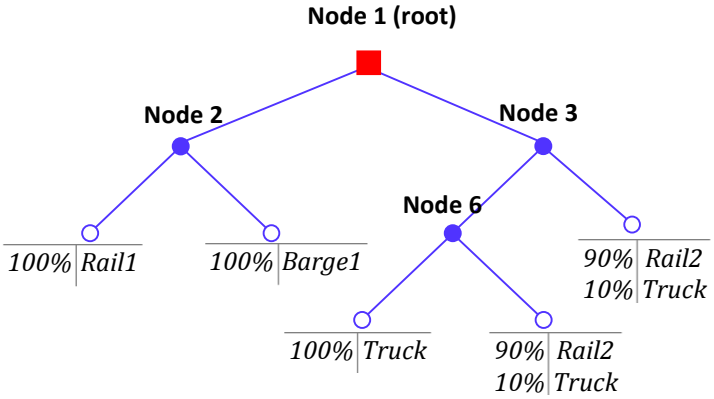


Figure 9. Tree obtained from the learning method (for Scenario 1)

Scenario 2 is identical to Scenario 1, except for one property: we now assume that all containers of flow 4 have a weight of 25 tonnes (all other flows remain 10 tonnes per TEU). If the solution of Scenario 1 would be applied to this scenario, Train 2 will most likely not be used to its full TEU capacity as the weight limit is reached well before that. So, for this scenario the optimal solution is slightly different: all containers of flow 5 must be planned on Train 2, while the containers of flow 4 are planned on Train 2 if possible. The remaining containers of flow 4 must be transported by trucks. With this solution, Train 2 can be used to its full TEU capacity. The corresponding tree for Scenario 2 was generated in a similar way as in Scenario 1 (depicted in Appendix B). Using (22), (25) and (28), we determine the expected error types 0, 1 and 2 for of applying this tree for scenario 2 in an online setting. Table 7 provides the expected error for the number of assigned containers to all four services including standard deviations. This results in a 95% confidence interval for the total error costs of [277, 1917] in excess of the estimated total cost of 36807. This corresponds to a 95% confidence interval for the expected competitive ratio between 95.1% and 99.3% using the tree in Figure 9.

Table 7. Estimation of error types 0,1 and 2 for scenario 2 (50 test sets)

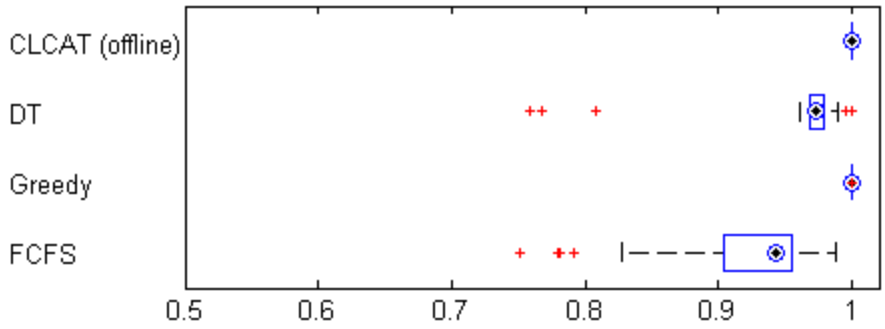
Service	c_b	$e_{b,0} \pm \sigma_{b,0}$	$e_{b,1} \pm \sigma_{b,1}$	$e_{b,2} \pm \sigma_{b,2}$	$e_b \pm \sigma_b$
Train 1	100	0±1.6	0±0	1±0.3	1±1.6
Train 2	100	0±1.2	-5±13.9	-2±15.7	-7±21.0
Barge	50	0±2.0	0±0	1±0.3	-1±2.1
Truck	285	0±2.6	5±13.9	1±15.3	6±20.8

