Research School for Operations Management and Logistics

Scheduling Drayage Operations in Synchromodal Transport

A.E. Pérez Rivera, M.R.K. Mes

Beta Working Paper series 531

BETA publicatie	WP 531 (working paper)
ISBN	
ISSN	
NUR	
Eindhoven	June 2017

Scheduling Drayage Operations in Synchromodal Transport

Arturo E. Pérez Rivera and Martijn R.K. Mes

Department of Industrial Engineering and Business Information Systems
University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
{a.e.perezrivera,m.r.k.mes}@utwente.nl

Abstract. We study the problem of scheduling drayage operations in synchromodal transport. Besides the usual decisions to time the pick-up and delivery of containers, and to route the vehicles that transport them, synchromodal transport includes the assignment of terminals for empty and loaded containers. The challenge consists of simultaneously deciding on these three aspects while considering various resource and timing restrictions. We model the problem using mixed integer linear programming (MILP) and design a matheuristic to solve it. Our algorithm iteratively confines the solution space of the MILP using several adaptations, and based on the incumbent solutions, guides the subsequent iterations and solutions. We test our algorithm under different problem configurations and provide insights into their relation to the three aspects of scheduling drayage operations in synchromodal transport.

Keywords: Drayage operations, synchromodal transport, matheuristic

1 Introduction

During the last years, intermodal transport has received increased attention from academic, industrial, and governmental stakeholders due to potential reductions in cost and environmental impact [9]. To achieve such benefits, these stakeholders have proposed new forms of organizing intermodal transport. One of these new initiatives is synchromodality, which aims to improve the efficiency and sustainability of intermodal transport through flexibility in the choice of mode and in the design of transport plans [11]. However, the potential benefits of any new form of intermodal transport depend to a great extent on the proper planning of drayage operations, also known as pre- and end-haulage or first and last-mile trucking. Drayage operations, which account for 40% of the total transport costs in an intermodal transport chain [5], are the first step where the synchromodal flexibility in transport mode can be taken advantage of. In this paper, we study the scheduling of drayage operations of intermodal transport considering terminal assignment (i.e., long-haul mode) decisions.

Drayage operations in intermodal transport include delivery and pick-up requests of either empty or loaded containers, to and from a terminal where long-haul modes arrive and depart. These operations occur, for example, at a Logistic

Service Provider (LSP) handling both import and export containers. The planner scheduling drayage operations must decide upon the time to fulfill each request and the route of the vehicles that will carry out all requests. In synchromodality, the planner must also decide to which terminal to bring a loaded container and to which terminal or customer to bring an empty container. All these decisions must be made simultaneously, considering constraints such as time-windows for requests, terminals, containers, trucks, and decoupling of requests for the delivery of an empty container and the subsequent pickup of a loaded one (and vice versa). Furthermore, the schedule of these requests is allowed to change as new information arrives (e.g., requests, delays, etc.). In such a dynamic environment, making assignment, timing, and routing decisions together is difficult [5]. Nonetheless, scheduling drayage operations with an integrated approach can bring significant savings [1].

In this paper, we develop an integrated approach to make assignment, timing, and routing decisions of drayage operations dynamically. First, we categorize the drayage requests in synchromodality and analyze their relations. With our categorization, we identify challenges and opportunities for scheduling methods. Second, we formulate the problem as a Mixed-Integer Linear Programming (MILP) model based on our categorization of requests. Third, we present several adaptations to the MILP model and design a heuristic algorithm around them to schedule drayage operations and update the schedule as new requests arrive.

2 Literature Review

We briefly review the literature about scheduling drayage operations in intermodal transport. We examine the characteristics of the proposed models and study their applicability to our problem. We finalize by stating our contribution.

Most studies about scheduling drayage operations use mathematical programming. This technique allows researchers to model various problem characteristics at the price of high computational complexity. For this reason, researchers consider one problem characteristic at a time. For example, studies that consider more than one terminal, such as [10] and [1], assume a homogenous fleet. Studies that consider a flexible origin or destination for some requests, such as [2] and [6], consider only one terminal. Studies that do not assume a homogenous fleet, such as [7], avoid other constraints such as request time-windows. All in all, mathematical programming can relate various problem attributes to optimal decisions but requires further developments to handle the actual scheduling.

There is a variety of approaches available to solve the actual scheduling of drayage operations. There are sequential approaches, such as [2] and [10], that pair delivery and pickup customers before the routing. There are also integrated approaches, such as [1,6,12,14], which handle paring (i.e, scheduling) and routing decisions simultaneously. Particularly, these integrated approaches show that a combination of parts of the mathematical problems with other heuristics perform well in solving the problem. Finally, the majority of approaches focuses on "one plan" per day with no re-planning, except for [5], which re-schedules when the

problem conditions change. Naturally, dynamic re-scheduling is another attribute of drayage operations that increases the complexity of the problem.

Although drayage operations contribute significantly to the total costs of intermodal transport [8], research on these operations has been limited in modeling considerations and solution approaches [3]. The need for dynamic scheduling methods for intermodal routing that take into account multiple attributes of the problem has been recognized [3]. For these reasons, our contribution to the literature is two-fold: (i) we model many attributes of the scheduling of drayage operations in synchromodal transport as an integrated MILP with various adaptations, and (ii) we develop a dynamic matheuristic to solve the model.

3 Problem Description

We study the problem of scheduling drayage requests in a synchromodal network with the objective of minimizing routing and terminal (i.e., long-haul mode) assignment costs. There are three simultaneous decisions: (i) timing the execution of requests, (ii) routing the vehicles that carry out the requests, and (iii) assigning long-haul terminals (or customers) to the requests. These decisions are subject to the characteristics of the requests, available trucks, available containers, and terminals. Requests are characterized by customer location, type of truck (e.g., driver clearance, chasis, trailer, etc.), type of container (e.g., size, security, refrigeration, etc.), time-window, service (i.e., loading, unloading) time, and decoupling allowance. Trucks are characterized by start and end location, type, maximum working time, setup cost, and variable cost. Containers are characterized by location, type, and amount. Terminals are characterized by location, time-window, and an assignment cost that represents costs for using a certain long-haul mode, container storage, etc.

