Simulation-Based Optimization for Housekeeping in a Container Transshipment Terminal

Jean-François Cordeau[a], Pasquale Legato[b,*], Rina Mary Mazza[b], Roberto Trunfio[c]

[a]*HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, H3T 2A7 Canada*
[b]*DIMES, Università della Calabria, 87036, Rende (CS), Italy*
[c]*DLISE, Università della Calabria, 87036, Rende (CS), Italy*

Abstract

An important activity in container transshipment terminals consists in transferring containers in the yard from their current temporary positions to different positions closer to the point along the berth from which the containers will be boarded on departing vessels. This housekeeping process aims at speeding-up discharge and loading operations and, thus, relieving congestion. This article introduces a heuristic procedure to manage the routing of multi-trailer systems and straddle carriers in a maritime terminal. A simulation model embedded in a local search heuristic allows a proper evaluation of the impact of different vehicle schedules on congestion and throughput. Computational experiments performed on test instances derived from real-life data show that important improvements in routing distance and vehicle waiting time can be obtained compared to the use of standard scheduling policies.

Keywords: Container Terminal, Logistics, Vehicle Routing, Simulation-Optimization

1. Introduction

The recent contraction in trade resulting from the present economic downturn has somewhat eased and, according to the IMF - International Monetary Fund, world trade is expected to increase by 3.6% in 2013 [1]. Since the amount of goods transported by sea accounts for approximately 90% of the world's total freight volume, the availability, low cost and efficiency of maritime transport via container will remain vital to the functioning of the global economy.

Although a high barrier to entry generally applies for the new players in the international container shipping industry, those already in the market will in any case have to spare no efforts to capture favorable business and turn adversity into opportunity. To do so, they will need to respond to the growing demand for their services by emphasizing the development of their core business and strengthening their competitiveness. In particular, container terminal operators will need to efficiently provide flexible, yet slim handling and storage solutions to their clients, while keeping productivity and quality up to standards.

In this sense, a major limit on the efficiency of container terminal operators is certainly represented by congestion. Congestion carries with it loss of time, rise of cost and drop in quality of service for both shipping companies and terminal operators, as well as significant deterioration of the overall terminal performance. According to a macroeconomic perspective, congestion arises when the actual demand of the shipping companies exceeds the actual capacity of the terminal facility. From an operational point of view, it can be caused by a number of factors pertaining to the container logistics chain such as the availability of limited resources, the lack of synchronization among different types of equipment dedicated to container handling and transfer and, above all, the occurrence of unforeseen events which are normally not accounted for in the terminal's organization, resource management and activity scheduling. Practically, as is well summarized in [2], various bottlenecks can be identified along the port-calling chain. A vessel

*Corresponding author
URL: jean-francois.cordeau@hec.ca (Jean-François Cordeau), legato@dimes.unical.it (Pasquale Legato), rmazza@dimes.unical.it (Rina Mary Mazza), roberto.trunfio@unical.it (Roberto Trunfio)

that is heading from open sea to a seaport may experience congestion consecutively in the following places or corridors, depending on the location and structure of the port: maritime access route, locks, berths, discharge/loading, storage, customs inspection, hinterland discharge/loading, and hinterland connections.

In the present study, we consider the standpoint of a container terminal devoted to pure transshipment (i.e. container transfer occurs from one vessel to another rather than from vessels to train or truck units). Our contribution to keeping the terminal's productivity and quality up to standards by lessening the congestion arising at the main intra-terminal bottlenecks (i.e. berthing, discharge/loading and storage processes) is centered on the so-called housekeeping process.

Housekeeping consists in transferring a container on the yard from a temporary position to a final position with the latter being closer to the point along the berth from which the container will be boarded on a departing vessel. The object of this operational procedure is to speed-up discharge/loading operations. As a practical consequence, terminal productivity will increase by minimizing service delays and profits will rise by preventing the payment of extra charges for not achieving the level of service granted to clients. Shipping companies will also benefit from the implementation of this operational procedure by keeping their vessels on schedule while travelling from one port to another.

To our knowledge, the housekeeping process has not received much attention in either theoretical or empirical research. Besides [3, 4], in which a queuing-based representation of the housekeeping process is proposed for a real container terminal and solved by discrete-event simulation, housekeeping is often mentioned only in a marginal way as a possible choice lying behind yard operating rules and, in particular, behind intelligent yard stacking [5–7]. Our work aims at filling in this modeling gap.

