Noname manuscript No.
(will be inserted by the editor)

Truck Scheduling in Tank Terminals

Thomas Van den Bossche · Hatice Çalık · Evert-Jan Jacobs · Túlio A. M. Toffolo · Greet Vanden Berghe

Received: date / Accepted: date

Abstract Tank terminals play a significant role in the storage and international transportation of goods such as liquids and gases. This paper focuses on a real-world problem where trucks must be assigned to loading stations which connect to tanks containing the product to (un)load. The typically limited operational space results in a specific type of blocking where trucks may be blocked not only at their assigned loading station, but also on their way towards it. The difficulty of fully understanding the repercussions of scheduling trucks to certain loading positions makes it extremely challenging for human operators to schedule trucks efficiently. The problem is contrasted against a range of active problems from the scheduling literature in order to establish its unique scientific merit. We provide a mixed integer programming formulation for this problem and a heuristic approach. The heuristic outperforms an estimation of the dispatching rules currently enforced by terminals, thereby providing such terminals with a practical tool for optimizing their operations. Moreover, in order to stimulate further research, a set of instances derived from real-world data has been made publicly available.

Keywords Tank terminals · Truck scheduling · Heuristics · Blocking constraints · Conflict-free scheduling

Thomas Van den Bossche
KU Leuven, Department of Computer Science, CODeS - Belgium
E-mail: thomas.vandenbossche@cs.kuleuven.be

Hatice Çalık
KU Leuven, Department of Computer Science, CODeS - Belgium
E-mail: hatice.calik@kuleuven.be

Evert-Jan Jacobs
KU Leuven, Department of Computer Science, CODeS - Belgium
E-mail: evertjan.jacobs@kuleuven.be

Túlio A. M. Toffolo
Federal University of Ouro Preto, Department of Computing - Brazil
E-mail: tulio@toffolo.com.br

Greet Vanden Berghe
KU Leuven, Department of Computer Science, CODeS - Belgium
E-mail: greet.vandenberghe@cs.kuleuven.be

1 Introduction

A significant number of liquid and gas products are transported internationally every day. Given that these products are typically moved over vast distances, it is not uncommon for their transportation to be divided between multiple modes of transport (such as ships, trucks and trains). As many liquid and gas products are potentially hazardous, they must be handled safely and as efficiently as possible. These safety requirements necessitate the use of intermediate storage facilities when transferring products between different modes of transport. This is accomplished via specialized tanks located in tank terminals.

Since the operational space of tank terminals is usually restricted in size, terminal operations must be well coordinated to guarantee efficient scheduling. The truck scheduling problem in tank terminals (TSTT) introduced in this paper involves a set of trucks, their expected arrival times, and the corresponding products to be (un)loaded. The tanks that store the products are connected to several loading stations which enable the transfer of products to or from the trucks. Each truck must be assigned to a single loading station in the terminal, chosen from a subset of stations which are compatible for the transfer of its specific product.

The service time of different stations may differ for the same truck as the flow rate is station-specific. Loading stations connected to the same tank cannot be operated simultaneously and the (un)loading operation of a truck cannot be interrupted.

Figure 1 presents a schematic overview of the terminal layout. Tanks are located in one area whereas trucks connect to loading stations in another. Loading stations connect to multiple tanks and all connections are via underground pipes.

Each tank is assumed to have infinite capacity, which means trucks can always (un)load their product completely. As a consequence, partial product transfers are not allowed. Trucks must (un)load at fixed loading stations grouped at a number of loading racks, which contain the equipment necessary for enabling the safe transfer of products. These racks are positioned along a small number of terminal aisles.

Trucks access their loading station by traversing one of the limited number of directed paths through the yard, which are defined as a sequence of narrow terminal aisles. Paths always begin at the terminal entrance and end at a common exit point. Some aisles can only be traversed in one direction and, due to the limited operational space, only one truck can traverse an aisle at a time and overtaking is not permitted. The terminal entrance is assumed to have infinite capacity, so there is no limit on the number of trucks that can be waiting simultaneously.

In addition to the actual product transfer, various tasks such as connection, verification and disconnection must be conducted by human operators to guarantee safety before initiating and after ending product transfer at loading stations. Each of these tasks requires a setup time which adds up to the truck's length of stay at their loading station. Since certain products are associated with very specific risks, operators assigned to a product transfer must have the necessary skills to conduct the task. When all suitable operators are occupied, a truck must wait for a skilled operator to become available.

Terminal layout restrictions combined with the frequency of truck arrivals introduce some unique blocking constraints regarding a truck's capability of reaching and exiting its assigned loading station in a timely fashion. Blocking may occur in one of the following situations:

1. The assigned loading station is occupied by another truck.
2. The tank to (un)load from, which may be connected to different loading stations in the yard, is serving another truck.
3. The path to reach the assigned loading station (before initiating the product transfer) is occupied by one or more trucks.
4. The path to reach the exit point (after the transfer is completed) is occupied by one or more trucks.

A truck may have to wait at two different locations at the terminal: (i) the entrance and (ii) the loading station it is assigned to. The truck waits at the entrance until its loading station becomes free and reachable and there is an operator available to connect the truck to the loading station. Once this is the case, the truck drives and connects to its assigned loading station. It then remains at the loading station until the product transfer has ended and the path to the exit is clear. Once the truck reaches the exit, it leaves the system immediately.

Since trucks may be delayed due to both blocking and the absence of skilled operators, truck scheduling and operator assignment are highly interrelated and have a significant impact on modelling decisions.

The objective of the TSTT is to minimize the trucks' total tardiness, which will be formally defined in Section 3. Blocking is the primary bottleneck associated with the TSTT and therefore it is crucial that blocking restrictions are addressed by the proposed scheduling approach.

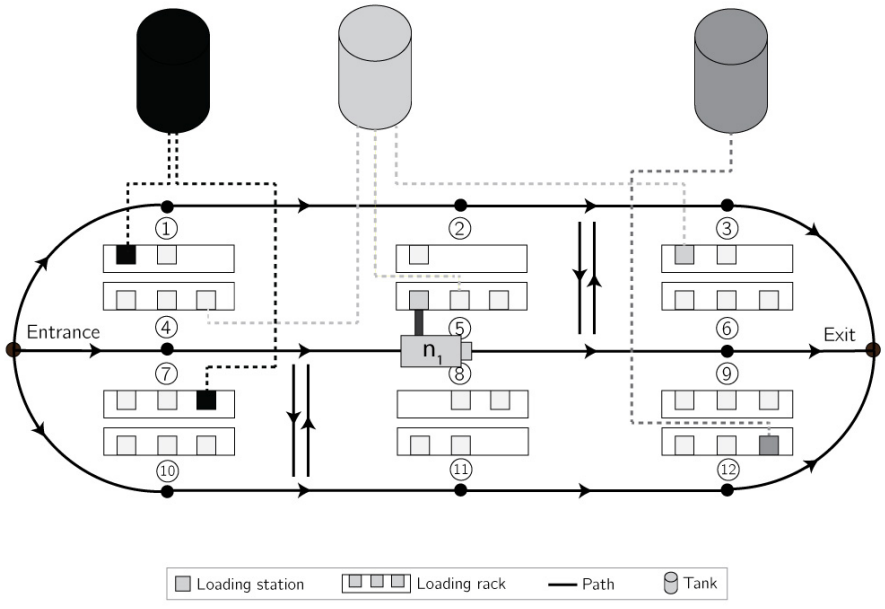

Fig. 1 Sample terminal layout

The remainder of the paper is structured as follows. Related real-world problems where similar types of blocking occur are discussed throughout Section 2. Section 3 describes the TSTT as well as the blocking constraints which arise due to the spatially-restricted operational area. Section 4 introduces a mixed integer programming formulation for the TSTT. Section 5 presents a constructive heuristic and a refinement procedure which obtains good quality results within reasonable computation time. Section 6 provides the results obtained from experiments conducted on instances derived from data provided by a real-world tank terminal. Finally, conclusions are outlined in Section 7 in addition to discussing possible future research directions.

2 Related work

The compact layout of tank terminals results in a unique type of blocking. Nevertheless, other real-world problems can be identified in which specific actions or operations must be scheduled and some kind of blocking at path segments occurs. Most similar types of blocking are to be found in problems concerning the scheduling of vehicles to complete actions or retrieve goods, such as aircraft scheduling, train scheduling, and order picking at warehouses. Certain distinctions can be made regarding various characteristics such as the operation to be scheduled and the position in the network where blocking may arise. Since the type of blocking introduced in the present paper bears similarities with those in academic work, a thorough assessment of these previous problems will help isolate and identify the precise novelty of blocking in the TSTT.

2.1 Aircraft scheduling

The aircraft scheduling problem in a terminal maneuvering area concerns making conflict-free decisions with respect to take-off and landing operations where the most common objectives are minimizing delay propagation or deviation from the target landing and take-off times. A thorough classification of existing models has been offered by [18]. An airport network consists of air segments, runways and holdings. For aircraft to take-off and land, they must traverse these segments while respecting the given airport schedule as much as possible. Each aircraft is associated with a sequence of operations to be executed, where an operation is defined as traversing one of the airport's segments with a fixed speed. When traversing these segments, aircraft cannot overtake as this would result in safety risks. Blocking occurs whenever aircraft traversing the same segment do not respect a minimum safety distance between one another and is dependent upon both the aircraft sequence and their respective paths. As opposed to tank terminals, where a truck may be assigned to multiple loading stations which are accessible by multiple paths, aircraft are less flexible due to the limited number of segments for their routing. Additionally, the sequence of operations attributed to aircraft does not require additional operators and therefore no delays are caused by operator unavailability.

Generally, aircraft routes are assumed to be fixed, with their re-routing treated as a separate problem. Samà et al. [19] proposed a framework which optimizes

the two subproblems in a lexicographic manner. However, since routing (path assignment) and scheduling decisions (loading stations and operators) are highly interrelated in the present work, an integrated approach is required.

2.2 Railway scheduling

The single-track railway scheduling problem occurs when a set of trains must travel through a predefined railway network which consists of single tracks, sidings and stations. The objective is to minimize the total tardiness of trains, while respecting the time schedule initially provided by the railway company as much as possible. This problem can be modelled as a job-shop scheduling problem, by considering the train trips as jobs, which will be scheduled on tracks regarded as resources.

In the literature, the problem is modelled as a job-shop scheduling problem with blocking constraints. These constraints refer to situations in which trains occupy a track section longer than necessary until the succeeding section is available for traversal. Zhou and Zhong [21], Meng and Zhou [16], Törnquist and Persson [20] induce blocking indirectly by setting the finishing time of a train on a preceding track section equal to the starting time on the succeeding track. By contrast, Lange and Werner [11] only apply starting time variables and include blocking directly in additional constraints. Sections with parallel tracks are interpreted as parallel machines, one machine with parallel machine units, or intermediate buffers with a capacity equal to the number of parallel tracks.