The terminal assignment cost and the various request attributes in synchromodality enrich the common drayage operations in intermodal transport. To analyze this enrichment, we classify the requests into pre-haulage and end-haulage. In a pre-haulage request, an empty container is brought to a customer location and subsequently (after loading) brought to one of the long-haul terminals. In an end-haulage request, a loaded container is brought to a customer location and subsequently (after unloading) brought to a terminal for storage or to another customer who has a container-compatible pre-haulage request. Some of these requests allow *decoupling*, which means that a truck delivering an empty (or loaded) container does not need to wait for it to be loaded (or unloaded) and that another truck can pick up the loaded (empty) container later on. We refer to all possible pre- and end-haulage requests as *jobs* in the remainder of the paper. We now elaborate on the job configurations.

In drayage operations, there are various job configurations as seen in Fig. 1. These configurations arise due to different contractual agreements, types of freight, types of resources, etc. In the complete job configurations of the end-haulage, the origin is a fixed terminal, but the destination can be either a given terminal (or customer) or one of multiple terminals (or customers), as seen in

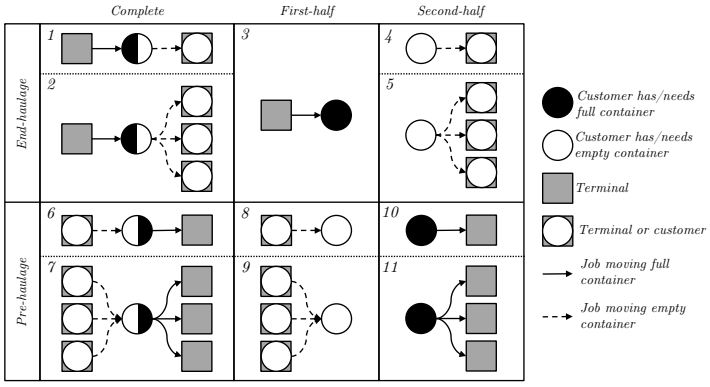

Fig. 1. Possible job configurations in synchromodal transport

Types 1 and 2, respectively. If decoupling is allowed, one can divide the end-haulage job configurations into first- and second-half, as seen in Types 3, 4, and 5. In the complete job configurations of the pre-haulage, the origin can be a given terminal (or customer) and the destination a given terminal, or the origin can be one of multiple terminals (or customers) and the destination one of multiple terminals, as seen in Types 6 and 7, respectively. Once more, if decoupling is allowed, the pre-haulage configurations can be divided into first- and second-half, as seen in Types 8, 9, 10, and 11. Due to the full-truckload and multi-resource nature of the job configurations, some challenges and opportunities arise. For example, executing some job configurations after each other (e.g., Type 3 followed by Type 10) will require an *empty movement* of a truck, i.e., truck moving without a container. In another example, executing some job configurations after each other (e.g., Type 1 followed by Type 6) can allow the truck to skip the visit to a terminal (e.g., supersede the use of an empty container at a terminal). In such opportunities, some job configurations can be merged to reduce the decision complexity as also proposed by [7]. Using this job categorization, we formulate an MILP model that captures the challenges and opportunities in drayage operations in the following section. In the remainder of this paper, when we talk about job *types*, we refer to the configurations seen in Fig. 1.

4 MILP formulation

Using the previous categorization of jobs, we construct two directed graphs $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ and $\mathcal{G}' = (\mathcal{V}, \mathcal{A}')$, which have the same nodes $\mathcal{V}$ but different arcs $\mathcal{A}$ and $\mathcal{A}'$. Nodes $\mathcal{V}$ represent all locations related to jobs and trucks: $\mathcal{V} = \mathcal{V}^{\mathrm{R}} \cup \mathcal{V}^{\mathrm{D}} \cup \mathcal{V}^{\mathrm{B}} \cup \mathcal{V}^{\mathrm{F}}$. Specifically, $\mathcal{V}^{\mathrm{R}}$ contains all job locations (i.e., request locations), $\mathcal{V}^{\mathrm{D}}$ all terminal locations, $\mathcal{V}^{\mathrm{B}}$ all beginning location of trucks and $\mathcal{V}^{\mathrm{F}}$ their finishing location. All nodes in $\mathcal{V}$ are indexed with i and j. Arcs A and $\mathcal{A}'$ are built to distinguish the assignment and the routing decisions, respectively. Arcs in A include all *job-arcs* between two nodes (connections as in Fig. 1). Job-arcs are connections between nodes that comply with all resource and long-haul mode constraints. We define $\delta^-(r) = \{j : (j, r) \in \mathcal{A}\}$ and $\delta^+(r) = \{j : (r, j) \in \mathcal{A}\}$ as

the sets of nodes that form job-arcs that are incoming to, and outgoing from, node $r \in \mathcal{V}^{\mathrm{R}}$, respectively. Arcs in $\mathcal{A}'$ include all *routing-arcs*. These arcs follow a similar logic as in the VRP with time-windows formulation of [4].

In $\mathcal{V}^{\mathrm{R}}$, each job is represented as a single node. We index nodes in $\mathcal{V}^{\mathrm{R}}$ with r. In $\mathcal{V}^{\mathrm{D}}$, each terminal $d \in \mathcal{U}^{\mathrm{D}}$ is represented as N_d identical nodes, in order to keep track of arrival times in the model. The set $\mathcal{U}^{\mathrm{D}}$ is the set with unique terminal nodes. We index both sets with d. Each node $i \in \mathcal{V}^{\mathrm{R}} \cup \mathcal{V}^{\mathrm{D}}$ has a service time S_i (i.e., time for loading, unloading, coupling, or decoupling a container), as well as a time-window described by an earliest arrival time E_i and a latest arrival time L_i. For nodes $r \in \mathcal{V}^{\mathrm{R}}$, which represent jobs, D_r gets a value of one if decoupling is allowed and zero otherwise. Traveling time between nodes i and j is denoted with $T_{i,j}$. Note that, for the identical nodes of each terminal, all time parameters are the same and traveling times between them are zero. Note also that two jobs can be at the same location, and thus traveling time between them is also zero. However, service times and time-windows can be different, depending on the job type.