In this paper, both discrete-event simulation and local search are exploited to support housekeeping decisions in a complex, dynamic and stochastic environment. On the one hand, simulation-based approaches have been widely used to model various planning problems arising in container terminals. Many examples can be found with reference to yard layout [8–12], container stacking [13, 14, 6, 7], vehicle dispatching [15–17, 3, 18, 19, 12] and alternative transfer systems [9, 10, 20]. On the other hand, although they sacrifice a complete representation of the uncertainty and congestion phenomena inherent to the housekeeping process, deterministic optimization algorithms have the ability of identifying optimal or near-optimal solutions to simplified deterministic problems. To benefit from both approaches, we thus resort to *Simulation-based Optimization* (SO) [21].

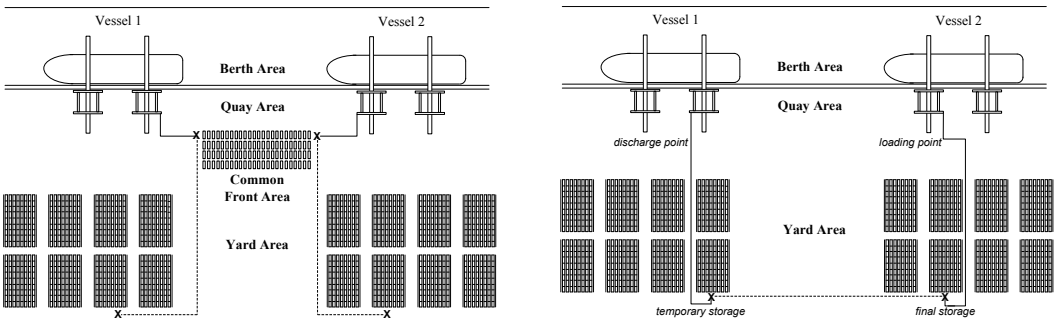
SO is a practical solution method based on the idea of embedding a simulation engine within an optimization algorithm to deal with hard combinatorial optimization problems arising in dynamic processes characterized by sources of randomness. The optimization algorithm is aimed at generating an initial feasible solution and then exploring the solution space until no further improvements of the performance measures are obtained or until the allocated computation time budget is exhausted. The use of the simulation engine is required in the evaluation process because the cost function cannot be evaluated by simply giving values to the decision variables in a closed-form formula.

The rest of the paper is organized as follows. The housekeeping process is described in Section 2 while a mathematical formulation of the problem is provided in Section 3. The SO solution approach is then presented in Section 4. This is followed by computational results in Section 5 and by conclusions in Section 6. Finally, further details on the simulation model are provided in the appendix.

2. Problem Description

A housekeeping operation consists in transferring a container from one storage location to another with the aim of speeding up discharge/loading operations and avoid congestion. This is possible because the latter storage position is supposed to be reasonably closer to the berth area along which the container will undergo future boarding on a pre-assigned departing vessel. Depending on the structure and organization of the facility, a container requiring housekeeping can be transferred from a common front area to a final storage position on the yard as illustrated by Figure 1a or between yard positions as illustrated by Figure 1b.

From here on, we will refer to the housekeeping implementation depicted in Figure 1b which mirrors precisely how housekeeping is carried out in our facility of interest: the container terminal located at the port of Gioia Tauro in Southern Italy. This terminal, which is almost entirely devoted to pure transshipment, features a very extensive horizontal yard layout (approximately 1.000.000 m^2). Rather than transfer cranes, the yard is equipped with 110 *straddle*



(a) containers are discharged to a buffer area and transferred afterwards to the yard.

(b) containers are discharged to temporary storage positions and transferred afterwards.

Figure 1: Different ways of implementing housekeeping.

carriers (SCs), i.e. very flexible and productive container handling equipment bearing an operational flexibility to both lift and transfer containers between the yard and the quay at high speed. Due to the dimensions and organization of the container yard, transfer operations are very time consuming. As a result, resorting to housekeeping becomes a very logical choice to be taken when aiming at speeding-up the container discharge/loading cycle. Practically speaking, in the above container terminal, in which the yard is divided into blocks, the housekeeping process concerns the simultaneous transfer of a batch of containers from a source block to a destination block. This generally occurs just a matter of hours before the vessel on which containers are meant to be boarded is scheduled for departure.