5.2. Experiments

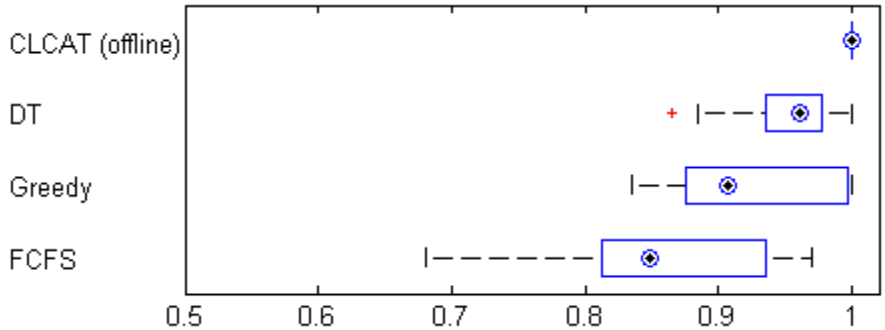
In order to test our method, we apply our DT heuristic to both scenarios described, as well as a greedy and a FCFS heuristic. In the greedy heuristic, every incoming order is assigned to the available slot with the lowest price, whereas the FCFS heuristic selects the earliest available slot. We compare using the competitive ratio. Hence, we compare three online heuristics and the optimal offline plan in a series of simulations.

We generate $N = 20$ new demand sets of a week for both scenarios. The sets for both scenarios are equal, except for the weight of flow 4: 25 tonnes per TEU instead of 10 tonnes. For each of these sets $P = 50$ optimal solutions are computed offline using $(8) - (10)$ and the results are used by the learning algorithm to generate the decision tree. Subsequently, we similarly generated 50 test sets for both scenarios and applied the four methods. Figure 10 shows boxplots of the competitive ratios of the 50 test sets for both scenarios. The box plots consists of the median in a box of from the 25 to 75 percentile. The whiskers denote the minimum and maximum competitive ratio, truncated at a whisker length of 1.5 times the length of the box. Results exceeding the whiskers are plotted as outliers. In Scenario 1, both the DT heuristic and the greedy heuristic found the optimum in all 50 test sets. The FCFS heuristic performed much worse with a competitive ratio of 0.92. From Figure 8 we can deduct that a FCFS heuristic will allocate orders of flow 2 to Train 1, and requires the use of trucks for flow 1 when Train 1 is full. This results in additional costs and a lower competitive ratio.

In Scenario 2, all heuristics have an average competitive ratio lower than 1. On average, the DT heuristic outperformed the other heuristics with a competitive ratio of 95.4% compared to 92.5% (greedy) and 86.1% (FCFS). A Wilcoxon signed-rank test was used to test the measured results for significance. This showed that the solution of the Greedy approach and the optimal solution in scenario 1 did not differ significantly. All other methods in scenario 1 and 2 resulted in significantly different competitive ratios. The results are in accordance with our expectations: with randomly distributed demand, the DT heuristic performs comparable with the low-level greedy heuristic as in Scenario 1. However, if the demand follows specific patterns, as in Scenario 2, the DT heuristic outperforms the other heuristics. Also, we see that the resulting competitive ratio is within the 95% confidence interval we have determined for the tree. This shows that our method performs as expected.



a) Boxplots of competitive ratios of 50 test sets (scenario 1)



b) Boxplots of competitive ratios of 50 test sets (scenario 2)

Figure 10. Experiment results of case study

6. Conclusions & future research

In this paper we proposed a new approach for decision support using information of offline optimal solutions of container transportation problems in an online setting. Three problems concerning existing optimization methods are addressed with this new method:

- The method provides instant decisions for an incoming order, for direct feedback to the customer. The method does not use additional planning updates after the first allocation, allowing the customer to align the container transportation with the subsequent steps in his supply chain.
- The method does not need the level of automation and standardized communication protocols for information exchange that centralised planning optimizations methods need. In our method optimisation can be carried out offline for historic data.
- The method provides a white box decision tree representation of decision rules, useful and acceptable for human operators in logistics as indicated by Huysmans *et al*. (2011).

We have selected decision tree classifiers as the supervised learning method, as this method can use offline input, it is suitable for classifying into multiple classes and it allows for categorical attributes. A general four step method is proposed: historic data assembly, optimisation of historic sets, decision tree inference, applying the decision tree in real time. In this article, we have used a container transportation setting to develop these steps. The proposed method uses an offline optimal planning method (CLCAT) to get optimal results of historic transportation problems. The results are translated into a decision tree with the CART inference method. Also, we used a mathematical model for obtaining multiple offline optima with equivalent objective value as input for the learning algorithm. Finally, we proposed an analysis to assess the quality of the tree using three error types: the historic data error, the misclassification error and the over-capacity error. By determining the size of the errors, the quality of the obtained decision tree can be estimated before applying it in practice.