To carry out all jobs, there is a fleet of heterogeneous trucks $\mathcal{K}$. The trucks that can carry out job $r \in \mathcal{V}^{\mathrm{R}}$ are represented with $\widetilde{K}(r) \subseteq \mathcal{K}$. Each truck begins its route in node $B_k \in \mathcal{V}^{\mathrm{B}}$ and finishes its route in node $F_k \in \mathcal{V}^{\mathrm{F}}$. All trucks have a maximum working time T_k^{K}. Truck movements are modeled using the binary variable $x_{i,j,k}$, which gets a value of 1 if node j is visited immediately after node i by truck k, and 0 otherwise. Note that truck movements can either be to carry out a request (i.e., truck has an empty or loaded container) or to reposition the truck (i.e., no container). To model time in the movements of trucks, we use the auxiliary variable w_i, which represents the time at which the chosen truck arrives at node i. Note that w_i does not depend on k since each job can be done by only one truck and we duplicate the terminal nodes such that each node is again visited by only one truck.

The goal is to perform all jobs, within their time-window, while minimizing routing and terminal assignment costs. To model the routing costs, we introduce (i) a fixed cost C_k^{F} for using truck $k \in K$ and (ii) a variable cost $C_{i,j,k}^{\mathrm{V}}$ for its movement over arc $(i,j) \in \mathcal{A}'$. To model the terminal assignment costs, we introduce a cost $C_{r,d}^{\mathrm{D}}$ for assigning terminal $d \in \mathcal{V}^{\mathrm{D}}$ to job $r \in \mathcal{V}^{\mathrm{C}}$. Using the parameters and variables above, the optimization goal can be achieved solving the mathematical program shown in (1).

$$\min z = \sum_{k \in \mathcal{K}} \left(C_k^{\mathrm{F}} \cdot \sum_{j \in \delta'^{+}(B_k)} x_{B_k,j,k} \right) + \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}'} C_{i,j,k}^{\mathrm{V}} \cdot x_{i,j,k} + \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{V}^{\mathrm{R}}} \sum_{d \in \delta^{+}(r) \cup \mathcal{V}^{\mathrm{D}}} C_{i,j}^{\mathrm{D}} \cdot x_{r,d,k} \tag{1a}$$

s.t.

$$\sum_{k \in \widetilde{\mathcal{K}}(r)} \sum_{j \in \delta^{+}(r)} x_{r,j,k} = 1, \ \forall\, r \in \mathcal{V}^{\mathrm{R}} \,|\, \delta^{+}(r) \neq \emptyset \tag{1b}$$

$$\sum_{k \in \widetilde{\mathcal{K}}(r)} \sum_{j \in \delta^{-}(r)} x_{j,r,k} = 1, \ \forall\, r \in \mathcal{V}^{\mathrm{R}} \,|\, \delta^{-}(r) \neq \emptyset \tag{1c}$$

$$(1 - D_r)\left(\sum_{j \in \delta^+(r)} x_{r,j,k} - \sum_{j \in \delta^-(r)} x_{j,r,k}\right) = 0, \tag{1d}$$
$$\forall\, r \in \mathcal{V}^{\mathrm{R}} \,\Big|\, \delta^+(r) \neq \emptyset \text{ and } \delta^-(r) \neq \emptyset, k \in \widetilde{K}(r)$$

$$\sum_{k \in \mathcal{K}} \sum_{j \in \delta'^+(r)} x_{r,j,k} = 1,\ \forall\, r \in \mathcal{V}^{\mathrm{R}} \tag{1e}$$

$$\sum_{k \in \mathcal{K}} \sum_{j \in \delta'^+(d)} x_{d,j,k} \leq 1,\ \forall\, d \in \mathcal{V}^{\mathrm{D}} \tag{1f}$$

$$\sum_{j \in \delta'^+(i)} x_{i,j,k} - \sum_{j \in \delta'^-(i)} x_{j,i,k} = 0,\ \forall\, i \in \mathcal{V}^{\mathrm{C}} \cup \mathcal{V}^{\mathrm{D}}, k \in \mathcal{K} \tag{1g}$$

$$E_i \leq w_i \leq L_i,\ \forall\, i \in \mathcal{V} \tag{1h}$$

$$\sum_{k \in \mathcal{K}} \left(x_{i,j,k} \cdot (w_i + S_i + T_{i,j} - w_j)\right) \leq 0,\ \forall\, i, j \in \mathcal{V} \tag{1i}$$

$$\sum_{k \in \mathcal{K}} \left(x_{B_k,j,k} \cdot T_{B_k,j}\right) \leq w_j,\ \forall\, j \in \mathcal{V} \tag{1j}$$

$$x_{i,F_k,k} \cdot \left(w_i + S_i + T_{i,F_k} - T_k^{\mathrm{K}}\right) \leq 0,\ \forall\, i \in \delta'^-(F_k), k \in \mathcal{K} \tag{1k}$$

$$\sum_{(i,j) \in \mathcal{A}'} x_{i,j,k} - M_k^{\mathrm{A}} \cdot \sum_{j \in \delta'^+(B_k)} x_{B_k,j,k} \leq 0,\ \forall\, k \in \mathcal{K} \tag{1l}$$

$$\sum_{j \in \delta'^+(B_k)} x_{B_k,j,k} \leq M_k^{\mathrm{K}},\ \forall\, k \in \mathcal{K} \tag{1m}$$

$$\sum_{i \in \delta'^-(F_k)} x_{i,F_k,k} - \sum_{j \in \delta'^+(B_k)} x_{B_k,j,k} = 0,\ \forall\, k \in \mathcal{K} \tag{1n}$$

$$x_{i,j,k} = 0,\ \forall\, i \in \mathcal{V}^{\mathrm{B}} \setminus \{B_k\}, j \in \mathcal{V}^{\mathrm{R}} \cup \mathcal{V}^{\mathrm{D}}, k \in \mathcal{K} \tag{1o}$$

$$x_{i,j,k} = 0,\ \forall\, i \in \mathcal{V}^{\mathrm{R}} \cup \mathcal{V}^{\mathrm{D}}, j \in \mathcal{V}^{\mathrm{F}} \setminus \{F_k\}, k \in \mathcal{K} \tag{1p}$$

$$w_i \in \mathbb{R},\ \forall\, i \in \mathcal{V} \tag{1q}$$

$$x_{i,j,k} \in \{0, 1\},\ \forall\, i, j \in \mathcal{V}, k \in \mathcal{K} \tag{1r}$$