A step-by-step description of the housekeeping process is given in the following and summarized by Figure 2, according to which one can easily identify the pertaining congestion points. A request for housekeeping is performed by means of a transfer system; if the assigned system is unavailable, the container to be housekept waits in its current position on the yard and, thus, plays a role in creating potential congestion points that affect the overall housekeeping work schedule. Once available, the transfer system must access a specific row (or column) of the source block in order to retrieve the target container. For security reasons, no vehicle is allowed to access a row if container stacking/retrieval operations are already occurring in that row, as well as in the adjacent ones. Hence, transfer vehicles queuing in front of a block are another factor likely to contribute to congestion. Once access is granted to the row, the transfer system must also provide for handling: it requires time to reach the stack with the target container, eventually shuffle other containers located on top and perform the actual pick-up operation. Then, container transfer from the source block to the destination block follows and depends on the distance lying in between the blocks. Stacking operations are initiated at the destination block only if the transfer system gains immediate access to the row of interest; otherwise, the transfer vehicles must queue with the loaded container(s) in front of the block. Again, congestion is likely to occur. Once access is granted to the vehicles, container stacking within the row closes the housekeeping work cycle.

The logic behind housekeeping is quite clear. However, whether or not to perform housekeeping operations depends on the distance to cover when transferring a discharged container from its current position along the quay to its storage location on the yard. Let d be this distance. If d is less than a fixed threshold, then a *direct transfer* occurs, meaning that the container is definitely transferred to the yard position or slot in which it will remain until rescheduled for departure on another vessel. In the specific terminal under analysis, direct transfer is implemented with human operated SCs and, for our convenience, we refer to this as D-SCs. If d is greater than the aforesaid threshold, then containers will be stacked in a temporary position on the yard and undergo *indirect transfer* or housekeeping only afterwards. In this case, different types of transfer systems are used: SCs providing for housekeeping and referred to H-SCs are used over short distances, whereas multi-trailer systems (MTSs) over long distances. MTSs combine *i*) SCs dedicated to container discharge/loading from/on MTSs and referred to M-SCs and *ii*) tractor units to haul the trailers from one yard block to another. The number of trailers used by an MTS can affect the way to carry out the working cycle of the housekeeping process. To fix ideas, one may consider the configuration in which five trailers are used against three trailers. In the first case, the time required by the SCs to discharge/load a multi-trailer is certainly greater than the second. Thus, to avoid an unacceptable idle time for the associated tractor, the same tractor may be assigned to two different multi-trailers such that while one is busy being discharged/loaded the other can undergo

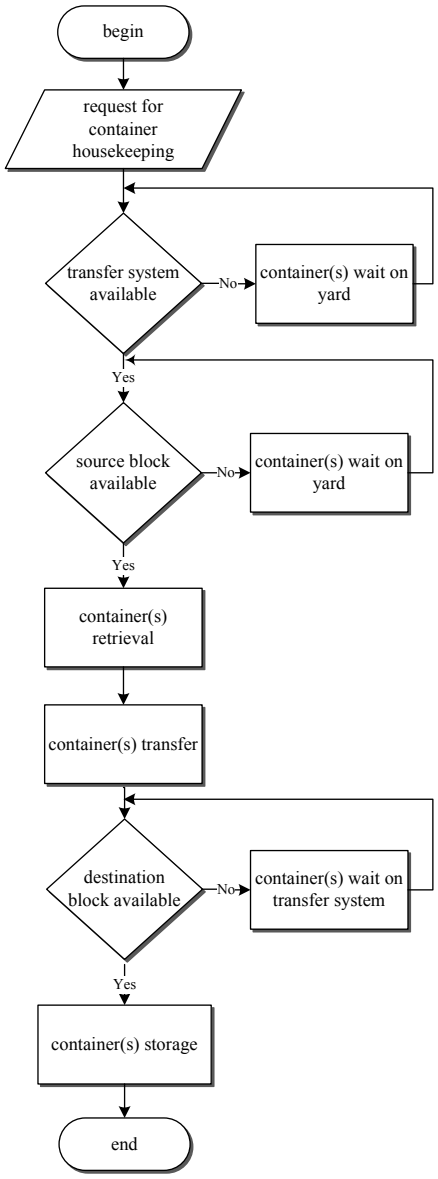


Figure 2: General view of the housekeeping process.

transfer from one yard block to another. This will increase both the utilization of the tractor and the productivity of the entire housekeeping process. In the case in which only three trailers are hauled by a tractor, then the shorter discharge/loading times will likely not allow the same tractor to circle between two different multi-trailers working at two different blocks without affecting the average waiting of the multi-trailers. So, each multi-trailer will have its own tractor for transfer purposes.

The values for both of the previously described *modus operandi* in the container terminal under analysis are given in Table 1. As one may also observe from the information reported in column four, when housekeeping is performed via SC one container is transferred at a time, whereas a batch of containers is transferred if MTSs are used.