The proposed DSS framework shows that traditional manual planning can be improved without extensive IT development. The DT heuristic uses optimal solutions and translates that to real-time decisions. Managers responsible for operational planning should consider such an approach as an alternative for costly, time-consuming and/or infeasible system integration projects. In a case study we found that the DT approach could reduce inefficiency, as measured by the competitive ratio, by half, amounting to a 3% reduction of transportation costs compared to a Greedy approach. Secondly, our tools for the algorithm analysis in Section 4 will support management in assessing the expected benefits before applying it in practice. Specifically for the case of intermodal transportation providers, our case study results show that this approach improves the quality of the plan significantly in comparison with every day practices such as Greedy or FCFS.

With the decision tree method it is possible to exploit specific demand patterns in historic data. We showed that the proposed method was able to identify a specific pattern automatically, such as a categories of containers with specific weights. However, if no specific patterns exist in the historic demand, the method will give results comparable to alternative low-level heuristics, such as first come, first serve (FCFS) or a greedy heuristic. It is important to use the proposed algorithm analysis for each case, to identify whether significant improvement can be expected from using the DT approach in comparison with alternatives.

Secondly, in this paper, we used the DT heuristic to model the human operator using the decision tree. In practice, this method supports the human operator by indicating suitable services for an order, while the operator is still able to incorporate his specific knowledge into his decisions.

Finally, the applicability of the DT approach is restricted by the problem size of the historic optimisation problems. Although our method results in a real-time DSS with a low complexity, by carrying out the more complex optimisation of historic data and the DT inference process offline, the problem size may be restrictive if it is not feasible to find historic optima at all. In that case an alternative approach for finding good solutions for the historic data may be used.

Future research on this method includes two directions. On the one hand, additional studies may improve the proposed method by iteratively improving the decision tree using online updating or introducing a guided decision tree learning process. On the other hand, new applications of the proposed method must be studied. The proposed DT approach may have value for any operational decision process with the three requirements of the problem addressed in this article, i.e. real-time decision support, no system automation, a white box system. The method is especially promising in situations that lack automated and standardised information exchange, such as planning in a hospital environment or scheduling in rail applications. The proposed four step method is suitable for applications in these topics, but future research must show its value in those cases.

Acknowledgements

This research is partially supported by the NWO/STW VENI project 'Intelligent multi-agent control for flexible coordination of transport hubs' (project 11210) of the Dutch Technology Foundation STW, and Erasmus Smart Port. The studied operational planning decisions and decision support for the intermodal problem were identified and discussed in cooperation with European Gateway Services, a subsidiary of Europe Container Terminals (ECT), Rotterdam. We thank our colleague Michel van de Velden for comments that greatly improved the manuscript.

References

Arentze, T. A., and Timmermans, H. J. (2004). A learning-based transportation oriented simulation system. *Transportation Research Part B: Methodological*, 38(7), 613-633.

Bandeira, D. L., Becker, J. L., and Borenstein, D. (2009). A DSS for integrated distribution of empty and full containers. *Decision Support Systems*, *47*(4), 383-397.

Bertsimas, D., Brown, D. B., and Caramanis, C. (2011). Theory and applications of robust optimization. *SIAM review*, *53*(3), 464-501.

Borgwardt, K. H. (1982). The average number of pivot steps required by the simplex-method is polynomial. *Mathematical Methods of Operations Research*, *26*(1), 157-177.

Borodin, A., and El-Yaniv, R. (1998). *Online computation and competitive analysis*. Cambridge: Cambridge University Press.

Branley, B., Fradin, R., Kimbrough, S. O., and Shafer, T. (1997, January). On heuristic mapping of decision surfaces for post-evaluation analysis. In *Proceedings of the Thirtieth Hawaii International Conference on System Sciences,* 416-425.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). Classification and regression trees. Boca Raton, FL: CRC press.

Carbonell, J. G., Michalski, R. S., and Mitchell, T. M. (1983). An overview of machine learning. In *Machine learning* (pp. 3-23). Springer Berlin Heidelberg.