The objective is to minimize the total costs z as shown in (1a). Constraints (1b) state that only one incoming job-arc can be used for job r. Note that it is possible that job r does not require incoming job-arcs (e.g., Type 10), and thus $\delta^+(r) = \emptyset$. Similarly, (1c) ensure that only one outgoing job-arc can be used for job r. For jobs that have both incoming and outgoing job-arcs (i.e., Types 1, 2, 6, and 7), (1d) ensure that the same vehicle does both the incoming and outgoing job-arc if decoupling is not allowed for job r. Constraints (1e) ensure that all jobs $r \in \mathcal{V}^{\mathrm{R}}$ are carried out by one truck only. Constraints (1f) ensure that all terminal nodes $d \in \mathcal{V}^{\mathrm{D}}$ are visited at most once. Remind that a terminal has duplicate nodes for keeping track of time, meaning that the same terminal might be visited multiple times (e.g., for different jobs) but each time to a different duplicated node. Constraints (1g) ensure flow conservation, meaning that all nodes that are exited must be entered as well. The time-windows of jobs, terminals, and truck locations are enforced in (1h). Constraints (1i) and (1j) keep track of the

time variables. The maximum working time of trucks is guaranteed by (1k). Constraints (1l) and (1m) establish that each truck can only depart once from its starting location if it is used for doing jobs. In (1l), M_k^{A} works as a "big-M" parameter that can be initialized, for example, with $M_k^{\mathrm{A}} = |\mathcal{V}^{\mathrm{R}}| + |\mathcal{V}^{\mathrm{D}}| + 1$. However, it can also be used to *restrict* the number of routing-arcs that vehicle k can traverse, as we will explain in Section 5. In a similar way, the auxiliary parameter M_k^{K} can be used to restrict the use of vehicle k by setting $M_k^{\mathrm{K}} = 0$. Initially, we set $M_k^{\mathrm{K}} = 1, \forall k \in K$. Constraints (1n) state that each truck must end at its ending location if it has departed from its beginning location. Since the nodes $\mathcal{V}^{\mathrm{B}}$ and $\mathcal{V}^{\mathrm{F}}$ are used for modeling beginning and ending locations of trucks (i.e., not for carrying out jobs), we have to ensure that trucks do not visit them in any case, as shown in (1o) and (1p). Finally, Constraints (1q) and (1r) establish the domains of the variables.

Although the formulation above is not linear due to (1i) and (1k), we can linearize it by substituting these two with (2a) and (2b). An explanation on the logic behind these constraints can be found in [2].

$$w_i + S_i + T_{i,j}^T - (L_i + S_i + T_{i,j} - E_j) \cdot \left(1 - \sum_{k \in \mathcal{K}} x_{i,j,k}\right) \leq w_j \; \forall \; i,j \in \mathcal{V} \tag{2a}$$

$$w_i + S_i + T_{i,F_k} - (L_i + S_i + T_{i,F_k}) \cdot (1 - x_{i,F_k,k}) \leq L_{F_k}, \; \forall \; i \in \delta^{'-}(F_k), k \in \mathcal{K} \tag{2b}$$

Our MILP formulation models the jobs as arcs that need to be traversed by the trucks. Another option to represent jobs in drayage operations is to model them as nodes. Modeling jobs as nodes reduces the size of the graph if some of the nodes are fixed beforehand [1]. However, flexible jobs (such as ours with the terminal assignment) cannot be collapsed into a single-node [12] and the gains of an integrated approach are harder to obtain when modeling jobs as nodes [1]. Although modeling jobs as arcs come with the price of a larger graph, there are other opportunities to improve the formulation through valid inequalities and pre-processing, as we describe in the following section.

4.1 Valid Inequalities and Pre-Processing

Due to the full-truckload nature of our problem, all jobs deal with at most one terminal either as the origin or the destination of a container. This means that a truck carrying out jobs will never visit more than two terminals consecutively. Since we model the requests and terminals as separate nodes, this means that not all arcs between terminal nodes will be traversed. An arc between two terminal nodes will only be traversed when delivering a container to the first one, and picking up a container from the second one. Thus, we can use a bound M^{DE} on them as shown in (3).

$$\sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{V}^{\mathrm{D}}} \sum_{j \in \mathcal{V}^{\mathrm{D}}} x_{i,j,k} \leq M^{\mathrm{DE}} \tag{3a}$$

$$M^{\mathrm{DE}} = \sum_{r \in \mathcal{V}^{\mathrm{R}}} \sum_{d \in \mathcal{U}^{\mathrm{D}}} B_{r,d} \,\Big|\, B_{r,d} = \begin{cases} 1 & \text{if } d \in \delta^-(r) \\ 0 & \text{otherwise} \end{cases} \tag{3b}$$

In addition to the bound on the number of arcs between all terminal nodes $\mathcal{V}^{\mathrm{D}}$, we can bound the traversed arcs between replicated nodes of a terminal using a similar logic. We define the set $\mathcal{V}_d^{\mathrm{DR}} \subseteq \mathcal{V}^{\mathrm{D}}$ as the set containing all duplicated nodes of terminal $d \in \mathcal{U}^{\mathrm{D}}$. We put a bound M_d^{DI} for each unique terminal node $d \in \mathcal{U}^{\mathrm{D}}$ as shown in (4).