Whatever be the transfer system used, at the beginning of the housekeeping process, the initial location of every single vehicle can vary according to the activities in which it was previously involved. With respect to this matter, different physical areas can be assigned to the above means as initial location or depots of reference. In the specific

Distance (d)* [m]	Modality	Transfer system	Cargo
$0 < d \leqslant 600$	direct transfer	D-SCs	single container
$600 < d \leqslant 800$	housekeeping	H-SCs	single container
$d > 800$	housekeeping	MTSs (supported by a couple of M-SCs)	batch of containers

*whatever the distance, reefer containers are always transferred with SCs.

Table 1: Different modalities of internal container transfer.

case at hand, three different depots can be conceived: depot 1 is a gathering point for vehicles that have undergone maintenance and/or completed refueling operations; depot 2 is used as reference area for vehicles that have been washed and are currently available for deployment in container transfer operations; depot 3 is used as exchange point for vehicles that are already working on the terminal yard. When housekeeping operations are being examined according to an operational perspective, i.e. the length of the time horizon is in the order of hours, it is often sufficient to consider only depot 3.

Once initiated, although a housekeeping operations is meant to be carried out separately from other housekeeping operations and direct transfers, interference among the vehicles at work may occur. In particular:

- An MTS may interfere with another MTS if they are both scheduled to work in the same yard block (case *MTS-MTS*).
- An SC may interfere with another SC if they are both scheduled to work in the same row or in adjacent rows of a yard block (case *SC-SC*).
- An SC may interfere with an MTS if any of the containers to be handled by the MTS are located in the same row or in adjacent rows in which an SC is scheduled to work (case *SC-MTS*).

At this point, it should be clear that interference among vehicles is one of the major sources of congestion and thus it must be taken into account when modeling the housekeeping process.

3. Approaches to problem modeling and solution

The housekeeping problem described in the previous section calls for an effective optimization approach to support the rational management of a heterogeneous fleet of vehicles employed to carry out a set of tasks in a fixed order. A task for the H-SC is a single container to be handled (i.e. picked-up, transferred and set-down) between a slot in a *source yard block* and a slot in a *destination yard block*. A task for the MTS is a *batch of containers* to be handled between a *source yard block* and a *destination yard block*.

Static and dynamic scheduling approaches in vehicle-based internal transport systems have been recently studied in [22] under the assumption that these systems can be suitably modeled by Integer Programming (IP) based formulations and solved by commercial tools and meta-heuristics. Scheduling studies concerning external transport are adapted for internal transport and fit to a dynamic context, with the aim of minimizing the waiting time of the scheduled tasks to be executed. Common assumptions are that: *i*) vehicles operate continuously without breakdown, *ii*) there are no traffic problems (like congestion or deadlocks) and *iii*) vehicle loading and unloading times are fixed and considered in travel times between loading and unloading locations.

Within the specific domain of a maritime container terminal, a genetic algorithm has been proposed to optimize all the internal container transfers among the quay area, yard area and truck area [23]. Transfer tasks are executed by a fleet of automated SCs whose travel cost and waiting cost should be minimized. Under the assumption that a collision free path planning has been successfully implemented, planned timings (task start and finish) as well as travel times based on theoretical distances are known with certainty and inserted within an IP formulation as parameters [**?**].

Generally speaking, under the assumption that travel times are deterministic and no queuing or blocking phenomena may arise neither at the source blocks nor at the destination blocks, a *Multiple Depot Vehicle Scheduling Problem* (MDVSP) with two types of vehicles [24] appears as the natural starting point to develop an IP based model for the housekeeping process described in Section 2. Special attention must be given to the interference between vehicles at

both the source block as well as the destination block of any single container or batch of containers to be picked-up or set-down.

In an MDVSP, a set of tasks, each defined by an origin location, a destination location, and a processing time, must be carried out by a fleet of vehicles based at several depots. Vehicles may have different features (e.g. *load capacity*, *speed*, *coverable distance*, etc.) and, thus, some types of vehicle may not be allowed to execute every task. The MDVSP consists in finding a minimum-cost set of schedules for the vehicles such that each task is performed exactly once by a vehicle. In the classical MDVSP (see, e.g., [25–27]), each task has a given starting time at which it must be performed by a vehicle. The MDVSP with Time Windows (MDVSPTW) (see, e.g., [28, 29]) is a generalization in which each task is instead required to begin within a specified time interval. The scheduling of SCs can be seen as a pure MDVSPTW whereas the scheduling of MTSs requires additional constraints to impose a minimal temporal separation between two tasks sharing a common origin or destination but assigned to different vehicles. In both problems, one must assign a sequence of tasks to each vehicle so as to minimize the total travel time of all vehicles (or some other measure of performance) while ensuring that each task is performed exactly once and all applicable constraints are satisfied. Because of interference constraints, however, one cannot separate the scheduling of MTSs and the scheduling of SCs. Instead, the two problems should be handled jointly.