Generally, work from the literature assumes that all trains take the same amount of time to traverse track segments. The waiting time during which they are stationary at platforms or sidings is assumed to be far less than the time required for them to travel the railway network. This is in stark contrast to the TSTT, where transferring a product at a loading station is assumed to be the most time-consuming activity. Additionally, routing is generally not part of the decision-making process as a fixed route is provided for each train.

2.3 Order picking

Similarities with respect to the present work can also be found in the literature on warehouse operations, a comprehensive review of which has been offered by Karasek [10]. Among these operations, order picking is the most relevant with respect to the TSTT. In warehouses, order pickers navigate through narrow aisles to retrieve items from shelves, placing them in a cart. The most common objective in order picking is to minimize total retrieval time. Each item in an order is stored at only one pick-up location and an automatic picker always returns to its original position whenever a trip is complete. Picker blocking occurs whenever a picker enters an aisle in which another picker is already present. Similar to the truck scheduling problem, pickers are physically incapable of overtaking due to the narrow warehouse aisles. Each item is only available at a single location whereas in the TSTT a product may be provided at multiple loading stations, which are themselves accessible by multiple paths. Furthermore, traveling to and from the picking location takes far longer than retrieving the item.

2.4 Collision-free vehicle routing

Blocking constraints are generally referred to as *conflicts* when considering collision-free vehicle routing problems. The most common applications wherein such blocking constraints occur are robot movement and container transportation by automated guided vehicles.

One approach consists of a dynamic routing algorithm [9] which computes collision-free routes by considering time-expanded networks wherein it is permitted to wait at vehicle start positions as well as on the edges which constitute paths. When delays or disturbances occur, routes must be recomputed. More recently, Gawrilow et al. [8] approached the problem by way of static route computations. In order to avoid collisions, a vehicle is assigned to the subsequent part of its route while other vehicles are not allowed to traverse it.

Whereas Gawrilow et al. [9, 8] assume that vehicles have a predefined start and end point in the network, a truck's destination is not fixed in the TSTT since it depends on the assigned loading station. Furthermore, Gawrilow et al. [9, 8] assume that vehicles may only be blocked at their paths. By contrast, in the TSTT blocking may occur at both paths and loading stations, and depends on the availability of resources, namely the assigned tank and skilled operators. Therefore, truck routing, scheduling, and operator assignments should be integrated.

The real-world problems outlined in the preceding pages are all closely related to the present work insofar as they all concern the routing of vehicles in environments where blocking occurs. Certain fundamental differences between the TSTT and those discussed in relation to blocking can be summarized as follows:

1. The travel time in the discussed problems is, proportionally, far longer than the time required to conduct the primary activity or operation. Trains spend more time traveling between stations than they spend stationary and aircraft taxiing takes far longer than take-off or landing, for example.
2. The physical environments in the aforementioned problems are less interconnected since routes are generally assumed to be fixed. Consequently, there are fewer options as regards how to route vehicles toward their destinations. However, in the TSTT scheduling and routing decisions are highly interrelated.
3. In the TSTT blocking at multiple resources is considered, namely at loading stations and paths.

In addition to the assignment of paths and loading stations to the trucks, the TSTT requires the assignment of operators to each truck's task. This operator assignment procedure requires solving an integrated staff scheduling problem which is interrelated with the scheduling of the trucks. Recently, a classification of staff scheduling problems which include task assignment has been offered by [7]. Since there is a limited number of skilled operators available, the operator assignment scheme may also become a blocking factor and therefore influences the truck schedules.

3 Problem description

The problem under consideration consists of a tank terminal with a set of tanks H; a set of loading stations S; a set of trucks N to be served; and a set of operators $O = O^C \cup O^V \cup O^D$ where O^C, O^V, and O^D are the - not necessarily disjoint - sets of operators who perform connection, verification, and disconnection tasks, respectively. Loading stations are grouped together into a set of racks R and each loading station $s \in S$ is connected to a tank $h(s) \in H$. Let $h_n \in H$ be the tank associated with the product of truck $n \in N$ and $S_n \subset S$ denote the set of loading stations compatible with the transfer of the product of $n \in N$, such that $h(s) = h_n, \forall s \in S_n$. Truck n must be assigned to one of the stations in S_n. The notation regarding tanks can therefore be omitted from the remainder of this study. An operator must be assigned to each of the connection, verification, and disconnection tasks associated with each truck. In order to ensure a safe product transfer and keep human error at a minimum during these operations, the proper connection of a truck to a loading station must be verified by another operator. Therefore, the operator who conducted a connection is not allowed to verify this task themselves. Once the connection and verification tasks have ended, the transfer can be performed without the need of an operator. However, an operator will once again be required for disconnection after the transfer is complete.

The service time for the connection, verification, and disconnection tasks, which are fixed and identical for each $n \in N$, are denoted as T^C, T^V and T^D, respectively. The transfer time needed for n's product at station $s \in S$ is defined as T^P_{ns}. The arrival time at the yard entrance is also fixed for each n and is denoted by T^A_n.

The most time-consuming component in the TSTT is transferring the product rather than traversing the path to and from this product's location. In the real-world case (un)loading may take several hours, while the time spent traveling through the yard is at most a few minutes. Given that the travel time is negligible compared to the transfer time, it is ignored for the purpose of the present paper.

Trucks may need to wait either at the entrance or at the assigned loading station for one of the following reasons:

1. A truck must wait at the entrance
   (a) if the assigned loading station is occupied.
   (b) if a station neighboring the assigned loading station is occupied.
   (c) if the path to the assigned loading station is blocked.
   (d) if the tank of the required product is busy with the transfer of another truck.
   (e) for the operator to perform connection.
2. A truck must wait at the loading station
   (a) for the operator to perform verification.
   (b) for the operator to perform disconnection.
   (c) if the path to the exit is blocked.

Let us denote the departure time of truck $n \in N$ as t^D_n and the loading station it is assigned to as s_n. Then, the total time truck n spends in the terminal before its departure, say w_n, can be expressed as follows:

$$w_n = t^D_n - T^A_n \tag{1}$$

Note that the values for s_n and t_n^D are not given a priori and are decisions to be taken. A truck can only be assigned from its arrival time onwards.

Due to established agreements with contractors, trucks are generally allocated a predefined maximum duration T_{max} wherein they should be served. Exceeding this duration will result in significant additional costs for the terminal because they must suffer all overtime expenses. Let $(w_n - T_{max})^+ = \max\{0, w_n - T_{max}\}$ denote the tardiness of truck $n \in N$. The objective is to minimize the total tardiness over all trucks, which we specify as follows:

$$\min \sum_{n \in N} (w_n - T_{max})^+ \tag{2}$$

Before formally describing the problem, Section 3.1 explains all possible blocking situations in detail, along with examples.

3.1 Blocking constraints

Each truck must be assigned to a feasible loading station s_n, a terminal path to access this station and operators to perform its tasks. The choice of loading station and path may significantly impact the scheduling process since blocking may occur with respect to either. Therefore, routing and scheduling decisions are highly interrelated.

3.1.1 Types of blocking

Given that yard occupancy (defined as the number of truck arrivals per time unit) is higher at busier periods, multiple trucks may have to traverse the same path simultaneously. Trucks may consequently be blocked from reaching their assigned loading station, resulting in significant delays.

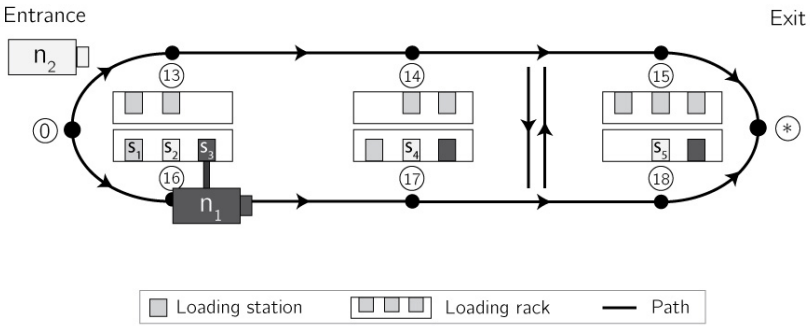

Fig. 2 Direct and indirect blocking.

In order to visually describe blocking, a simplified yard layout is provided in Figure 2 with six loading racks, each containing multiple loading stations. The

terminal entrance and exit are denoted as nodes 0 and $*$, respectively. We then have four terminal paths from the entrance to the exit: $0 \to 13 \to 14 \to 15 \to *$, $0 \to 13 \to 14 \to 18 \to *$, $0 \to 16 \to 17 \to 18 \to *$ and $0 \to 16 \to 17 \to 15 \to *$.

Next, assume truck n_1 (the dark gray truck in Figure 2) is assigned to loading station s_3 belonging to rack 16. There is only one path which enables n_1 to reach rack 16 ($0 \to 16$). While n_1 is located in the loading yard, truck n_2, which is denoted by the light gray truck, is waiting at the terminal entrance to (un)load its product at one of its potential loading stations (s_2, s_4 and s_5). In this example, these potential loading stations are considered to be located at racks 16, 17 and 18. In Table 1, we provide a list of paths to access these loading racks.

Rack	**Accessible paths**
16	$0 \to 16$
17	$0 \to 16 \to 17$
18	$0 \to 13 \to 14 \to 18$ $0 \to 16 \to 17 \to 18$

Table 1 Possible paths for reaching racks 16, 17 and 18 from the entrance.

Let us address the possibility of n_2 (un)loading at each of these racks:

- **Rack 16.**
  Blocking occurs if both trucks n_1 and n_2 would be assigned to the same loading station or neighboring loading stations of the same rack simultaneously due to the trucks' length. This type of blocking is referred to as `direct blocking` since the required resource itself is unavailable. On the sample layout of Figure 2, once truck n_1 is connected to station s_3 of rack 16, then, although station s_2 of the same rack is reachable from the entrance, truck n_2 cannot be connected to it because it is blocked by truck n_1. The assignment of truck n_2 to rack 16 would be possible only by scheduling it after n_1's departure.
- **Rack 17.**
  When assigning n_2 to a station belonging to rack 17, no physical blocking occurs given that this rack is clearly vacant. However, there is only one path by which rack 17 may be accessed ($0 \to 16 \to 17$), part of which is currently occupied by truck n_1. Since overtaking is not permitted, n_2 cannot access its loading station at the desired time. This type of blocking is referred to as `indirect blocking`, since the assigned resource is available, but the path towards it is not.
- **Rack 18.**
  This rack is accessible by the same path employed by truck n_1: $0 \to 16 \to 17 \to 18$, which would imply indirect blocking may occur. However, since another path to this rack is possible ($0 \to 13 \to 14 \to 18$) which is currently not occupied by any truck, `no blocking` occurs and therefore truck n_2 can immediately proceed to its loading station at this rack.