Chandra, B., and Paul Varghese, P. (2009). Moving towards efficient decision tree construction. *Information Sciences*, *179*(8), 1059-1069.

Cheung, R. K., and Chen, C. Y. (1998). A two-stage stochastic network model and solution methods for the dynamic empty container allocation problem. *Transportation science*, *32*(2), 142-162.

Cormen, T.H., Leiserson, C. E. and Rivest, R. L. (1990). *Introduction to algorithms*. Cambridge: MIT press.

Douma, A. M. (2008). *Aligning the operations of barges and terminals through distributed planning*. University of Twente.

Gal, T., and Davis, G. V. (1979). *Postoptimal analyses, parametric programming, and related topics*. New York: McGraw-Hill International.

Giboney, J.S., Brown, S. A., Lowry, P. B., Nunamaker, J. F. (2015), User acceptance of knowledge-based system recommendations: Explanations, arguments, and fit, *Decision Support Systems* 72, 1-10.

Greenberg, H. J. (1998). An annotated bibliography for post-solution analysis in mixed integer programming and combinatorial optimization. In *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search.* Operations Research/Computer Science Interfaces 9*,* 97-147, Springer, USA.

Huysmans, J., Dejaeger, K., Mues, C., Vanthienen, J., Baesens, B., An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models, *Decision Support Systems*, 51(1), 141-154.

Ishfaq, R., and Sox, C. R. (2012). Design of intermodal logistics networks with hub delays. *European Journal of Operational Research*, *220*(3), 629-641.

Jansen, B., Swinkels, P. C., Teeuwen, G. J., de Fluiter, B. V. A., and Fleuren, H. A. (2004). Operational planning of a large-scale multi-modal transportation system. *European Journal of Operational Research*, *156*(1), 41-53.

Jenkins, L. (1990). Parametric methods in integer linear programming. *Annals of Operations Research*, *27*(1), 77-96.

Kass, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics* 29(2), 119-127.

Kotsiantis, S. B., (2007). Supervised machine learning: A review of classification techniques. *Informatica, 31*, 249-268

Li, L., Negenborn, R. R., and De Schutter, B. (2013, October). A sequential linear programming approach for flow assignment in intermodal freight transport. In *16th International IEEE Conference on Intelligent Transportation Systems* (pp. 1224-1230). IEEE.

Murthy, S. K. (1998). Automatic construction of decision trees from data: A multi-disciplinary survey. *Data mining and knowledge discovery*, *2*(4), 345-389.

Murty, K. G., Liu, J., Wan, Y. W., and Linn, R. (2005). A decision support system for operations in a container terminal. *Decision Support Systems*, *39*(3), 309-332.

Nabais, J.L., Negenborn, R.R., Carmona Benitez, R.B., Botto, M.A. A constrained MPC heuristic to achieve a desired transport modal split at intermodal hubs. In *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems* (IEEE ITSC 2013), The Hague, The Netherlands, pp. 714-719, October 2013.

Ngai, E. W., Cheng, T. E., Au, S., and Lai, K. H. (2007). Mobile commerce integrated with RFID technology in a container depot. *Decision Support Systems*, *43*(1), 62-76.

O'Kelly, M. E., and Lao, Y. (1991). Mode Choice in a Hub-and-Spoke Network: A Zero-One Linear Programming Approach. *Geographical Analysis*, *23*(4), 283-297.

Pak, K., and Dekker, R. (2004). *Cargo revenue management: Bid-prices for a 0-1 multi knapsack problem* (No. ERS-2004-055-LIS). ERIM Report Series Research in Management.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81-106.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann.

Rinnooy Kan, A. H. G., Stougie, L., and Vercellis, C. (1993). A class of generalized greedy algorithms for the multi-knapsack problem. *Discrete applied mathematics*, 42(2), 279-290.

Rokach, L., and Maimon, O. (2005). Top-down induction of decision trees classifiers - A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *35*(4), 476-487.

Safavian, S. R., and Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man and Cybernetics,* 21(3), 660-674.

Shen, W. S., and Khoong, C. M. (1995). A DSS for empty container distribution planning. *Decision Support Systems*, *15*(1), 75-82.