The natural objective to be considered in our real environment of reference should be to maximize the number of tasks performed during the planning horizon and minimize the related, unavoidable waiting phenomena. Usually, in an IP-based formulation the minimization of the total distance traveled by the fleet of vehicles is considered, since it is expected to be inversely correlated with the number of tasks performed and therefore is recognized as a good approximation to the natural objective. However, it should be relevant to also consider a deterministic approximate measure of the waiting time in the objective function.

We propose a formulation for the housekeeping problem based on a graph where each vertex represents an individual container handling operation at a source/destination yard block and each arc represents either a container transfer operation or the relocation of a vehicle after a transfer operation. To this end, we define the following notation:

- T – the time-horizon.
- P^{SC} and P^{MTS} – the set of n_1 and n_2 task source points, respectively for the H-SCs and the MTS, such that $P^{SC} = \{1, \ldots, n_1\}$ and $P^{MTS} = \{n_1 + 1, \ldots, n_1 + n_2\}$.
- P – the global set of $n = n_1 + n_2$ task source points, such that $P = P^{SC} \cup P^{MTS}$.
- D^{SC} and D^{MTS} – the set of $\tilde{n}_1$ and $\tilde{n}_2$ task destination points, respectively for the H-SCs and the MTS, such that $D^{SC} = \{1, \ldots, \tilde{n}_1\}$ and $D^{MTS} = \{\tilde{n}_1 + 1, \ldots, \tilde{n}_1 + \tilde{n}_2\}$. Observe that, since an individual container handling operation at a source block must match with another individual container handling operation at a destination block, $n_1 = \tilde{n}_1$ and $n_2 = \tilde{n}_2$ must result.
- D – the global set of $\tilde{n} = \tilde{n}_1 + \tilde{n}_2$ task destination points, such that $D = D^{SC} \cup D^{MTS}$. Clearly $n = \tilde{n}$.
- K^{SC} – the set of m_1 H-SCs.
- K^{MTS} – the set of m_2 MTSs.
- K – the set $K = K^{SC} \cup K^{MTS}$ of $m = m_1 + m_2$ transfer vehicles.
- S, F – the starting and final depot of a transfer vehicle, respectively.
- p_i – the processing time of task i. It corresponds to: the container pickup-up time if $i \in P^{SC}$; the container set-down time if $i \in D^{SC}$; the time required to handle the batch of containers in the source yard block if $i \in P^{MTS}$; the time required to handle the batch of containers in the destination yard block if $i \in D^{MTS}$. Notice that $p_S = p_F = 0$.
- r_i – the release time of task i.
- d_i – the due date for task i.

- t_{ij} – the time required by a vehicle to move from the current vertex i to the next vertex j.
- t^k_{Si} – the time required by vehicle k to move from its initial location to the location of i.
- t^k_{iF} – the time required by vehicle k to move from the location of i to the final location of k.
- h_{ij} – with $i \in P, j \in D$ is equal to 1 if and only if the container or batch of containers has to be handled from i to j; 0 otherwise.
- P_k and D_k – the set of task source points and task destination points, respectively, allowed to be serviced by vehicle k. In particular, we assume that $P_k = P^{SC}$ and $D_k = D^{SC}$ for all the vehicles in K^{SC} and that $P_k = P^{MTS}$ and $D_k = D^{MTS}$ for all the vehicles in K^{MTS}.
- K_i – the set of vehicles allowed to service a source/destination point i. In particular, we assume that $K_i = K^{SC}$ for all points in P^{SC} or D^{SC}; and $K_i = K^{MTS}$ for all points in P^{MTS} or D^{MTS}; and, finally, $K_S = K_F = K$.
- K_{ij} – the set of vehicles allowed to service two source/destination points i and j such that $K_{ij} = K_i \cap K_j$.
- δ – the time span that must elapse between two handling operations executed by two MTSs occurring at the same source or destination yard block to ensure that they do not interfere with each other. No time span is considered for the handling operations of the SCs. In fact, once that an SC releases an occupied row of a yard block, another SC is immediately allowed to occupy the same row or another adjacent to it. This happens because SCs, for technical constraints, enter a row from the bottom of the yard block and exit from the top.

To formally define the interference sets, given a task i let: sb_i and db_i be the source block and the destination block of i, respectively; sr_i and dr_i be the row in the source block and the row of the destination block of i, respectively; csb_i and cdb_i be the set of containers of i to be handled in source block and destination block, respectively.