Indirect blocking scenarios When indirect blocking occurs, the resulting delay depends on the position of the blocked truck with respect to the blocking one. Consider now the different layout provided in Figure 3 and a schedule that assigns

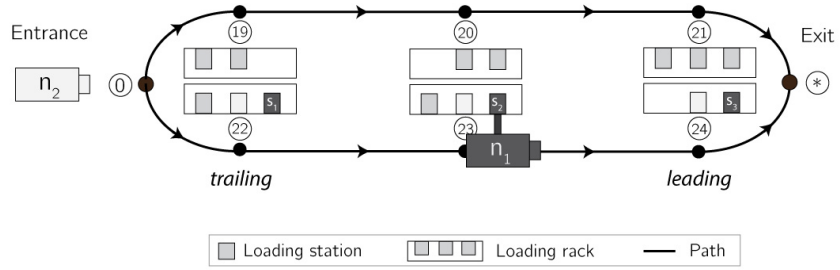

Fig. 3 Trailing and leading trucks.

truck n_1 to loading station s_2. Another truck, n_2, is waiting at the terminal entrance and can only be assigned to either s_1 or s_3.

Depending on which loading station truck n_2 is assigned to, it can either be referred to as *trailing* or *leading*. In the former, truck n_2 is assigned to s_1 which is preceding truck n_1's loading station on the path, while in the latter n_2 is assigned to s_3 which succeeds truck n_1 on the path. This spatial aspect has implications concerning the nature of the delay incurred by blocking.

When truck n_2 is *trailing*, it may already proceed to its loading station and begin transferring the product, assuming skilled operators are available. However, its departure time is delayed until n_1 has departed:.

$$t^D_{n_2} > t^D_{n_1}$$

By contrast, when truck n_2 is *leading*, it cannot proceed to its loading station because the path toward it is clearly blocked. Therefore, its release time $t^R_{n_2}$ (the earliest time when a truck's first task can be executed) is delayed until n_1 has departed:

$$t^R_{n_2} > t^D_{n_1}$$

Another type of indirect blocking occurs when the tank of the product required by one truck is already being used by another. Consider again the layout in Figure 3. Assume trucks n_1 and n_2 both require access to the same tank serving loading stations s_1, s_2 and s_3. Imagine truck n_1 is already assigned to loading station s_2. Although there are no other trucks in the system yet, truck n_2 cannot proceed to loading stations s_1 or s_3 before the operation of truck n_1 is finished.

4 Problem formulation

We provide a mixed integer programming (MIP) formulation of the TSTT based on a complete enumeration of feasible allocations. An allocation assigns all trucks to paths and loading stations. In addition, it assigns operators to perform the connection, verification and disconnection tasks associated with each truck. An allocation is considered feasible if the path assigned to each truck is a path existing in the terminal layout and the assigned loading station is on this path. Additionally,

operator assignments must respect the skill requirements and the safety condition that connection and verification cannot be performed by the same operator for the same truck. We define $M = \{m_1, \ldots, m_{|M|}\}$ as the set of all such feasible allocations.

Consider the sample problem instance wherein two trucks must be scheduled in a terminal whose layout is represented in Figure 4. The terminal has four loading stations and there are two identical operators, each capable of performing all operation types. Truck 1 can be served at s_1 or s_2 whereas truck 2 can be served at s_3 or s_4.

One of the feasible allocations for this problem instance assigns truck 1 to s_1, truck 2 to s_4, both trucks to path $0 \rightarrow s_1 \rightarrow s_4 \rightarrow *$, and the connection, verification and disconnection of both trucks to operators 1, 2 and 1, respectively.

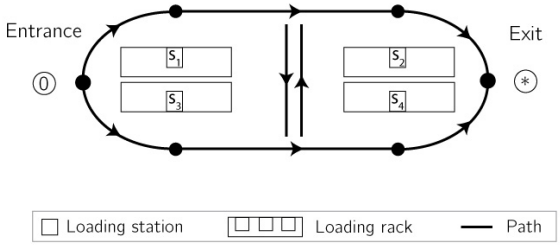

Fig. 4 Sample layout with four loading stations.

Given the path, loading station and operator assignments for all trucks, one can model the problem as a scheduling problem with blocking constraints. To do so, we construct a disjunctive graph $G^m = (V^m, Z^m, W^m)$ for each allocation $m \in M$ as described in Section 4.1.

4.1 Disjunctive graphs for allocations in M

Roy and Sussmann [17] were the first to formulate scheduling problems with disjunctive constraints, a method which has since become common in the scheduling literature. Balas [2] provides an implicit enumeration algorithm to minimize the makespan of job-shop scheduling problems via disjunctive graphs. Adams et al. [1] introduce a shifting bottleneck procedure utilizing disjunctive graph representations to solve the same problem. Early efficient branch-and-bound algorithms were developed by Carlier and Pinson [6] and Brucker et al. [3]. Mason et al. [15] introduce a modified shifting bottleneck algorithm to minimize the total tardiness in job-shop scheduling problems. This algorithm is later compared with an MIP based heuristic by Mason et al. [14] and although the MIP based heuristic performs better on small size instances, the algorithm of [15] proves to be more efficient on larger ones.

An arbitrary allocation $m \in M$ includes complete information concerning the loading station, path and operators assigned to every truck. Given this information, the problem reduces to scheduling jobs of all trucks where each job has five tasks

to go through (connection, verification, transfer, disconnection, departure). Some of these tasks may need to be processed on the same machine (meaning the same operator or path) and since no buffer is allowed, some tasks might continue to block the machine even after their actual processing is complete.

Mascis and Pacciarelli [13] are the first to formulate the job-shop scheduling problem with blocking constraints by generalizing disjunctive graphs to what they term alternative graphs. In [13], blocking constraints represent the situations where there is no space for storage between machines. Later, Liu and Kozan [12] utilize alternative graphs to formulate the train scheduling problem as a blocking parallel-machine job-shop scheduling problem. Additionally, Burdett and Kozan [4] use alternative graphs to represent blocking constraints in the scheduling of health care activities in hospitals.

The disjunctive graph $G^m = (V^m, Z^m, W^m)$ of an arbitrary $m \in M$ is composed of a set V^m of vertices, a set Z^m of conjunctive arcs, and a set W^m of disjunctive arcs. We construct V^m, Z^m and W^m as follows.

Define $J^0 = \{Con, Ver, Trans, Dis, Dep\}$ as the set of tasks and $J = \{1, \ldots, 5\}$ as the corresponding index set. For each task $j \in J$, define $V_{1j}^m = \bigcup_{n \in N} < n, j >$ where $< n, j >$ is a node corresponding to task j of truck n. Then, introduce a dummy node for each node in V_{1j}^m and denote the set of dummy nodes as V_{2j}^m for $j \in J$. Finally, set $V^m = \{0\} \cup \{*\} \cup V_1^m \cup V_2^m$ where $V_1^m = \bigcup_{j \in J} V_{1j}^m$ and $V_2^m = \bigcup_{j \in J} V_{2j}^m$.

The set of conjunctive arcs Z^m is constructed as follows:

$$\begin{aligned} Z^m = &\{(0, i) : i \in V_{11}^m\} \\ &\cup \{(i, k) : i \in V_{1j}^m, k \in V_{2j}^m, j \in J\} \\ &\cup \{(i, k) : i \in V_{2j}^m, k \in V_{1(j+1)}^m, j = 1, \ldots, 4\} \\ &\cup \{(i, *) : i \in V_{25}^m\} \end{aligned}$$

Figure 5 provides the disjunctive graph for another allocation $m \in M$ of the sample problem associated with the layout depicted in Figure 4. In this particular allocation, truck 1 is assigned to loading station s_1 and truck 2 is assigned to loading station s_4. Both trucks are assigned to path $0 \rightarrow s_1 \rightarrow s_4 \rightarrow *$. The operator indices assigned to the connection, verification, and disconnection for truck 1 are [1,2,1] and they are [2,1,2] for truck 2.

In Figure 5, there are two nodes associated with each truck-task pair. From left to right, the first one of the two denotes the start of the corresponding task and the second one denotes the end. In this figure, the nodes also indicate operator assignments to the task when applicable. For example, the node labeled $1, O^C = 1$ denotes the start of connection for truck 1 done by Operator 1 whereas $1, O^C = 1'$ represents the end of this task. In this graph, $|V_1^m| = |V_2^m| = 2 \times 5 = 10$ and $|V^m| = 22$ (as a reminder, $V^m = \{0\} \cup \{*\} \cup V_1^m \cup V_2^m$).

The solid arcs in Figure 5 are the conjunctive arcs. Recall that T^C, T^V, T_{ns}^P and T^D are the service times for the connection, verification, transfer and disconnection tasks, respectively. This figure distinguishes three groups of disjunctive arc pairs using three types of dashed lines.

The first group has one pair of disjunctive arcs: $(< 1, O^C = 1' >, < 2, O^C = 2 >)$ and $(< 2, Dep' >, < 1, Dep >)$. These two arcs define a disjunctive pair which implies that if the connection of truck 1 precedes the connection of truck 2 then the departure of truck 2 cannot precede the departure of truck 1. This is because

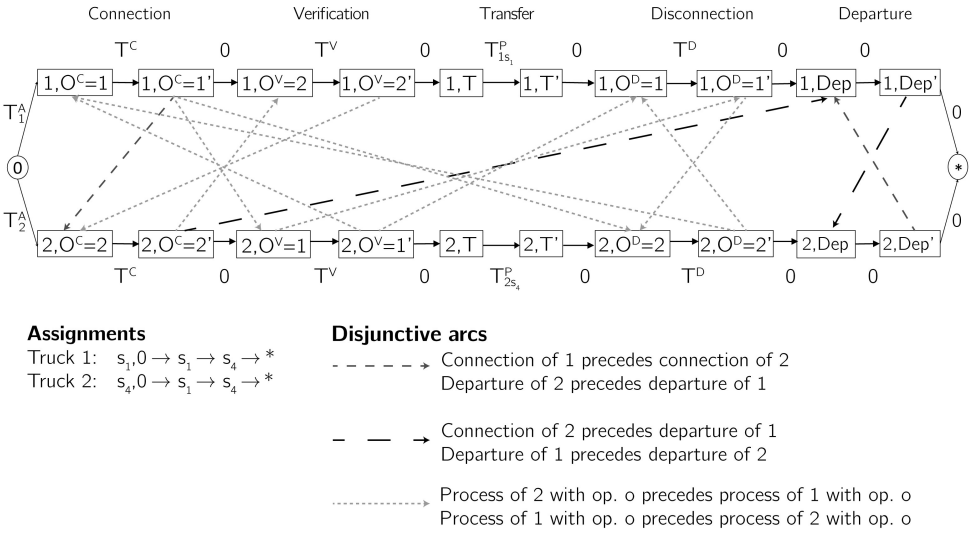

Fig. 5 Disjunctive graph for 2 trucks, 2 operators, 4 loading stations. Truck 1 is allocated to loading station s_1, truck 2 is allocated to loading station s_4, and both trucks are allocated to path $0 \to s_1 \to s_4 \to *$.

the loading station of truck 1 (s_1) is located on the path of truck 2 and therefore truck 2 cannot reach its loading station when truck 1 is at s_1.