$$\sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{V}_d^{\mathrm{DR}}} \sum_{j \in \mathcal{V}_d^{\mathrm{DR}}} x_{i,j,k} \leq M_d^{\mathrm{DI}}, \ \forall\, d \in \mathcal{U}^{\mathrm{D}} \tag{4a}$$

$$M_d^{\mathrm{DI}} = \sum_{r \in \mathcal{V}^{\mathrm{R}}} \sum_{i \in \mathcal{V}_d^{\mathrm{DR}}} B_{r,i} \,\middle|\, B_{r,i} = \begin{cases} 1 & \text{if } i \in \delta^-(r) \\ 0 & \text{otherwise} \end{cases}, \ \forall\, d \in \mathcal{U}^{\mathrm{D}} \tag{4b}$$

Taking advantage that our problem deals with jobs that have at most one origin and at most one destination, we can compute a minimum traveling distance and traveling time to fulfill all jobs by choosing the origin and destination with the shortest distance and time, respectively. Using this information, we can calculate the minimum number M^{LK} of trucks needed (since trucks have a maximum working time) and a lower bound on the routing costs M^{LC}. Furthermore, using a constructive heuristic (e.g., the one we benchmark to in Sect. 6), we can find upper bounds M^{UK} and M^{UC} for the number of trucks needed and the routing costs, respectively. Thus, we can limit the number of trucks as shown in (5) and the routing costs as shown in (6).

$$M^{\mathrm{LK}} \leq \sum_{k \in K} \sum_{j \in \delta'^{+}(B_k)} x_{B_k,j,k} \leq M^{\mathrm{UK}} \tag{5}$$

$$M^{\mathrm{LC}} \leq \sum_{k \in K} \left(C_k^F \cdot \sum_{j \in \delta'^{+}(B_k)} x_{B_k,j,k} \right) + \sum_{k \in K} \sum_{(i,j) \in \mathcal{A}'} C_{i,j,k}^{\mathcal{V}} \cdot x_{i,j,k} \leq M^{\mathrm{UC}} \tag{6}$$

The last adaptation we introduce is the pre-processing of time-windows. In our model, there are duplicated nodes (i.e., same location, service time, and time-window) for each terminal to keep track of time. However, each duplicated terminal node can only be used for one job. Since we duplicate a terminal for each job that might use that terminal, we can use the time-window of the job to reduce the time-window of the duplicated node for that terminal. As an example, consider Fig. 2. In this figure, we see a job of Type 1 that requires a full container from terminal d and delivers an empty container to terminal d'. In order to carry out this job within its time-window $[E_r, L_r]$, the full container must be put on a truck and travel from terminal d anywhere between $[E_r - (S_d + T_{d,r}), L_r - (S_d + T_{d,r})]$. Similarly, after unloading the container, the empty container can arrive to terminal d' anywhere between $[E_r + (S_r + T_{r,d'}), L_r + (S_r + T_{r,d'})]$. We can repeat this logic with all jobs, their associated (possible) terminals, and the duplicated nodes for those terminals.

The benefit of the aforementioned enhancements of the MILP is twofold. First, the valid inequalities tighten the feasible solution. Second, the time-window pre-processing breaks the symmetry in MILP solutions introduced by the duplicated terminal nodes. However, these modifications are sufficient to solve only small problems. In the following section, we elaborate on further adaptations of the MILP that can allow it to be applied to larger problems.

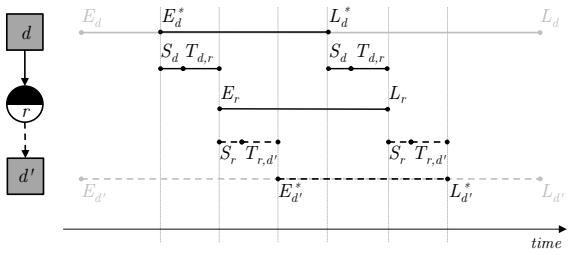

Fig. 2. Example of pre-processing of time-windows for a job Type 1

5 Matheuristics

In our problem, MILP solvers are able to find a good feasible solution fast, but struggle on improving it further or in proving its optimality. In this section, we design three adaptations to the MILP that are aimed to help a solver find good feasible solutions faster. Furthermore, we design two matheuristics: (i) a static matheuristic to solve a single instance of the problem using Math-Heuristic Operators (MHOs), and (ii) a dynamic matheuristic to solve a re-planning instance of the problem using Fixing Criteria (FCs), as shown in the pseudo-code of Algorithms 1 and 2, respectively. We now elaborate on the MHOs, FCs, and parts of each algorithm.

Algorithm 1 Static Matheuristic

Require: Graph $\mathcal{G}$ and associated parameters
Initialize best solution
while Stopping criterion not met **do**
  Get MHOs (7), (8), and (9)
  Build adapted MILP
  Solve adapted MILP
  if Current solution $\leq$ Best solution **then**
    Best solution = Current Solution
  end if
end while
return Best solution

Algorithm 2 Dynamic Matheuristic

Require: Re-planning trigger and current schedule
Determine current state
Fix trucks with FCs (10) and (11)
Determine re-planning jobs
Build $\mathcal{G}$ and associated parameters
Run Algorithm 1
return Solution

5.1 Static Matheuristic

Our static matheuristic uses three adaptations to the MILP, iteratively and in a local-search fashion. These adaptations, denoted by MHOs, are basically additional constraints in the MILP that can be seen as cutting planes that reduce the feasible space. Since our formulation results in a lot of arcs, our MHOs focus on fixing those arcs in an intuitive way. We now explain each MHO in more detail.