Steiger, D. M. (1998). Enhancing user understanding in a decision support system: a theoretical basis and framework. *Journal of Management Information Systems*, 199-220.

Su, J., and Zhang, H. (2006, July). A fast decision tree learning algorithm. In *Proceedings of the National Conference on Artificial Intelligence, 21*(1), p. 500.

Ursavas, E. (2014). A decision support system for quayside operations in a container terminal. *Decision Support Systems*, *59*, 312-324.

Valiant, L. G. (1979). The complexity of computing the permanent. *Theoretical computer science*, 8(2), 189-201.

van der Horst, M. R., and de Langen, P. W. (2008). Coordination in hinterland transport chains: a major challenge for the seaport community. *Maritime Economics & Logistics*, *10*(1), 108-129.

Van Hentenryck, P., and Bent, R. (2009). *Online stochastic combinatorial optimization.* The MIT Press.

Van Hentenryck, P., Bent, R., and Upfal, E. (2010). Online stochastic optimization under time constraints. *Annals of Operations Research*, 177(1), 151-183.

Van Riessen, B., Negenborn, R.R., Dekker, R. and Lodewijks, G. (2014-a). Impact and relevance of transit disturbances on planning in intermodal container networks using disturbance cost analysis. *Maritime Economics and Logistics.* Available from: http://dx.doi.org/10.1057/mel.2014.27.

Van Riessen, B., Negenborn, R.R., Dekker, R. and Lodewijks, G. (2014-b). Service network design for an intermodal container network with flexible transit times and the possibility of using subcontracted transport. *International Journal of Shipping and Transport Logistics.*

Van Riessen, B., Negenborn, R.R., Dekker, R. (2014-c). White box optimization of container transport planning. In *Proceedings of the International Forum on Operation Research and Statistics 2014.* Barcelona, Spain.

Veenstra, A., Zuidwijk, R., and Asperen, E. van (2012). The extended gate concept for container terminals: Expanding the notion of dry ports. *Maritime Economics & Logistics*, 14(1), 14-32.

Venkat, V., Jacobson, S. H., and Stori, J. A. (2004). A post-optimality analysis algorithm for multi-objective optimization. *Computational Optimization and Applications*, *28*(3), 357-372.

Verwer, S., Zhang, Y. and Q. C. Ye (2014), *Learning optimization models in the presence of unknown relations.* Available from: http://arxiv.org/abs/1401.1061v2.

Wallace, S. W. (2000). Decision making under uncertainty: Is sensitivity analysis of any use? *Operations Research*, 48(1), 20-25.

Ziliaskopoulos, A., and Wardell, W. (2000). An intermodal optimum path algorithm for intermodal networks with dynamic arc travel times and switching delays. *European Journal of Operational Research*, 125(3), 486-502.

Appendix A – Results on optimal slot reservation in characteristic intermodal problem.

A.1 Optimal slot reservation

For a known demand pattern, the booking limit that results in the highest expectation of used slots can be determined. We assume independent distributions for the number of containers for destination B and C during a planning period, denoted by N_B and N_C, respectively. For each destination d and mode m, we consider the number of used slots (i.e. planned containers), denoted by S^{dm}. The optimal booking limit on the train for each destination can be found by minimising the total costs for all transports, denoted by c_T. The expected costs are a sum of the costs for all four transportation options:

$$\mathbb{E}(c_T) = c_{\mathrm{BR}}\mathbb{E}(S^{\mathrm{BR}}) + c_{\mathrm{BT}}\mathbb{E}(S^{\mathrm{BT}}) + c_{\mathrm{CR}}\mathbb{E}(S^{\mathrm{CR}}) + c_{\mathrm{CT}}\mathbb{E}(S^{\mathrm{CT}}), \tag{A1}$$

where $\mathbb{E}$ is the expectancy operator. For a known booking limit K_{B} (and K_{C}), we can determine the expected number of slots used on each mode and each destination. First, we determine the expected slots used for transportation to destination B. The expected number of train slots used for destination B is:

$$\mathbb{E}(S^{\mathrm{BR}}) = \mathbb{E}(N_{\mathrm{B}} \mid N_{\mathrm{B}} \leq K_{\mathrm{B}})\mathbb{P}(N_{\mathrm{B}} \leq K_{\mathrm{B}}) + K_{\mathrm{B}}\mathbb{P}(N_{\mathrm{B}} > K_{\mathrm{B}}), \tag{A2}$$