The second group also has a single pair of disjunctive arcs: $(< 2, O^C = 2' >, < 1, Dep >)$ and $(< 1, Dep' >, < 2, Dep >)$. This pair implies that if the connection of truck 2 precedes the departure of truck 1 then the departure of truck 1 cannot precede the departure of truck 2. In this case, the loading station of truck 2 (s_2) is located on the path of truck 1 after its loading station. Therefore, truck 1 cannot depart when truck 2 is at s_2.

The third group of disjunctive arcs represents the precedence relation between the tasks assigned to the same operators. If two different tasks are assigned to the same operator, one of them should finish before the other starts.

In this particular example, the trucks are assigned to different loading stations which are also on different racks. Fortunately, blocking due to assignment of trucks to the same or neighboring loading stations at the same rack can also be represented by disjunctive arcs. Assume an additional truck, say truck 3, is assigned to the same loading station as truck 1. One can then represent the blocking relation by the disjunctive arc pair $(< 3, Dep' >, < 1, O^C = o_1 >) \leftrightarrow (< 1, Dep' >, < 3, O^C = o_3 >)$ where o_1 and o_3 are the indices of the operators assigned to the connection of truck 1 and truck 3, respectively. The blocking of neighboring loading stations at the same racks can also be handled in a similar fashion.

We generate all pairs of disjunctive arcs by using the three aforementioned groups of blocking scenarios. We then define P^m as the set of disjunctive arc pairs where each arc pair, say $(i, j) \leftrightarrow (k, l) \in P^m$, is associated with two disjunctive arcs $(i, j), (k, l) \in W^m$.

Let T_{ij}^m be the weight of arc $(i,j) \in Z^m \cup W^m$, $j(i) \in J$ be the task index, and $n(i)$ be the truck index of node $i \in V_1^m \cup V_2^m$. Then,

$T_{0i}^m = T_{n(i)}^A, i \in V_{11}^m$,
$T_{ik}^m = T^C, (i,k) : i \in V_{11}^m, k \in V_{21}^m$,
$T_{ik}^m = T^V, (i,k) : i \in V_{12}^m, k \in V_{22}^m$,
$T_{ik}^m = T_{n(i)s_{n(i)}}^P, (i,k) : i \in V_{13}^m, k \in V_{23}^m$,
$T_{ik}^m = T^D, (i,k) : i \in V_{14}^m, k \in V_{24}^m$, and
$T_{ik}^m = 0$ for all other arcs in $Z^m \cup W^m$.

4.2 A mixed integer programming formulation

After constructing the disjunctive graphs G^m for all $m \in M$, we define the following sets of decision variables:

$u^m = 1$ if allocation $m \in M$ is selected, 0 otherwise.

$x_{ij}^m = 1$ if task of node i precedes task of node j whereas $x_{ij}^m = 0$ if task of node k precedes task of node l for disjunctive arc pair $(i,j) \leftrightarrow (k,l) \in P^m$ for $m \in M$.

t_i^m is the time label of node $i \in V_1^m \cup V_2^m$ of $m \in M$.

ω_i^m is the tardiness of $n(i)$ for node $i \in V_{25}^m$ of $m \in M$.

Note that t_i^m corresponds to the starting time of task $j(i) \in J$ for truck $n(i) \in N$ when $i \in V_1^m$ and to the ending time when $i \in V_2^m$. Moreover, $t_i^m = t_j^m$ for $i \in V_{15}^m, j \in V_{25}^m : n(i) = n(j)$.

Given that L is a sufficiently large positive real valued parameter, the following is an MIP formulation for the TSTT:

$$\min \sum_{m \in M} \sum_{i \in V_{25}^m} \omega_i^m \tag{3}$$

$$\text{s.t.} \sum_{m \in M} u^m = 1, \tag{4}$$

$$\omega_i^m \geq t_i^m - T_{n(i)}^A - T_{max} \qquad \forall i \in V_{25}^m, m \in M \tag{5}$$

$$t_i^m + T_{ij}^m u^m \leq t_j^m, \qquad \forall (i,j) \in Z^m, m \in M \tag{6}$$

$$t_i^m \leq t_j^m + L(u^m - x_{ij}^m), \qquad \forall (i,j) \leftrightarrow (k,l) \in P^m, m \in M \tag{7}$$

$$t_j^m \leq t_i^m + L x_{ij}^m, \qquad \forall (i,j) \leftrightarrow (k,l) \in P^m, m \in M \tag{8}$$

$$t_l^m \leq t_k^m + L(u^m - x_{ij}^m), \qquad \forall (i,j) \leftrightarrow (k,l) \in P^m, m \in M \tag{9}$$

$$t_k^m \leq t_l^m + L x_{ij}^m, \qquad \forall (i,j) \leftrightarrow (k,l) \in P^m, m \in M \tag{10}$$

$$\omega_i^m \geq 0, \qquad \forall i \in V_{25}^m, m \in M \tag{11}$$

$$u^m \in \{0,1\}, \qquad \forall m \in M \tag{12}$$

$$x_{ij}^m \in \{0,1\}, \qquad \forall (i,j) \leftrightarrow (k,l) \in P^m, m \in M. \tag{13}$$

Objective function (3) minimizes the total tardiness of trucks. Constraints (5) and (11) ensure that the tardiness of any truck n is no less than $max\{0, t_n^D - T_n^A -$

$T_{max}\}$ for the selected allocation. Constraint (4) selects exactly one of the feasible allocations enumerated a priori. For each conjunctive arc of the selected allocation, Constraints (6) ensure that the time label of a node is no less than the time label of its predecessor plus the weight of the arc (processing time). Constraints (7)-(10) ensure that for each disjunctive arc pair of an allocation, exactly one of the disjunctive arcs is active. In other words, among the two precedence relations, exactly one is chosen. Finally, (12) and (13) are binary restrictions.

4.3 Symmetry elimination and evaluation of the MIP formulation

When the operators are identical, we face symmetry regarding the choice of operators. Consider again the problem discussed in Section 4.1 for the sample layout visualized in Figure 4. Let [1,2,1] and [1,2,1] be the indices of the operators assigned to the connection, verification and disconnection operations of truck 1 and truck 2, respectively. Assume truck 1 is assigned to loading station 1, truck 2 is assigned to loading station 4, and that both trucks are assigned to path $0 \rightarrow 1 \rightarrow 4 \rightarrow *$. An alternative solution leading to the same objective value could be obtained by keeping the loading station and path assignments as they are and changing the operator assignments to [2,1,2] and [2,1,2].

For the problem in Figure 4, 16 feasible station-path-operator assignments are possible for each truck. These assignments illustrated by Figures 6 and 7 in Appendix A lead to $|M| = 16 \times 16 = 256$ feasible allocations. The following paragraph explains one way of symmetry elimination.

Let $\overline{o} = \{\overline{o}_1, \overline{o}_2, \overline{o}_3, \ldots\ \overline{o}_{3(|N|-1)+1}, \overline{o}_{3(|N|-1)+2}, \overline{o}_{3|N|}\}$ be a generic operator assignment, in which $\overline{o}_{3(n-1)+1}$, $\overline{o}_{3(n-1)+2}$, and $\overline{o}_{3n}$ correspond to the operator assignments of connection, verification and disconnection, respectively, for truck n. Without loss of generality, assume $\overline{o}_i$ and $\overline{o}_j$ are two distinct operators for some $i, j \leq 3|N|$. Now consider a different operator assignment $\overline{o}'$ such that $\overline{o}'_k = \overline{o}_j$ for every $k \leq 3|N| : \overline{o}_k = \overline{o}_i$ and $\overline{o}'_l = \overline{o}_i$ for every $l \leq 3|N| : \overline{o}_l = \overline{o}_j$. Then, $\overline{o}$ and $\overline{o}'$ are symmetric operator assignments and one can safely exclude the allocations with operator assignments of type $\overline{o}'$ from M.

By using this symmetry elimination, we need only construct half the disjunctive graphs (128). We conduct experiments on the MIP formulation using Java and CPLEX 12.8 with and without the symmetry elimination. The instances employed are based on an artificial terminal layout with four loading stations and two identical operators. The number of trucks ($|N|$) ranges from 2 to 4. Instances of the same $|N|$ value differ in configurations for the set of potential loading stations and/or the set of feasible paths for each truck.

Table 2 presents the results of the experiments on the MIP formulation. The first column gives the number of trucks, while the second and the third columns show the number of potential stations and the number of potential paths for each truck, respectively. Column 'Opt' indicates the total tardiness (in minutes) of the corresponding problem solution. Columns 5-7 present the results for the experiments, in which the symmetric assignments of the operators are kept whereas the last three columns present the results when those symmetric solutions are eliminated from the solution space through the aforementioned procedure. Columns '$|M|$', 'Time', and 'B&B nodes' represent the number of allocations generated, the time spent, and the number of branch-and-bound nodes visited by the model to

solve the problem. We introduce a time limit of one hour and a memory limit of 40 GB. We use the notation 'NA' if the model cannot provide the corresponding information due to the memory or time limit.

	Number of potential			Symmetry			Symmetry eliminated		
$\|N\|$	Stations	Paths	Opt	$\|M\|$	Time (sec)	B&B nodes	$\|M\|$	Time (sec)	B&B nodes
2	2,2	2,3	7	128	0.77	0	64	0.40	0
2	2,2	3,3	5	256	2.69	0	128	0.67	0
3	1,2,1	1,3,1	25	256	28.27	0	128	6.32	0
3	2,2,2	3,3,3	25	4096	1030.02	1775	2048	454.46	786
4	1,1,1,1	2,2,2,2	47	2048	1440.15	2054	1024	671.48	1823
4	2,2,2,2	3,3,3,3	NA	65536	3600.00	NA	32768	3600.00	NA

Table 2 Evaluation of the MIP formulation on small problem instances with two identical operators.

An immediate observation from Table 2 is that symmetry elimination halves the number of allocations that need to be considered in the model. This reduces the size of the formulation by halving the number of binary decision variables and leads to a considerable decrease in the solving time.

Table 2 reveals that the MIP formulation already has difficulty in solving problems with four trucks in a relatively simple terminal layout. The model is unable to find any integer feasible solution within the time limit for the problem with four trucks where each has three potential paths. Therefore, we develop a dedicated heuristic approach in Section 5 to handle problems of realistic size.

5 Heuristic approach

The developed heuristic method begins with the construction of an initial feasible solution, which is then improved by a refinement procedure. The following sections explain all the necessary details for implementing the constructive heuristic and the refinement procedure.