MHO 1: For N^{M1} random jobs $r \in \mathcal{V}^{\mathrm{R}}$, we limit the number of feasible job-arcs to at most two, i.e., $|\delta^-(r)| \leq 2$ and $|\delta^+(r)| \leq 2$. These arcs are from (or to) the two closest locations (i.e., shortest traveling time). In other words, all remaining

job-arcs are cut out, as shown in (7).

$$x_{j,r,k} = 0, \forall k \in \mathcal{K}, j \in \delta^-(r) \setminus \left\{i, i'\right\} \Big| i = \underset{j \in \delta^-(r)}{\arg\min}\, T_{j,r} \text{ and } i' = \underset{j \in \delta^-(r) \setminus \{i\}}{\arg\min}\, T_{j,r} \quad (7a)$$

$$x_{r,j,k} = 0, \forall k \in \mathcal{K}, j \in \delta^+(r) \setminus \left\{i, i'\right\} \Big| i = \underset{j \in \delta^+(r)}{\arg\min}\, T_{r,j} \text{ and } i' = \underset{j \in \delta^+(r) \setminus \{i\}}{\arg\min}\, T_{r,j} \quad (7b)$$

MHO 2: For N^{M2} times, the arc between a job r of Type 2 and a job r' of Type 7 with the minimum traveling time is fixed. Remind that the arc is feasible when $r \in \delta^-(r')$ and $r' \in \delta^+(r)$, and thus the fixing of a pair of jobs r and r' can be done as shown in (8).

$$\sum_{k \in \mathcal{K}} x_{r,r',k} = 1 \Big| r = \underset{j \in \delta^-(r')}{\arg\min}\, T_{j,r'} \quad (8)$$

MHO 3: For N^{M3} random jobs $r \in \mathcal{V}^{\mathrm{R}}$, we fix the feasible job-arcs from (or to) the closest location (i.e., shortest traveling time), as shown in (9).

$$\sum_{k \in \mathcal{K}} x_{i,r,k} = 1 \Big| i = \underset{j \in \delta^-(r)}{\arg\min}\, T_{j,r} \quad \text{and} \quad \sum_{k \in \mathcal{K}} x_{r,i,k} = 1 \Big| i = \underset{j \in \delta^+(r)}{\arg\min}\, T_{r,j} \quad (9)$$

Assigning a value to the parameters $N^{\mathrm{M}*}$ requires tuning (see Sect. 6.2). In general, the larger the value of $N^{\mathrm{M}*}$ the smaller the problem becomes for the solver, but the higher the chance of ruling out the global optimum.

5.2 Dynamic Matheuristic

Our dynamic matheuristic builds upon the static and is used for re-planning situations (e.g., new jobs arrived, delays, etc.). Jobs sequences that are being executed during the re-planning trigger are completed (i.e., no preemption). For jobs that have not been started yet, we have the option to use the previous plan or to re-plan them. In the first option, we fix jobs in current truck routes using two Fixing Criteria (FC). The idea of these criteria is to identify good routes from the current schedule that can be kept in the new plan. In the second option, we use the static matheuristic to build a new schedule. The fixing of a route means that the routes (i.e., job sequences) will be preserved, but the time at which and the truck by which they are executed is left flexible. This time flexibility allows new jobs to be added to the trucks already being used and also to handle delays.

FC 1: We fix N^{F1} routes k^{C} from the current schedule x^{C} with the largest number of jobs. We define $\mathcal{F}^1(k^{\mathrm{C}})$ as the set of arcs (i,j) that fulfill this criteria, as shown in (10), and fix the routes by $\sum_{k \in \mathcal{K}} x_{i,j,k} = 1, \forall (i,j) \in \mathcal{F}^1(k^{\mathrm{C}})$.

$$\mathcal{F}^1(k^{\mathrm{C}}) = \left\{ (i,j) \in \mathcal{A} : x^{\mathrm{C}}_{i,j,k^{\mathrm{C}}} = 1, k^{\mathrm{C}} = \underset{k' \in \mathcal{K} | \sum_{j \in \mathcal{V}} x^{\mathrm{C}}_{B_{k'},j,k} = 1}{\arg\max} \sum_{i \in \mathcal{V}^{\mathrm{R}}} x^{\mathrm{C}}_{i,j,k} \right\} \quad (10)$$

FC 2: We fix N^{F2} routes with the shortest traveling time, similar to FC 1. We define $\mathcal{F}^2(k^{\mathrm{C}})$ as the set of arcs (i,j) that fulfill this criteria, as shown in (11).

$$\mathcal{F}^2(k^{\mathrm{C}}) = \left\{ (i,j) \in \mathcal{A} : x^{\mathrm{C}}_{i,j,k^{\mathrm{C}}} = 1, k^{\mathrm{C}} = \underset{k' \in \mathcal{K} | \sum_{j \in \mathcal{V}} x^{\mathrm{C}}_{B_{k'},j,k} = 1}{\arg\min} \sum_{(i,j) \in \mathcal{A}'} x^{\mathrm{C}}_{i,j,k} T_{i,j} \right\} \quad (11)$$

Just as in the static matheuristic, the best value of the parameters $N^{\mathrm{F}*}$ depends on circumstances such as the current schedule, the instance $\mathcal{G}$, and the re-planning trigger. In the following, we present a brief proof-of-concept of the algorithms just described.

6 Numerical Experiments

We test our solution approach in two numerical experiments. First, we examine the benefits that our adaptations have in solving the MILP. Second, we test the gains of using our dynamic matheuristic compared to a benchmark heuristic. Our goal is to explore our approach and gain insights for further research.

6.1 Experimental Setup

We design 32 problem instances containing 25 jobs each. The location, time-window, and service time for each job is obtained from the first 25 customers of the Solomon instances for the VRPTW [13]. We use the first eight instances of four categories in [13]: C1, C2, R1, and R2, where C stands for clustered locations, R for random locations, 1 for short time-windows, and 2 for long-time windows. For each instance, there are 25 homogeneous trucks to guarantee there is a feasible solution. For each truck, the fixed cost is 1000 and the traveling time and variable cost is equal to the Euclidean distance between two locations.