According to the previous description given in Section 2, we can define three interference sets $\Phi^{MTS-MTS}$, Φ^{SC-SC}, Φ^{SC-MTS}, respectively for interferences occurring between two MTSs, two SCs or an SC and an MTS, such that:

- $\Phi^{MTS-MTS} = \{(i,j,k_1,k_2) : i,j \in S^{MTS} \cup D^{MTS}, i \neq j, k_1, k_2 \in K^{MTS}, k_1 \neq k_2, (sb_i = sb_j \vee db_i = db_j \vee sb_i = db_j \vee db_i = sb_j)\}$.
- $\Phi^{SC-SC} = \{(i,j,k_1,k_2) : i,j \in S^{SC} \cup D^{SC}, i \neq j, k_1, k_2 \in K^{SC}, k_1 \neq k_2, ((sb_i = sb_j \wedge sr_i \in [sr_j - 1, sr_j + 1]) \vee (db_i = db_j \wedge dr_i \in [dr_j - 1, dr_j + 1]) \vee (sb_i = db_j \wedge sr_i \in [dr_j - 1, dr_j + 1]) \vee (db_i = sb_j \wedge dr_i \in [sr_j - 1, sr_j + 1]))\}$.
- $\Phi^{SC-MTS} = \{(i,j,k_1,k_2) : i \in S^{SC} \cup D^{SC}, j \in S^{MTS} \cup D^{MTS}, k_1 \in K^{SC}, k_2 \in K^{MTS}, ((sb_i = sb_j \wedge \exists j^* \in csb_j \,|\, sr_i \in [sr_{j^*} - 1, sr_{j^*} + 1]) \vee (db_i = db_j \wedge \exists j^* \in cdb_j \,|\, dr_i \in [dr_{j^*} - 1, dr_{j^*} + 1]) \vee (sb_i = db_j \wedge \exists j^* \in cdb_j \,|\, sr_i \in [dr_{j^*} - 1, dr_{j^*} + 1]) \vee (db_i = sb_j \wedge \exists j^* \in csb_j \,|\, dr_i \in [sr_{j^*} - 1, sr_{j^*} + 1]))\}$.

Notice that for all the pairs of tasks in the 4-tuple in $\Phi^{MTS-MTS}$, a time-span δ has to be maintained between their operations if and only if performed by different MTSs. For convenience, introduce $\Phi = \Phi^{MTS-MTS} \cup \Phi^{SC-SC} \cup \Phi^{SC-MTS}$. For all the pairs of tasks in the 4-tuple in Φ, a non simultaneity constraint must be considered to avoid interference on the housekeeping operations. In addition, the set $\Psi = \{(i,j) \,|\, \exists k_1 \in K_i, k_2 \in K_j, (i,j,k_1,k_2) \in \Phi\}$ is introduced.

The graph G at the basis of the formulation is defined as $G = (V, A)$, where $V = \{S, F\} \cup P \cup D$ is the set of vertices and A is the arc set. For practical reasons, $A = \{(S,F)\} \cup A^{SC} \cup A^{MTS}$, where $A^{SC} = A^{SC}_1 \cup A^{SC}_2$ and $A^{MTS} = A^{MTS}_1 \cup A^{MTS}_2$ are defined as:

- $A^{SC}_1 = \{(i,j) : i \in P^{SC}, j \in D^{SC}, h_{ij} = 1\}$;
- $A^{SC}_2 = \{(i,j) : i \in D^{SC} \cup \{S\}, j \in P^{SC} \cup \{F\}, i \neq j\}$;
- $A^{MTS}_1 = \{(i,j) : i \in P^{MTS}, j \in D^{MTS}, h_{ij} = 1\}$;
- $A^{MTS}_2 = \{(i,j) : i \in D^{MTS} \cup \{S\}, j \in P^{MTS} \cup \{F\}, i \neq j\}$;

and, for convenience, $A_1 = A_1^{SC} \cup A_1^{MTS}$ and $A_2 = A_2^{SC} \cup A_2^{MTS}$.

The following decision variables are introduced:

- X_{ij}^k, is equal to 1 if and only if vertex i is visited by vehicle k immediately before vertex j, 0 otherwise.
- Z_{ij}, is equal to 1 if task i is completed before task j starts, 0 otherwise.
- Y_i^k, is an auxiliary variable equal to 1 if vertex i is assigned to vehicle k, 0 otherwise.
- B_i, is the starting time of operations on vertex i.
- W_i, is the waiting time before starting the operations on vertex i. It is computed as the difference between the beginning time of the operations on i and the time instant when the transfer vehicle assigned to i arrives at the physical location of vertex i.