5.1 Constructive dispatching heuristic

The initial solution is obtained via a constructive heuristic which mimics the rule of thumb dispatching strategy currently employed by the tank terminal. Manual decision makers assign trucks to available loading stations which are located nearest to the yard exit.

The construction of the initial solution consists of two phases: an *Assignment Phase* followed by a *Scheduling Phase*. The *Assignment Phase* assigns each truck to an eligible loading station and path, while also determining the order in which trucks will be served in the yard. Based on the information obtained from this first phase, the *Scheduling Phase* then generates a feasible schedule that assigns operators to each truck task while taking potential blocking into account.

5.1.1 Assignment Phase

The *Assignment Phase* is detailed in Algorithm 1. First, all trucks are sorted in non-decreasing order of arrival times T_n^A (Line 1). Next, for each truck $n \in N$, the loading station which is closest to the yard exit - and not yet assigned to another truck - is selected (Lines 4 and 5). Should all loading stations already be occupied, the station with the required product which is nearest to the exit is chosen (despite its current unavailability). As a consequence, the corresponding truck must wait at the terminal entrance until the loading station becomes available. Next, a path is randomly selected from all of those which start at the entrance and lead to the selected loading station (Line 6). These decisions at truck level - loading station and path - are associated with each of the separate truck tasks. We refer to each of these associations as an assignment (*makeAssignment* at Line 8). Once this procedure has been performed for all trucks, the *Assignment Phase* ends by returning assignment list A (Line 12).

Algorithm 1: *Assignment Phase*

```
1  N = {n_1, ..., n_|N|} ← ordered list of trucks (non-decreasing arrival times);
2  A ← ∅;                                          ▷ initialize list of assignments
3  for n ← 1 to |N| do
4  |   S_n ← set of eligible, unassigned loading stations for n;
5  |   s_n ← closest loading station s ∈ S_n to the yard exit;
6  |   p_n ← random path which provides access to s_n;
7  |   for task j ← 1 to 4 do ▷ connection, verification, transfer, disconnection
8  |   |   a_nj ← makeAssignment(j, s_n, p_n);
9  |   |   A ← A ∪ {a_nj};
10 |   end
11 end
12 return A;
```

5.1.2 Scheduling Phase

Given list A obtained in the first phase, a feasible schedule is generated by assigning human operators to all tasks by taking into account direct and indirect blocking. This *Scheduling Phase* is outlined in Algorithm 2 and works as follows.

Starting from an empty schedule P, list A is iterated over sequentially (Line 3). For each assignment $a \in A$ where its associated task type is *Connection*, it is checked whether or not blocking with already-scheduled trucks is present in P. When no blocking occurs, an operator is immediately assigned to perform task j of a. Otherwise, the corresponding blocking type (direct or indirect) is returned for each conflicting truck (Lines 11 and 12). Depending on the type of blocking, the truck's release time t_n^R and completion time of j are determined, again by taking into account the eligibility of skilled operators (Lines 13-22).

The succeeding truck tasks - *Verification*, *Transfer* and *Disconnection* - are not subject to blocking and can therefore be assigned directly to operators (Line 28). The *Scheduling Phase* ends by returning the complete schedule for all trucks and assigned operators.

Algorithm 2: *Scheduling phase*

```
1  P ← ∅;                                                    ▷ initialize schedule
2  A ← list of assignments returned by Algorithm 1;
3  for every assignment a ∈ A do
4      n ← truck associated with a;
5      j ← task associated with a;
6      if typeOf(j) = Connection then
7          t_n^R ← t_n^A;                          ▷ set release time = truck arrival time
8          if no blocking occurs then
9              completion(j) ← assignOperator(t_n^R, j);
10         else
11             for every scheduled truck n_p ∈ P whose path intersects with n do
12                 b ← blocking type between n_a and n_p;
13                 if b = DIRECT BLOCKING then
14                     t_n^R ← max(t_n^R, t_{n_p}^D);
15                     completion(j) ← assignOperator(t_n^R, j);
16                 else if b = INDIRECT BLOCKING then
17                     if n is trailing then
18                         completion(j) ← assignOperator(t_n^R, j);
19                         completion(j) ← max(completion(j), t_{n_p}^R);
20                     else if n is leading then
21                         t_n^R ← max(t_n^A, t_{n_p}^R);
22                         completion(j) ← assignOperator(t_n^R, j);
23                     end
24                 end
25             end
26         end
27     else if typeOf(j) = transfer then
28         completion(j) ← completion(j − 1) + d_j;
29     else
30         completion(j) ← assignOperator(completion(j − 1), j);
31     end
32     P ← P ∪ {a};                                          ▷ insert a into schedule P
33 end
34 return P;
```

The *assignOperator* component embedded in the *Scheduling Phase* is detailed in Algorithm 3. Given a task j and its release time as input, the earliest available skilled operator is assigned and the completion time of task j is returned. This component works as follows. For each operator $o \in O_j$, their schedule K_o - which may already consist of scheduled tasks - is iterated over sequentially and the earliest feasible start time of task j is determined, while respecting that tasks should not overlap. When a feasible insertion time is identified, the completion time of task j by operator o is set by taking into account the task duration d_j (Lines 7-12). If the starting time of this feasible insertion's operator is no later than the task's release time, it means that this operator can begin the task immediately at the given release time and is therefore assigned to the task without considering any other operators (Lines 13-15). Once a feasible insertion is found we stop iterating over the tasks which are already assigned to the operator (Line 17). If an insertion in the operator's schedule cannot be found, j is appended to the operator's schedule after their last scheduled task and the completion time of task j by operator o is updated accordingly (Lines 20 and 21).

This process ensures that task j is assigned to the operator who can complete it the earliest (Lines 23-25). The *assignOperator* component then terminates by updating the best operator's schedule to include j and returning j's completion time.

Algorithm 3: Operator scheduling

```
1  Function assignOperator(release time, j)
2      O_j ← set of skilled operators for j;
3      d_j ← duration of task j;
4      isInserted ← false;
5      earliestCompletion ← MAX_VALUE;        ▷ Initialize earliest completion
6      for operator o ← 1 to |O_j| do
7          K_o = {k_1, ..., k_|K_o|};          ▷ Set of tasks already assigned to o
8          for task k ← 2 to |K_o| do      ▷ Sequentially iterate over tasks k ∈ K_o
9              start(o) ← max(completion(k − 1), release time);
10             if start(o) + d_j ≤ start(k) then
11                 completion(o) ← start(o) + d_j;
12                 isInserted ← true;
13                 if start(o) = release time then       ▷ o can start j immediately
14                     o_best ← o;
15                     o ← |O_j| + 1;
16                 end
17                 k ← |K_o| + 1;
18             end
19         end
20         if !isInserted then                    ▷ Conduct j after last task of o
21             completion(o) ← max(release time, completion(|K_o|) + d_j;
22         end
23         if completion(o) < earliestCompletion then
24             earliestCompletion ← completion(o);
25             o_best ← o;
26         end
27     end
28     K_o_best ← K_o_best ∪ j;                   ▷ Add j to schedule of o_best
29     return earliestCompletion;
```

5.2 Refinement procedure

We introduce four types of neighborhood for the refinement procedure. At each iteration, one of these neighborhoods is randomly selected. In the selected neighborhood, one neighboring solution is also chosen at random. The current solution is replaced with this neighboring solution only if it satisfies the acceptance criterion detailed in Section 5.2.2. We move to the next iteration and again randomly select a neighborhood for the current solution. This procedure is repeated until the stopping criterion is met.

5.2.1 Neighborhoods

The four neighborhoods introduced can be categorized into two subgroups: the first two change the path of trucks that have already been scheduled, while the

last two change the order in which trucks are served in the terminal. All these neighborhoods require re-generating the schedule by using the *Scheduling Phase* described in Section 5.1.2.

ChangePath: Most trucks have many alternative paths. A neighboring solution of this type is constructed by randomly selecting a truck. This truck is then randomly assigned to another one of its potential paths. Next, a random available loading station on this path is selected. If all such loading stations are occupied, an already assigned one is selected.

ChangePathOfBlockedTruck: This neighborhood is a variant of *ChangePath*. The solely difference is that only trucks whose paths are blocked are selected (again randomly). When no trucks are blocked in the current schedule, *ChangePath* is employed instead.

ReInsertTruck: This neighborhood first randomly selects a truck, removes all its assignments from list A, and reinserts this sequence of assignments into a different random position in the list.

ReInsertBlockedTruck: As a variant of *ReInsertTruck*, solutions of this neighborhood are obtained by only selecting and shifting tasks of trucks which are blocked in the current schedule. When there are no trucks blocked in the current schedule, *ReInsertTruck* is employed instead.

5.2.2 Acceptance and stopping criteria

The present work employs Late Acceptance Hill Climbing [5]. Solutions obtained by the refinement procedure only replace the current solution if their quality is not worse than the solution evaluated a fixed number of iterations (say β) ago. The refinement procedure terminates when a computational runtime limit is reached, with five minutes being reasonable for real-world applicability.

Note that the neighborhoods we define only contain feasible solutions. Feasibility is ensured by assigning only eligible loading stations and paths in the *Assignment Phase*. Similarly, the schedules generated in the *Scheduling Phase* only consider the assignment of eligible operators by respecting their workload schedules.

6 Instances and experimental results

Data used in the experiments was provided by a tank terminal in a highly-congested port in Europe. The most common product types to be stored and transported are petroleum, chemicals and gases. Other services include tank-to-tank transfers and product treatments such as heating and blending. The data provided contains detailed information concerning the infrastructure (tanks, pipes, loading stations, aisles), trucks (expected arrival time, product type and quantity to transfer) and operators (skills and working hours).

Several experiments are conducted to evaluate the proposed method's performance. First, the constructive dispatching heuristic is compared against random

assignment decisions. Second, the impact of indirect blocking is assessed by comparing experimental results when indirect blocking constraints are enforced in one scenario and relaxed in the other. More specifically, by only considering direct blocking, we analyze the case where terminals enable truck overtaking although this could be physically impossible. These experiments also provide insights concerning strategic level decisions when designing terminal layouts. Third, a sensitivity analysis on the number of skilled operators is conducted in order to assess whether the current operator workforce is sufficient to perform all tasks.

In order to ease the reading of the tables and equations throughout the remainder of the paper, we introduce the following notation: let ω refer to the solution value obtained from any of the aforementioned methods. If the method requires multiple runs, ω^{avg}, ω^{best} and ω^{std} denote, respectively, the average, the best and the standard deviation values among these runs. Whenever different methods or different experimental settings of the same method are used in a single equation, we distinguish between them via four entries between brackets. These entries indicate the following information: method type (D for dispatching and R for random), initial (1) or refinement (2) solution, with (+) or without (–) indirect blocking, and the number of operators considered ($|O|$). For instance, $\omega_{(D1+11)}$ refers to initial solution value of the dispatching heuristic with direct and indirect blocking and 11 operators whereas $\omega^{best}_{(R2-10)}$ denotes the best solution value obtained from five runs of the refinement procedure considering the random heuristic with only direct blocking and 10 operators.