The job configurations for all instances are shown in Table 1. These job configurations are based on the average drayage operations of a Dutch LSP in the Eastern part of The Netherlands. This LSP has three terminals that we use as follows. Terminal 1 is located in the same location as the Depot in the corresponding Solomon instance and has a terminal assignment cost of 500. Terminal 2 and Terminal 3 are located at (60,60) and (10,10), which are points along the diagonal in the Euclidean space that are close to the most distant customers from the geographical center of each of the Solomon instances. The assignment costs vary per instance, and are defined as $500 - 2\beta$, where β is the length of the diagonal formed by the extremes of the corresponding Solomon instance. The rationale is to make the distant terminals worth "assigning" if a job is within half of the diagonal. The time-window of each terminal, and maximum working time for each trucks, is twice the time-window of the depot in the corresponding Solomon instance. Terminal 1 is the beginning and finishing location of nine trucks, and Terminals 2 and 3 of eight trucks each.

In both the static and dynamic experiments, we compare to the use of a benchmark heuristic that follows the logic from [2]. First, each job of Type 2 is

Table 1. Job configuration for all instances

Characteristic	Job Type										
	1	2	3	4	5	6	7	8	9	10	11
Number of jobs	2	3	2	2	2	2	4	3	3	1	1
Jobs decoupling D_r	1	1	-	-	-	1	1	1	-	-	-

paired with a job of Type 7 that incurs the minimum variable cost considering all constraints. Note that the origin of the full container of Type 2 is known, thus the pairing occurs in the destination of the empty container of Type 2 with the source of the empty container of Type 7. Subsequently, the closest terminal is assigned to the full container of Type 7. The paired jobs are sorted in non-decreasing route distance and scheduled using a cheapest insertion method. This method schedules the paired jobs in the position of the route that yields the lowest routing cost. All remaining jobs are then scheduled with a similar cheapest insertion method. For the jobs that have a flexible source or destination of a container, all combinations of sources and destinations are examined and the position with the cheapest routing and terminal assignment cost is chosen. To use this heuristic dynamically, the job sequences that are being executed during the re-planning trigger are fixed (i.e., no preemption of the current schedule). Jobs that have not been started and that are not in a non-preemptive sequence are re-scheduled with the steps described before. We now describe each experiment in detail and present their results.

6.2 Static Experiments

In the static experiments, we test the effect that the Valid Inequalities (VIs), Time-Window Pre-Processing (TWPP), and the three Math-Heuristic Operators (MHOs) have on the total costs. For each of the eight instances in categories C1, C2, R1, and R2, we test the MILP without any modification, the MILP with the VIs, the MILP with the TWPP, and the MILP with the VIs, TWPP, and each of the MHOs. We use CPLEX 12.6.3 (via the C API) with a limit of 300 seconds and a warm-start given by the solution of the Benchmark Heuristic (BH). For MHO 1 and 3, we perform nine iterations; and at each iteration, we choose seven random jobs in MHO 1 and one random job in MHO 3. MHO 2 has no randomness, so we perform only three iterations and we fix 1, 2, and 3 jobs respectively. The "random" settings are arbitrary, since we just test their usefulness rather than tuning them. We show the aggregated results in Table 2.

In Table 2, four interesting observations arise. First, the VIs do not improve the solution of the MILP but the TWPP does. Second, MHO 3 performs the best in the clustered (C) instances and MHO 1 performs best in the random (R) ones. It is reasonable that MHO 1 performs better in R instances than C ones because it increases the chance of assigning the closest origin and destination to each job in a network with more disperse locations. Third, MHO 2 performs worse than the other MHOs. It seems that choosing a job as the origin/destination

Table 2. Total costs for various MILP adaptations

Instances	BH	MILP	VIs	TWPP	MHO 1	MHO 2	MHO 3
C1	77,960	77,926	77,960	76,924	76,829	77,926	75,189
C2	52,904	52,882	52,904	52,049	51,841	52,078	50,802
R1	111,087	111,078	110,904	107,649	107,254	107,647	107,736
R2	50,500	50,435	50,500	50,497	50,255	50,500	50,378

of an empty container (i.e., logic of the BH) is not better than allowing a job or a terminal to be origin and destination, as MHO 1 and 3 allow. Fourth, in instances C1, C2, and R1, our adaptations to the MILP result in savings (from the BH) between 3-4%, but in instances R2 there are no noticeable savings. It seems that in R2, which has long time-windows and longer traveling times, the cheapest insertion and job-pairing nature of the BH results in good solutions.

6.3 Dynamic Experiments

In the dynamic experiments, we test the effect of the Fixing Criteria FC 1 and FC 2. In addition, we test a no-fixing criteria FC 0 meaning that all non-preemptive jobs can be re-scheduled. The problem instances are similar to the static experiments. We consider five stages for re-scheduling after the initial planning of the first 25 jobs of each Solomon instance (i.e., the static setup). These five stages are uniformly distributed within the first half of the trucks and terminals maximum working time (i.e., half a day). At each stage, five new jobs are revealed, which correspond to the next five jobs in each Solomon instance, and whose time-window is increased proportionally to the stage to guarantee they occur after they are revealed. Non-preemptiveness applies to all jobs scheduled in a truck before the next stop at a terminal. For the static matheuristic within the dynamic matheuristic, we use MHO 3 for C instances and MHO 1 for R instances. We define the number of truck routes to fix for FC1 and FC2 as a percentage of the available truck routes: $N^{F*} = (0.1, 0.5)$. The results are shown in Fig. 3 and 4.