The problem is formulated as follows:

$$\textbf{Problem } (\mathbf{P^H}) \quad \textit{minimize} \quad \sum_{(i,j)\in A_2} \sum_{k\in K_{ij}} t_{ij} \cdot X_{ij}^k + \sum_{i\in P\cup D} W_i \tag{1}$$

subject to

$$\sum_{j:(i,j)\in A} \sum_{k\in K_{ij}} X_{ij}^k = 1 \quad \forall i \in P \cup D \tag{2}$$

$$\sum_{j:k\in K_{Sj}} X_{Sj}^k = 1 \quad \forall k \in K \tag{3}$$

$$\sum_{i:k\in K_{iF}} X_{iF}^k = 1 \quad \forall k \in K \tag{4}$$

$$\sum_{j:(j,i)\in A} X_{ji}^k - \sum_{j:(i,j)\in A} X_{ij}^k = 0 \quad \forall i \in P \cup D, k \in K_i \tag{5}$$

$$\sum_{k\in K_{ij}} X_{ij}^k = h_{ij} \quad \forall i \in P, j \in D \tag{6}$$

$$\sum_{j:(i,j)\in A} X_{ij}^k = Y_i^k \quad \forall i \in P \cup D, k \in K_i \tag{7}$$

$$X_{ij}^k = 1 \Rightarrow \quad B_j \geqslant B_i + p_i + t_{ij} \quad \forall (i,j) \in A, k \in K_{ij} \tag{8}$$

$$r_i \leqslant B_i \leqslant d_j - p_j - t_{ij} - p_i \quad \forall (i,j) \in A_1 \tag{9}$$

$$r_j + p_j + t_{ji} \leqslant B_i \leqslant d_i - p_i \quad \forall (i,j) \in A_2 \tag{10}$$

$$X_{Sj}^k = 1 \Rightarrow \quad B_j \geqslant B_S + t_{Sj}^k \quad \forall j \in S, k \in K_j \tag{11}$$

$$X_{iF}^k = 1 \Rightarrow \quad T \geqslant B_i + t_{iF}^k \quad \forall i \in D, k \in K_i \tag{12}$$

$$X_{ij}^k = 1 \Rightarrow \quad W_j \geqslant B_j - (B_i + p_i + t_{ij}) \qquad \forall (i,j) \in A, k \in K_{ij} \tag{13}$$

$$Y_i^{k_1} + Y_j^{k_2} \leqslant 1 + Z_{ij} + Z_{ji} \quad \forall (i,j,k_1,k_2) \in \Phi \tag{14}$$

$$Y_i^{k_1} = 1, Y_j^{k_2} = 1, (Z_{ij} = 1 \vee Z_{ji} = 1) \Rightarrow \; B_j \geqslant B_i + p_i + \delta \qquad \forall (i,j,k_1,k_2) \in \Phi^{MTS-MTS} \tag{15}$$

$$Y_i^{k_1} = 1, Y_j^{k_2} = 1, (Z_{ij} = 1 \vee Z_{ji} = 1) \Rightarrow \; B_j \geqslant B_i + p_i \qquad \forall (i,j,k_1,k_2) \in \Phi^{SC-SC} \cup \Phi^{SC-MTS} \tag{16}$$

$$X_{ij}^k \in \{0,1\} \quad \forall (i,j) \in A, k \in K_{ij} \tag{17}$$

$$Z_{ij} \in \{0,1\} \quad \forall (i,j) \in \Psi \tag{18}$$

$$Y_i^k \in \{0,1\} \quad \forall i \in P \cup D, k \in K_i \tag{19}$$

$$B_i, W_i \geqslant 0 \quad \forall i \in P \cup D \tag{20}$$

The overall objective (1) consists in finding the vehicle schedule that minimizes both vehicle travel times and waiting times. Constraints (2) state that every *direct* or *indirect* housekeeping task has to be served exactly once by a transfer vehicle (i.e. a H-SC in the first case and an MTS in the latter). Equalities (3) and (4) guarantee that every transfer vehicle starts at the depot and returns to the depot at the end of its operations. Constraints (5) ensure the conservation of the flows. Constraints (6) guarantee that a vehicle moves from the task source point to the planned destination point. Constraints (7) compute the assignment variables Y. Constraints (8) eliminate subtours and compute the completion time of the operations on a vertex. Constraints (9) and (10) guarantee that the tasks are handled within the desired time window. Constraints (11) guarantee that the first task of a vehicle is not started before the due time to reposition from the depot. Constraints (12) ensure that the vehicle is repositioned at the depot within the time horizon. Constraints (13) compute the waiting time. Constraints (14)–(16) ensure that there is no interference between the operations of the transfer vehicles. Finally, (17)–(20) define the domains of the decision variables. Note that non-linear constraints, given in (8) and (11)–(16), can be linearized using big-M constants.