6.1 Instance generation

Regarding the terminal's infrastructure, a total of 193 tanks are defined and each contains only one product. The terminal consists of 21 loading stations, with each tank's product being accessible by at least one and at most 18 of these stations. There are 4156 pipe connections between tanks and loading stations and each one of them is characterized by a specific transfer rate, hence the varying transfer times T^P_{ns}. Although loading stations are connected to multiple tanks, we assume only one of these connections is activated throughout an instance's scheduling horizon.

Historical data was obtained for a time horizon of five days where a total of 350 trucks arrived at irregular times throughout the day. With truck arrivals given as input, three instance sets were generated ($|N| = 60$, 80 and 100 truck arrivals) with each set containing ten instances. Each instance spans a single day at the terminal and was generated by randomly selecting $|N|$ trucks from the historical data file, while only considering the time of day when parsing a truck's arrival time. Instances with different $|N|$ values are included in the test bed to represent different saturation levels at the real terminal, ranging from regular yard occupancy (60 arrivals) to busy (100 arrivals). Furthermore, a total of 11 operators are considered, with each operator having one or more skills. All benchmark instances have been made publicly available [1].

[1] `http://benchmark.gent.cs.kuleuven.be/scheduling_trucks_in_a_tank_terminal`

6.2 Computational setup and parameter settings

The heuristic approach was coded in Java 8 to facilitate implementation by the tank terminal who co-funded the research project. Experiments were conducted on an Intel®Core^{TM}i7-2600 CPU, 3.40 GHz computer with 31.4 GB of RAM running Ubuntu Linux 12.04 LTS. The refinement procedure requires a few parameters to be set. The stopping criterion is defined as the computation time and, in correspondence with real-world operations, is set to five minutes. Meanwhile, β is assigned a value of 25. This value was obtained through an independent fine tuning procedure by assessing the algorithm's performance for values of β ranging from 0 to 100, increased by five for each run.

In this work, it is also assumed that a truck's stay time in the terminal should not exceed 90 minutes ($T_{max} = 90$).

6.3 Evaluation of different initial solution generation methods

This section first presents the results obtained from the constructive dispatching heuristic and its improvement via the refinement procedure. Due to a confidentiality agreement with the terminal concerning operational costs and the difficulty associated with reconstructing actual schedules, it is worth recognizing that providing a thorough comparison against manual scheduling is challenging.

In Table 3, the first column indicates the instance name which also includes the number of trucks considered in the experiments. The dispatching heuristic is run once to construct an initial solution whose value is given in the second column. The refinement procedure is run five times, each time with a different seed value and a time limit of five minutes. Column 3 provides the best solution value from these five runs while Columns 4 and 5 give their standard deviation and average value, respectively. Columns 6 indicates the gap between the initial solution value and the best solution value obtained by the refinement procedure. Column 7 finally reports the gap between the initial and the average improved solution values.

These gap values are calculated as in (14) and (15).

$$\frac{\omega_{(D1+11)} - \omega^{best}_{(D2+11)}}{\omega_{(D1+11)} + \epsilon} * 100 \tag{14}$$

$$\frac{\omega_{(D1+11)} - \omega^{avg}_{(D2+11)}}{\omega_{(D1+11)} + \epsilon} * 100 \tag{15}$$

The sufficiently small $\epsilon > 0$ is added to the denominator to prevent potential division by zero in (14), (15) and all other equations detailed throughout the rest of the paper which are used for calculating similar gaps.

The refinement procedure improves initial solutions by an average of 45.4% across all instances. These results indicate a significant decrease in tardiness obtained, which could imply significant operational cost reductions.

We now repeat the same experiments using a random assignment strategy in the *Assignment Phase* and report the results in Table 4. Trucks are randomly

	Initial	Refinement			Gaps (%)	
Inst.	ω	ω^{best}	ω^{std}	ω^{avg}	**Best**	**Avg.**
60_0	5525	2640	132.6	2882.8	52.2	47.8
60_1	5153	2911	148.5	3168.6	43.5	38.5
60_2	4437	2340	142.1	2499.8	47.3	43.7
60_3	6883	2780	152.8	2972.2	59.6	56.8
60_4	5828	3245	140.8	3413.6	44.3	41.4
60_5	5195	3519	72.9	3574.2	32.3	31.2
60_6	5512	2850	190.9	3124.6	48.3	43.3
60_7	9023	4310	331.2	4824.2	52.2	46.5
60_8	5264	2115	197.5	2365.8	59.8	55.1
60_9	4331	2292	120.6	2413.4	47.1	44.3
80_0	14218	5101	755.2	6585.8	64.1	53.7
80_1	13526	6187	260.1	6602.8	54.3	51.2
80_2	7746	5030	98.5	5172.2	35.1	33.2
80_3	14097	6368	291.7	6739.0	54.8	52.2
80_4	8070	4786	104.2	4879.4	40.7	39.5
80_5	11717	6998	172.6	7204.4	40.3	38.5
80_6	10147	5621	132.2	5787.8	44.6	43.0
80_7	11874	5861	312.2	6212.0	50.6	47.7
80_8	14067	6633	719.0	7241.4	52.8	48.5
80_9	11201	5134	443.1	5635.2	54.2	49.7
100_0	21903	10197	686.3	11107.4	53.4	49.3
100_1	16847	10094	511.9	10594.2	40.1	37.1
100_2	25077	12399	501.1	13068.0	50.6	47.9
100_3	21231	10074	642.4	10988.2	52.6	48.2
100_4	17765	10875	305.2	11252.0	38.8	36.7
100_5	17250	9728	486.9	10197.4	43.6	40.9
100_6	32695	14024	888.9	15542.2	57.1	52.5
100_7	20928	10532	412.5	10984.4	49.7	47.5
100_8	14950	8878	375.5	9180.2	40.6	38.6
100_9	22264	9145	637.3	9696.4	58.9	56.4
Avg.	12824.1	6422.2	345.6	6863.7	**48.8**	**45.4**

Table 3 Total truck tardiness (in minutes) after running the dispatching heuristic (which imitates the current practice at the terminal) and its refinement procedure.

assigned to eligible loading stations and paths in the loading yard which are currently unassigned. This random assignment is similar to Algorithm 1, although now Line 5 is replaced by:

$$s_n \leftarrow \text{random loading station } s \in S_n$$

When no unassigned loading station is currently available, trucks are randomly assigned to the ones which are already occupied. For the remainder of this section, we will refer to this heuristic as the 'random heuristic'.

The random heuristic is run five times, each time with a different seed value which leads to five different random assignments of trucks to loading stations and paths. Therefore the values in Column 2 of Table 4 are averaged over five runs. The refinement procedure is then used to improve each of these initial solutions and is run for five minutes as before. We observe that the standard deviation of the solution values after the refinement procedure is almost doubled when employing the random heuristic (674.4) compared to using the dispatching heuristic (345.6).

Table 5 compares the solution values obtained by the dispatching and random heuristics as well as after their improvement via the refinement procedure. Column

	Initial	Refinement			Gaps (%)	
Inst.	ω^{avg}	ω^{best}	ω^{std}	ω^{avg}	**Best**	**Avg.**
60_0	5336.2	2355	310.9	2769.0	37.5	48.1
60_1	5848.8	3001	177.7	3241.8	40.3	44.6
60_2	5640.4	2350	167.4	2607.4	43.7	53.8
60_3	6546.8	2922	890.8	3531.0	31.9	46.1
60_4	5870.6	3232	178.8	3400.2	26.5	42.1
60_5	6911.4	3520	160.4	3691.6	22.2	46.6
60_6	5475.0	2541	53.6	2615.8	12.2	52.2
60_7	8358.4	4341	322.9	4853.8	11.3	41.9
60_8	5124.0	2590	350.9	3090.6	33.2	39.7
60_9	4919.8	2522	219.5	2803.2	42.3	43.0
80_0	13918.2	4425	1139.7	6381.6	38.7	54.1
80_1	14549.2	6078	492.1	6827.0	34.9	53.1
80_2	10852.2	4738	917.4	6149.6	27.5	43.3
80_3	13700.8	5669	757.4	6966.0	47.1	49.2
80_4	10413.2	5128	678.2	5913.2	33.0	43.2
80_5	15226.8	6109	971.3	7421.0	47.1	51.3
80_6	10585.8	5049	534.9	5716.2	46.1	46.0
80_7	11779.8	6121	261.8	6565.0	36.3	44.3
80_8	13123.2	6029	313.4	6343.8	37.6	51.7
80_9	12491.2	5416	715.6	6175.2	37.8	50.6
100_0	22460.0	9064	1206.1	11107.0	45.2	50.5
100_1	18542.8	9839	252.4	10079.8	31.7	45.6
100_2	24943.6	11908	2310.8	14270.8	42.9	42.8
100_3	24688.4	9863	690.6	11172.8	51.6	54.7
100_4	23627.6	11253	1283.2	12690.2	40.6	46.3
100_5	20392.4	10873	636.7	11376.8	42.1	44.2
100_6	31445.6	13790	1862.9	15354.0	52.3	51.2
100_7	21469.2	10378	1147.7	11797.6	41.8	45.0
100_8	17043.2	9294	392.3	9839.2	41.3	42.3
100_9	23175.8	8841	835.0	10364.8	52.5	55.3
Avg.	13815.3	6308.0	674.4	7170.5	**37.6**	**47.4**

Table 4 Total truck tardiness (in minutes) after running the random heuristic and its refinement procedure.

6 reports the gaps between the initial solution values given in Columns 2 and 4, while Column 7 gives the gaps between the average solution values reported in Columns 3 and 5. These gaps are calculated as in (16) and (17).

$$\frac{\omega^{avg}_{(R1+11)} - \omega_{(D1+11)}}{\omega^{avg}_{(R1+11)} + \epsilon} * 100 \tag{16}$$

$$\frac{\omega^{avg}_{(R2+11)} - \omega^{avg}_{(D2+11)}}{\omega^{avg}_{(R2+11)} + \epsilon} * 100 \tag{17}$$

When we compare the initial solution quality of the two methods, we observe that it is highly instance-dependent. The dispatching heuristic performs 7.1% better on the average, compared to the random heuristic. The gaps after employing the refinement procedure upon these initial solutions are rather small, which implies that the choice of method for constructing the initial solution is less significant after improvement. Since the performance of the dispatching heuristic is slightly better in terms of solution quality and robustness, the initial solutions are constructed using this method for the remaining experiments.