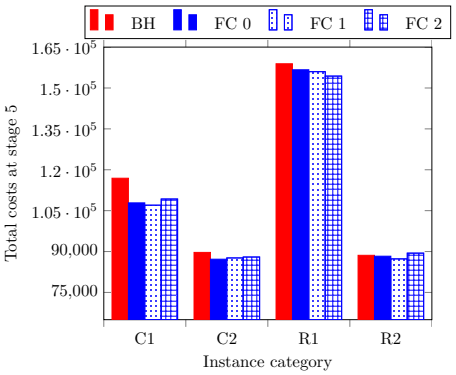

Fig. 3. Comparison FCs at last stage

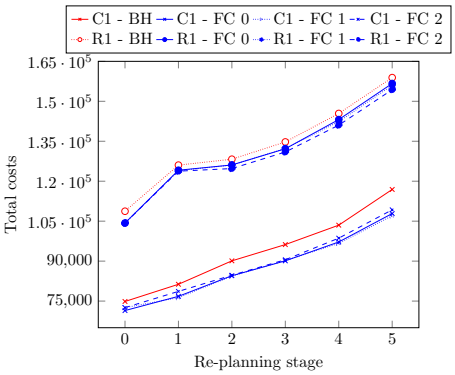

Fig. 4. Performance of best FC per stage

In Fig. 3, we observe that in the last re-planning stage (i.e., costs over the entire day), our dynamic matheuristic achieves significant savings compared to the BH in instances C1, C2 and R1 (around 8%, 3%, and 3% respectively). The winning FC, however, seems to vary per instance type. It is reasonable that fixing routes with the largest number of jobs (i.e., FC 1) is good in C1 since these instances contain closely located jobs with tight time-windows. Furthermore, it seems also reasonable that in instances with disperse located jobs such as R1, fixing routes with good traveling times (i.e., FC 2) is better. We focus on these two instance categories in Fig. 4. For R1 we observe that FCs 0 and 1have similar performance and that FC 2 starts differentiating more from the other FCs and the BH towards the last stages. For C1 we observe similarly that FCs 0 and 1 have comparable performance, and that the gap to the BH seems to widen towards the last stages.

6.4 Discussion

In the static experiments, we observe that the MHOs help obtaining a better MILP solution than the VIs and the TWPP. We observe also that the performance of the MHO depends on the problem settings. In the dynamic experiments, we observe that the dynamic matheuristic outperforms the BH but the performance of the FCs therein depends again on the problem settings. Although these experiments serve as a proof-of-concept and give an indication of the gains to be expected, they have three limitations. First, we do not tune the parameters with respect to the problem setting. As described in the results, our MHOs and FCs have implicit distance and time effects on the solution and thus require tuning. Second, we do not adapt the algorithms towards previous iterations or stages. Our MHOs and FCs are analogous to neighborhood operators in local search heuristics, and thus mechanisms that adapt them can be beneficial to further guide the algorithm to better solutions. Third, we limit the computational time of the matheuristics and use a heuristic for the warm-start. The interaction of these two methods and the solver has a larger effect on some problems than others. We observed that for some instance categories, the solver was able to find improvements at every stage, but in some others failed to find a different solution than the heuristic within the allowed time. These limitations in our study, however, give rise to new research questions, specially in the combination of exact and heuristic approaches, i.e., matheuristics: how to tune the parameters, how to adapt the algorithm that uses the parameters after each iteration, and how to handle the interaction between solver and solution are examples of promising research lines.

7 Conclusion

We developed a MILP and a matheuristic to schedule drayage operations in synchromodal transport. Timing, routing, and long-haul terminal assignment

decisions are integrated and simultaneously considered. Dynamic scheduling is done as new information is revealed throughout the day.

Through numerical experiments, we studied the performance of our adaptations to the MILP model and fixing criteria in the matheuristic. Overall, we observed that the gains of our approach are dependent on problem attributes such as customer dispersion, time-window lengths, and dynamic re-planning. Further research in the relation between these characteristics and our matheuristics is needed. The proper handling of such relations is essential for scheduling drayage operations in synchromodal transport.

Acknowledgment: This research has been partially funded by the Dutch Institute for Advanced Logistics, DINALOG, under the project SynchromodalIT.

References

1. Braekers, K., Caris, A., Janssens, G.: Integrated planning of loaded and empty container movements. OR Spectrum 35(2), 457–478 (2013)
2. Caris, A., Janssens, G.: A local search heuristic for the pre- and end-haulage of intermodal container terminals. Computers & Operations Research 36(10), 2763 – 2772 (2009)
3. Caris, A., Macharis, C., Janssens, G.K.: Decision support in intermodal transport: A new research agenda. Computers in Industry 64(2), 105 – 112 (2013)
4. Cordeau, J.F., Laporte, G., Savelsbergh, M.W., Vigo, D.: Chapter 6 vehicle routing. In: Barnhart, C., Laporte, G. (eds.) Transportation, Handbooks in Operations Research and Management Science, vol. 14, pp. 367 – 428. Elsevier (2007)
5. Escudero, A., Muñuzuri, J., Guadix, J., Arango, C.: Dynamic approach to solve the daily drayage problem with transit time uncertainty. Computers in Industry 64(2), 165 – 175 (2013)
6. Francis, P., Zhang, G., Smilowitz, K.: Improved modeling and solution methods for the multi-resource routing problem. European Journal of Operational Research 180(3), 1045 – 1059 (2007)
7. Imai, A., Nishimura, E., Current, J.: A lagrangian relaxation-based heuristic for the vehicle routing with full container load. European Journal of Operational Research 176(1), 87 – 105 (2007)
8. Konings, J.: Intermodal barge transport: network design, nodes and competitiveness. PhD thesis, TU Delft, Delft University of Technology (2009)
9. del Mar Agamez-Arias, A., Moyano-Fuentes, J.: Intermodal transport in freight distribution: a literature review. Transport Reviews 0(0), 1–26 (2017)
10. Nossack, J., Pesch, E.: A truck scheduling problem arising in intermodal container transportation. European Journal of Operational Research 230(3), 666 – 680 (2013)
11. Riessen, B., Negenborn, R.R., Dekker, R.: Synchromodal container transportation: An overview of current topics and research opportunities. pp. 386–397. Lecture Notes in Computer Science 9335, Springer International Publishing (2015)
12. Smilowitz, K.: Multi-resource routing with flexible tasks: an application in drayage operations. IIE Transactions 38(7), 577–590 (2006)
13. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35(2), 254–265 (1987)
14. Wang, X., Regan, A.C.: Local truckload pickup and delivery with hard time window constraints. Transportation Research Part B: Methodological 36(2), 97 – 112 (2002)