where $\mathbb{P}(\cdot)$ denotes the probability of event $\cdot$, and

$$\mathbb{E}(N_B \mid N_B \leq K_B) = \sum_{n\in\{1,2,\dots\}} n\frac{\mathbb{P}(N_B = n \cap n \leq K_B)}{\mathbb{P}(N_B \leq K_B)} = \sum_{n\in\{1,2,\dots,K_B\}} n\frac{\mathbb{P}(N_B = n)}{\mathbb{P}(N_B \leq K_B)}. \tag{A3}$$

By substituting (A3) in (A2), we get the expected number of rail slots used for destination B:

$$\mathbb{E}(S^{\mathrm{BR}}) = \sum_{n\in\{1,2,\dots,K_B\}} n\mathbb{P}(N_B = n) + K_B\mathbb{P}(N_B > K_B) \tag{A4}$$

The expected number of units that is transported to destination B by truck is computed as

$$\mathbb{E}(S^{\mathrm{BT}}) = \mathbb{E}(N_{\mathrm{B}} - K_{\mathrm{B}} \mid N_{\mathrm{B}} > K_{\mathrm{B}})\mathbb{P}(N_{\mathrm{B}} > K_{\mathrm{B}}) = (\mathbb{E}(N_{\mathrm{B}} \mid N_{\mathrm{B}} > K_{\mathrm{B}}) - K_{\mathrm{B}})\mathbb{P}(N_{\mathrm{B}} > K_{\mathrm{B}}) \tag{A5}$$

with

$$\mathbb{E}(N_B \mid N_B > K_B) = \sum_{n\in\{1,2,\dots\}} n\frac{\mathbb{P}(N_B = n \cap n > K_B)}{\mathbb{P}(N_B > K_B)} = \sum_{n\in\{K_B+1,K_B+2,\dots\}} n\frac{\mathbb{P}(N_B = n)}{\mathbb{P}(N_B > K_B)}. \tag{A6}$$

Again, by substituting (A6) in (A5), we get the result:

$$\mathbb{E}(S^{\mathrm{BT}}) = \sum_{n\in\{K_B+1,K_B+2,\dots\}} n\mathbb{P}(N_B = n) - K_B\mathbb{P}(N_B > K_B) \tag{A7}$$

The results for the expected number of slots used for transportation towards destination C are determined in a similar way as (A2) – (A7):

$$\mathbb{E}(S^{\mathrm{CR}}) = \sum_{n\in\{1,2,\dots,K_{\mathrm{C}}\}} n\mathbb{P}(N_{\mathrm{C}} = n) + K_{\mathrm{C}}\mathbb{P}(N_{\mathrm{C}} > K_{\mathrm{C}}) \tag{A8}$$

$$\mathbb{E}(S^{\mathrm{CT}}) = \sum_{n\in\{K_{\mathrm{C}}+1,K_{\mathrm{C}}+2,\dots\}} n\mathbb{P}(N_{\mathrm{C}} = n) - K_{\mathrm{C}}\mathbb{P}(N_{\mathrm{C}} > K_{\mathrm{C}}) \tag{A9}$$

Substituting (A4) and (A7) – (A9) in (A1) gives the expected costs for a given value of K_{B} (and $K_{\mathrm{C}} = K - K_{\mathrm{B}}$). The optimal booking limit for rail slots used for destination B, i.e., the optimal value of K_{B} is then determined by solving

$$K_{\mathrm{B}}^* = \underset{K_{\mathrm{B}}}{\operatorname{argmin}}\, \mathbb{E}(c_T).$$

In the next section, we'll determine the optimal value K_{B}^* for a uniform distribution of the demand to destinations B and C.