	Dispatching		Random		Gaps (%)	
	Initial	Refinement	Initial	Refinement	Initial	Refinement
Inst.	ω	ω^{avg}	ω^{avg}	ω^{avg}		
60_0	5525	2882.8	5336.2	2769.0	-3.5	-4.1
60_1	5153	3168.6	5848.8	3241.8	11.9	2.3
60_2	4437	2499.8	5640.4	2607.4	21.3	4.1
60_3	6883	2972.2	6546.8	3531.0	-5.1	15.8
60_4	5828	3413.6	5870.6	3400.2	0.7	-0.4
60_5	5195	3574.2	6911.4	3691.6	24.8	3.2
60_6	5512	3124.6	5475.0	2615.8	-0.7	-19.5
60_7	9023	4824.2	8358.4	4853.8	-8.0	0.6
60_8	5264	2365.8	5124.0	3090.6	-2.7	23.5
60_9	4331	2413.4	4919.8	2803.2	12.0	13.9
80_0	14218	6585.8	13918.2	6381.6	-2.2	-3.2
80_1	13526	6602.8	14549.2	6827.0	7.0	3.3
80_2	7746	5172.2	10852.2	6149.6	28.6	15.9
80_3	14097	6739.0	13700.8	6966.0	-2.9	3.3
80_4	8070	4879.4	10413.2	5913.2	22.5	17.5
80_5	11717	7204.4	15226.8	7421.0	23.1	2.9
80_6	10147	5787.8	10585.8	5716.2	4.1	-1.3
80_7	11874	6212.0	11779.8	6565.0	-0.8	5.4
80_8	14067	7241.4	13123.2	6343.8	-7.2	-14.1
80_9	11201	5635.2	12491.2	6175.2	10.3	8.7
100_0	21903	11107.4	22460.0	11107.0	2.5	0.0
100_1	16847	10594.2	18542.8	10079.8	9.1	-5.1
100_2	25077	13068.0	24943.6	14270.8	-0.5	8.4
100_3	21231	10988.2	24688.4	11172.8	14.0	1.7
100_4	17765	11252.0	23627.6	12690.2	24.8	11.3
100_5	17250	10197.4	20392.4	11376.8	15.4	10.4
100_6	32695	15542.2	31445.6	15354.0	-4.0	-1.2
100_7	20928	10984.4	21469.2	11797.6	2.5	6.9
100_8	14950	9180.2	17043.2	9839.2	12.3	6.7
100_9	22264	9696.4	23175.8	10364.8	3.9	6.4
Avg.	12824.1	6863.7	13815.3	7170.5	**7.1**	**4.1**

Table 5 Results obtained by the refinement procedure using different initial solution generation methods. All ω values are in minutes.

6.4 Impact of blocking constraints

Experiments are also conducted to assess the impact of indirect blocking constraints given that they are considered to be the problem's primary bottleneck. The dispatching heuristic and the refinement procedure is employed under two scenarios. In the first scenario, indirect blocking is relaxed by only enforcing direct blocking and thus allowing truck overtaking. In the second, both direct and indirect blocking are enforced, representing the real-world scenario. Both runs of the algorithm are terminated after five minutes.

Table 6 provides the results obtained from the comparison of the two aforementioned blocking scenarios. This table's second and third columns show the best and average tardiness values over five runs for only direct blocking, whereas the fourth and fifth columns depict these values for the case with both direct and indirect blocking enforced. The final two columns present the best and the average gaps, calculated as in (18) and (19), respectively.

$$\frac{\omega^{best}_{(D2+11)} - \omega^{best}_{(D2-11)}}{\omega^{best}_{(D2+11)} + \epsilon} * 100 \tag{18}$$

$$\frac{\omega^{avg}_{(D2+11)} - \omega^{avg}_{(D2-11)}}{\omega^{avg}_{(D2+11)} + \epsilon} * 100 \tag{19}$$

The results exhibit an average of 27.0% calculated from the best gaps over all runs and instances. Similarly, an average of 28.6% is reported with respect to the average gaps.

Although the ambition of current experiment is not to precisely quantify the cost introduced by enforcing blocking, it clearly emphasizes the significant impact indirect blocking has upon the solution quality. Furthermore, it raises an interesting discussion regarding terminal design since enabling trucks to overtake could significantly reduce tardiness values. Although modifying the existing terminal design may be impractical, insights obtained from the conducted experiments may assist strategical decision makers when designing new terminals.

	Direct blocking		**Direct and indirect**		**Gap (%)**	
Inst.	ω^{best}	ω^{avg}	ω^{best}	ω^{avg}	**Best**	**Avg.**
60_0	1941	1963.8	2640	2882.8	26.5	31.9
60_1	1952	2110.6	2911	3168.6	32.9	33.4
60_2	1761	1811.6	2340	2499.8	24.7	27.5
60_3	2038	2142.0	2780	2972.2	26.7	27.9
60_4	2297	2336.4	3245	3413.6	29.2	31.6
60_5	1869	1933.2	3519	3574.2	46.9	45.9
60_6	1976	2009.0	2850	3124.6	30.7	35.7
60_7	2201	2322.4	4310	4824.2	48.9	51.9
60_8	1789	1847.0	2115	2365.8	15.4	21.9
60_9	1613	1740.2	2292	2413.4	29.6	27.9
80_0	3695	3817.6	5101	6585.8	27.6	42.0
80_1	4129	4284.8	6187	6602.8	33.3	35.1
80_2	4027	4207.8	5030	5172.2	19.9	18.6
80_3	4611	4635.2	6368	6739.0	27.6	31.2
80_4	3928	3994.4	4786	4879.4	17.9	18.1
80_5	4406	4776.8	6998	7204.4	37.0	33.7
80_6	4859	4999.4	5621	5787.8	13.6	13.6
80_7	4517	4591.6	5861	6212.0	22.9	26.1
80_8	3686	4161.8	6633	7241.4	44.4	42.5
80_9	3968	4097.4	5134	5635.2	22.7	27.3
100_0	7991	8280.8	10197	11107.4	21.6	25.4
100_1	7126	7836.2	10094	10594.2	29.4	26.0
100_2	9988	10280.8	12399	13068.0	19.4	21.3
100_3	7324	7780.8	10074	10988.2	27.3	29.2
100_4	9004	9487.0	10875	11252.0	17.2	15.7
100_5	7609	8047.8	9728	10197.4	21.8	21.1
100_6	10168	10905.4	14024	15542.2	27.5	29.8
100_7	8233	8315.6	10532	10984.4	21.8	24.3
100_8	6930	7458.6	8878	9180.2	21.9	18.8
100_9	7117	7482.8	9145	9696.4	22.2	22.8
Avg.	4758.4	4988.6	6422.2	6863.7	**27.0**	**28.6**

Table 6 Comparison of the total truck tardiness with and without indirect blocking by using the refinement procedure. All ω values are in minutes.

6.5 Sensitivity analysis on the number of operators

In addition to blocking, delays may also occur due to the unavailability of skilled operators. In this experiment, we show the impact of the number of operators upon truck tardiness. Both the dispatching heuristic and the refinement procedure are run when considering one operator fewer (10), one additional operator (12) and when doubling the total number of operators (22). The operator to add or remove was randomly selected from the set of those who are skilled in connection and/or disconnection. When doubling the number of operators, each operator in the original set was duplicated. Similar to previous experiments, the dispatching heuristic is run once while the refinement procedure is run five times with a computational runtime limit of five minutes.

Table 7 reports the difference in tardiness - both in terms of gap percentages and absolute values - for these different numbers of operators. Tardiness values are aggregated per instance size (60, 80, and 100 trucks), after which the average value per size is reported. Column 1 reports the number of operators considered ($|O|$), while Column 2 reports the instance sizes ($|N|$). Columns 3-5 provide the gaps between the solution values obtained with the default number of operators, and those considered in the respective instance group. Similar to the previous tables, the gap and absolute values are reported for the initial solutions and the best and average solution values obtained after the refinement procedure. Gaps are calculated as follows for $|O| \in \{10, 12, 22\}$:

$$\frac{\omega_{(D1+11)} - \omega_{(D1+|O|)}}{\omega_{(D1+11)} + \epsilon} * 100 \tag{20}$$

$$\frac{\omega^{best}_{(D2+11)} - \omega^{best}_{(D2+|O|)}}{\omega^{best}_{(D2+11)} + \epsilon} * 100 \tag{21}$$

$$\frac{\omega^{avg}_{(D2+11)} - \omega^{avg}_{(D2+|O|)}}{\omega^{avg}_{(D2+11)} + \epsilon} * 100 \tag{22}$$

When one operator fewer is considered, the solution values are 17.1% worse on average with an increased tardiness of 1239.8 minutes compared to the base case of 11 operators. On the contrary, adding one operator to the existing crew results in a decrease of 6.4% on average, with a reduction of 480.4 minutes in tardiness. When the entire operator crew is doubled, the average improvement is 14.9%, and the tardiness value is decreased by 1128.8 minutes.

We observe that the availability of operators may, in addition to other factors such as blocking, have an important impact on truck tardiness values. Although it may seem advantageous to intuitively employ an extensive number of operators, it must also be noted that operators incur substantial labor costs for the terminal. Whereas a marginally lower improvement is obtained when adding one operator (6.4%) in contrast to the scenario where all operators are doubled (14.9%), the incurred labor cost for hiring skilled employees will most likely not justify this tactical decision. Therefore, a comprehensive cost analysis must be carried out while taking into account real operation costs.

		Gaps (%)			Absolute difference		
		Initial	Refinement		Initial	Refinement	
$\|O\|$	$\|N\|$		Best	Avg.		Best	Avg.
10	60	-11.4	-13.7	-14.2	-653.7	-398.1	-443.9
	80	-12.8	-17.5	-17.8	-1488.6	-1010.6	-1102.4
	100	-8.2	-19.0	-19.3	-1722.7	-2017.7	-2173.0
	Avg.	-10.8	-16.8	**-17.1**	-1288.3	-1142.1	**-1239.8**
12	60	6.3	3.6	5.6	362.0	104.0	175.6
	80	2.6	4.9	5.3	307.6	284.7	330.4
	100	4.0	8.1	8.3	848.9	860.1	935.3
	Avg.	4.3	5.5	**6.4**	506.2	416.3	**480.4**
22	60	11.8	6.4	10.2	657.6	183.1	307.1
	80	7.4	12.7	14.0	817.4	724.0	861.6
	100	10.1	18.8	20.4	1857.3	1909.2	2217.6
	Avg.	9.8	12.6	**14.9**	1110.8	938.8	**1128.8**

Table 7 Gaps and absolute difference (in minutes) with respect to tardiness values when considering 10 (decreased workforce), 12 (increased workforce) and 22 (double workforce) operators.

7 Conclusions and future research

This paper introduced a real-world truck scheduling problem in a tank terminal where operational space is at a premium. The compactness of the terminal yard results in a unique situation where blocking may occur not only at the loading stations, but also on their paths. The latter is referred to as indirect blocking and it is experimentally observed through this study that this type of blocking has a significant impact on the trucks' tardiness. A comparative analysis with respect to previously studied problems where blocking occurs helped identify the fundamental differences regarding blocking constraints. The challenging task of solving the problem as an MIP led us to developing a tailored heuristic.

The heuristic approach introduced is capable of constructing schedules in reasonable computation time, while efficiently handling the routing and scheduling decisions via a schedule generator. A benchmark dataset created by using the historical data obtained from a real-world tank terminal should help encourage future research regarding this topic. Experimental results demonstrate that the introduced approach performs significantly better than the currently-employed manual dispatching strategy, while also demonstrating the impact of blocking on the solution quality. Furthermore, the sensitivity analysis on the number of operators indicates that a considerable portion of the total tardiness may be due to insufficient workforce.

It is worthwhile to investigate the MIP formulation introduced in this paper to fully understand its merits in solving the TSTT or potential problem variants. It may also be beneficial to investigate other real-world contexts in which indirect blocking, or a variant thereof, is present. The authors hypothesize that indirect blocking will likely be present in other spatially-restrictive yet highly interconnected industrial environments through which vehicles of some type must pass. Furthermore, since the presence of indirect blocking often incurs high oper-

ational daily costs, follow-up research focused on strategical insights concerning the structural layout of terminals may help those who are planning to construct new terminals build ones which are not highly prone to indirect blocking.

Acknowledgments

This work was supported by Agidens, Oiltanking Stolthaven Antwerp NV (OTSA), Leuven Mobility Research Center, Dynamical Combinatorial Optimization (KU Leuven C2 project C24/17/012) and Data-driven logistics (FWO-S007318N). This research was partly funded by VLAIO research project 140876, and the Flemish Government under the programme "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen". Editorial consultation provided by Luke Connolly (KU Leuven).

References

1. Adams, J., Balas, E., and Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management science*, 34(3):391–401.
2. Balas, E. (1969). Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. *Operations research*, 17(6):941–957.
3. Brucker, P., Jurisch, B., and Sievers, B. (1994). A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics*, 49(1):107 – 127.
4. Burdett, R. L. and Kozan, E. (2018). An integrated approach for scheduling health care activities in a hospital. *European Journal of Operational Research*, 264(2):756–773.
5. Burke, E. K. and Bykov, Y. (2017). The late acceptance hill-climbing heuristic. *European Journal of Operational Research*, 258(1):70 – 78.
6. Carlier, J. and Pinson, É. (1989). An algorithm for solving the job-shop problem. *Management science*, 35(2):164–176.
7. Ernst, A., Jiang, H., Krishnamoorthy, M., and Sier, D. (2004). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3 – 27.
8. Gawrilow, E., Klimm, M., Möhring, R. H., and Stenzel, B. (2012). Conflict-free vehicle routing. *EURO Journal on Transportation and Logistics*, 1(1):87–111.
9. Gawrilow, E., Köhler, E., Möhring, R. H., and Stenzel, B. (2008). Dynamic routing of automated guided vehicles in real-time. In *Mathematics–Key Technology for the Future*, pages 165–177. Springer.
10. Karasek, J. (2013). An overview of warehouse optimization. *International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems*, 2(3):111–117.
11. Lange, J. and Werner, F. (2018). Approaches to modeling train scheduling problems as job-shop problems with blocking constraints. *Journal of Scheduling*, 21(2):191–207.
12. Liu, S. Q. and Kozan, E. (2009). Scheduling trains as a blocking parallel-machine job shop scheduling problem. *Computers & Operations Research*, 36(10):2840 – 2852.

13. Mascis, A. and Pacciarelli, D. (2002). Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498 – 517.
14. Mason, S., Fowler, J., Carlyle, W., and Montgomery, D. (2005). Heuristics for minimizing total weighted tardiness in complex job shops. *International Journal of Production Research*, 43(10):1943–1963.
15. Mason, S. J., Fowler, J. W., and Matthew Carlyle, W. (2002). A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops. *Journal of Scheduling*, 5(3):247–262.
16. Meng, L. and Zhou, X. (2014). Simultaneous train rerouting and rescheduling on an n-track network: A model reformulation with network-based cumulative flow variables. *Transportation Research Part B: Methodological*, 67:208–234.
17. Roy, B. and Sussmann, B. (1964). Les problèmes d'ordonnancement avec contraintes disjonctives. *Note DS*, No. 9 bis.
18. Samà, M., D'Ariano, A., D'Ariano, P., and Pacciarelli, D. (2017). Scheduling models for optimal aircraft traffic control at busy airports: Tardiness, priorities, equity and violations considerations. *Omega*, 67:81 – 98.
19. Samà, M., D'Ariano, A., Pacciarelli, D., Palagachev, K., and Gerdts, M. (2017). Optimal aircraft scheduling and flight trajectory in terminal control areas. In *5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 285–290. IEEE Washington, DC.
20. Törnquist, J. and Persson, J. A. (2007). N-tracked railway traffic re-scheduling during disturbances. *Transportation Research Part B: Methodological*, 41(3):342–362.
21. Zhou, X. and Zhong, M. (2007). Single-track train timetabling with guaranteed optimality: Branch-and-bound algorithms with enhanced lower bounds. *Transportation Research Part B: Methodological*, 41(3):320 – 341.

Appendix A

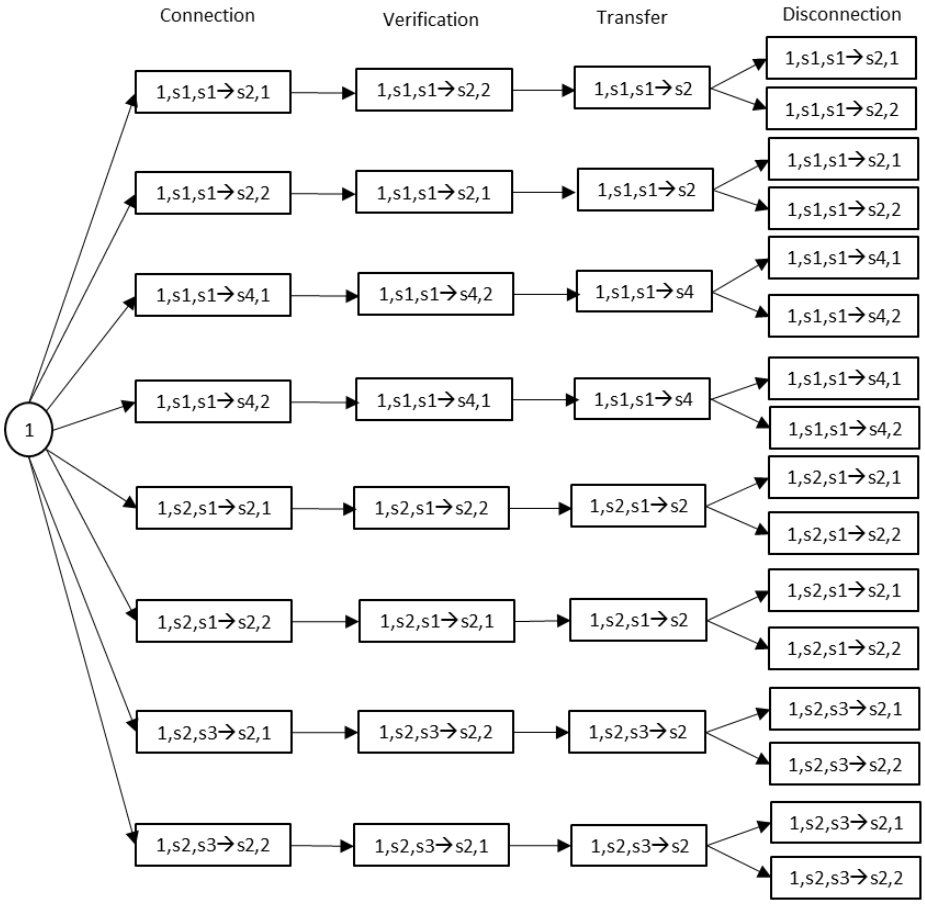

Fig. 6 Enumeration of feasible assignments for truck 1 in the problem instance of Figure 4. Each node is represented by truck #, loading station, path (excluding nodes 0 and $*$), operator #, respectively.

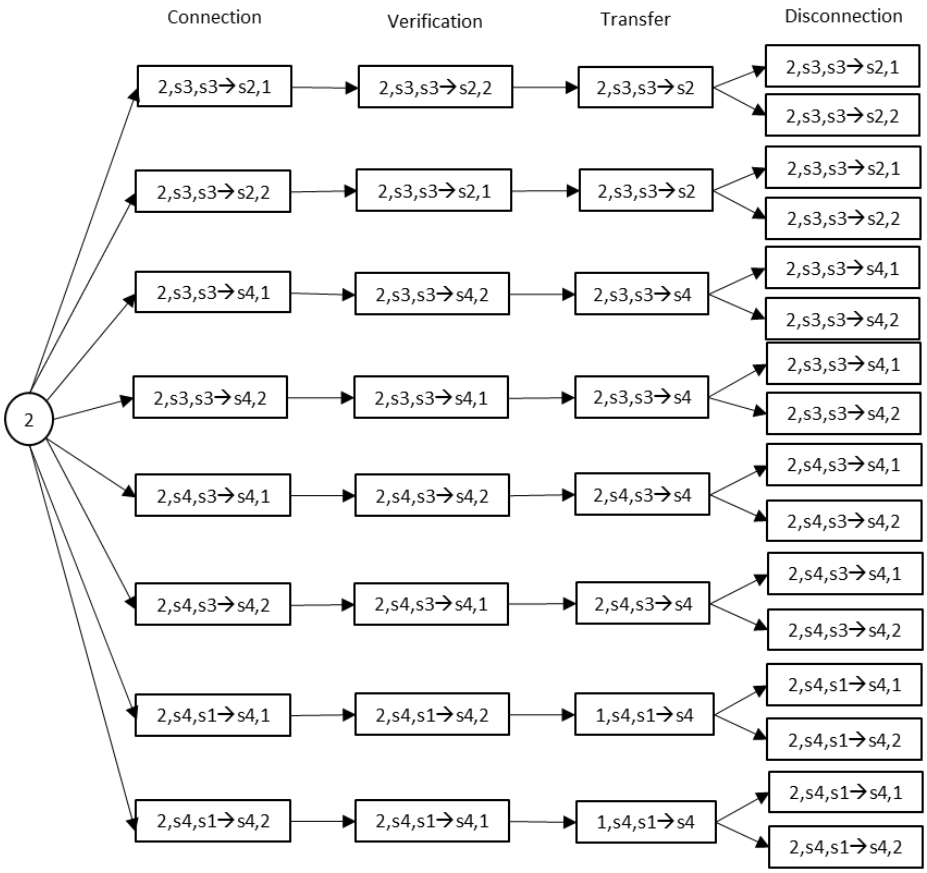

Fig. 7 Enumeration of feasible assignments for truck 2 in the problem instance of Figure 4. Each node is represented by truck #, loading station, path (excluding nodes 0 and $*$), operator #, respectively.