A.2 Optimal slot reservation for uniformly distributed demand

For known distributions of N_{B} and N_{C}, the value of K_{B} that results in the lowest expected costs can be computed. E.g., for uniform distributions $N_{\mathrm{B}} \sim U(b_1, b_2), N_{\mathrm{C}} \sim U(c_1, c_2)$, that allow a total demand exceeding the train capacity, i.e., $b_2 + c_2 > K$, the expected number of rail slots used for destination B can be estimated by rewriting (A4) as

$$\mathbb{E}(S^{\mathrm{BR}}) = (K_B - b_1)\frac{b_1 + K_B + 1}{2(b_2 - b_1)} + K_B\frac{b_2 - K_B}{b_2 - b_1}$$

$$\mathbb{E}(S^{\mathrm{BR}}) = \frac{{K_{\mathrm{B}}}^2 + {b_1}^2 - 2K_{\mathrm{B}}b_2 + b_1 - K_{\mathrm{B}}}{2(b_1 - b_2)}. \tag{A10}$$

The expected number of units transported to destination B by truck can be determined by rewriting (A7) as

$$\mathbb{E}(S^{\mathrm{BT}}) = (b_2 - K_{\mathrm{B}})\frac{b_2 + K_{\mathrm{B}} + 1}{2(b_2 - b_1)} - K_{\mathrm{B}}\frac{b_2 - K_{\mathrm{B}}}{b_2 - b_1}$$

$$\mathbb{E}(S^{\mathrm{BT}}) = -\frac{K_{\mathrm{B}}^2 + b_2^2 - 2K_{\mathrm{B}}b_2 + b_2 - K_{\mathrm{B}}}{2(b_1 - b_2)}. \tag{A11}$$

Similarly, we can rewrite (A8) and (A9) as

$$\mathbb{E}(S^{\mathrm{CR}}) = \frac{{K_{\mathrm{C}}}^2 + {c_1}^2 - 2K_{\mathrm{C}}c_2 + c_1 - K_{\mathrm{C}}}{2(c_1 - c_2)}, \tag{A12}$$

$$\mathbb{E}(S^{\mathrm{CT}}) = -\frac{K_{\mathrm{C}}^2 + c_2^2 - 2K_{\mathrm{C}}c_2 + c_2 - K_{\mathrm{C}}}{2(c_1 - c_2)}. \tag{A13}$$

Combining and rewriting (A1), (A10) – (A13) gives

$$\mathbb{E}(c_T) = c_{\mathrm{BR}}\mathbb{E}(S^{\mathrm{BR}}) + c_{\mathrm{CR}}\mathbb{E}(S^{\mathrm{CR}}) + c_{\mathrm{BT}}\mathbb{E}(S^{\mathrm{BT}}) + \quad c_{\mathrm{CT}}\mathbb{E}(S^{\mathrm{CT}}). \tag{A14}$$

By using $K_{\mathrm{C}} = \ K - \ K_{\mathrm{B}}$ and finding the minimum of (A14), we can find the analytical optimal value for $K_{\mathbf{B}}$:

$$K_{\mathrm{B}}^* = \frac{\left(c_2 - K + \frac{1}{2}\right)(c_{\mathrm{CR}} - c_{\mathrm{CT}})(b_1 - b_2) + \left(b_2 + \frac{1}{2}\right)(c_{\mathrm{BR}} - c_{\mathrm{BT}})(c_1 - c_2)}{(\mathrm{b}_1 - \mathrm{b}_2)(\mathrm{c}_{\mathrm{CR}} - \mathrm{c}_{\mathrm{CT}}) + \ (\mathrm{c}_1 - \ \mathrm{c}_2)(\mathrm{c}_{\mathrm{BR}} - \ \mathrm{c}_{\mathrm{BT}})} \tag{A15}$$

The optimal value K_B^* is the upper limit for slots that can be used for transporting containers to destination B by rail. Next, we will compare the result of the proposed decision tree method with the analytical optimum.

Appendix B – decision tree for case study scenario 2

Splitting rules per node

Node 1: $t^{c}_{\text{available}}$ < Tue 12PM
Go to node 2, else node 3

Node 2: Lead time < 3 days
Select Barge 1, else Rail 1

Node 3: $t^{c}_{\text{available}}$ < Thu 12PM
Go to node 6, else select Rail 2

Node 6: Weight < 17.5 tonnes
Select trucking, else go to node 9

Node 9: $t^{c}_{\text{available}}$ < Wed 6AM
Select trucking, else go to node 11

Node 11: $t^{c}_{\text{available}}$ < Wed 12PM
Select Rail 2, else trucking