4. Simulation-based Optimization Procedure

Model P^H could be solved by resorting to a relaxed formulation where the interference constraints that complicate the above VSP-based formulation are added dynamically within a branch-and-cut algorithm. However, such an approach would prevent a good approximation of the waiting times arising in the housekeeping process. Indeed, some of the input parameters are not known with certainty due to the stochastic nature of the duration of single activities as well as to queuing phenomena, such as those mentioned in the step-by-step description given in Figure 2.

Hence, starting from Figure 2, our choice was to design a discrete-event based simulation model capable of capturing all the stochastic features of the real framework. To fully exploit the benefits of the discrete-event simulation approach that, when used as a stand-alone tool, is just aimed at system analysis, we resort to an SO approach and develop a supporting tool for decision making. Nevertheless, the formulation (1)–(20) is useful in providing a formal and compact description of the sequence of activities and decisions that the simulation must reproduce in a more realistic stochastic environment.

Within our SO approach, we estimate the waiting times via multiple replications of discrete-event simulation, while the travel times are assumed to be fixed at their deterministic value observed in common practice where the vehicle's speed is unaffected by intra-terminal traffic conditions [23]. Thus, the objective function for P^H is rewritten as follows:

$$\textit{minimize} \qquad \sum_{(i,j)\in A_2} \sum_{k\in K_{ij}} t_{ij} \cdot X_{ij}^k + \sum_{i\in P\cup D} E\left[W_i\right] \tag{21}$$

where, as usual, $E[.]$ stands for the expected value to be estimated by the corresponding sample mean. For later convenience, the first and second terms in (21) are referred to as TT and WT as defined in the following:

$$TT = \sum_{(i,j)\in A_2} \sum_{k\in K_{ij}} t_{ij} \cdot X_{ij}^k \tag{22}$$

$$WT = \sum_{i\in P\cup D} E\left[W_i\right] \tag{23}$$

The feasibility of a solution of P^H is assessed via simulation and the problem constraints are captured in the proper event-driven constructs of the implemented simulation model. In the case at hand, any given solution is declared feasible after it has been evaluated by simulation if all its scheduled tasks can be completed within the time horizon.

Bearing this in mind, the SO approach can be tailored to the housekeeping problem. The SO framework conceived in this case is illustrated in Figure 3. Some differences between this approach and the classical SO scheme [21] deserve

some comments. The main feature of the iterative scheme proposed here aims to circumvent the difficulty arising when applying SO to combinatorial optimization problems: the number of solutions to be simulated. Since simulation is extremely time consuming, ranking solutions by means of a quantitative index allows us to steer solution evaluation via simulation only towards the most promising solutions.

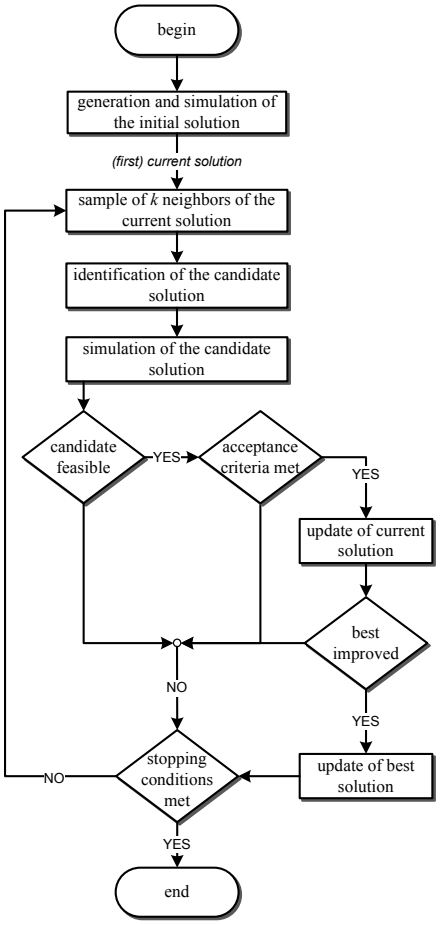


Figure 3: Scheme of the implemented SO procedure.

The scheme can be adapted to many well-known algorithms and, in particular, to metaheuristics. The algorithm first generates a feasible initial solution (or *first current solution*) whose objective function is estimated by simulation. Then, the whole neighborhood or a limited sample from the neighborhood of the current feasible solution is generated by applying operators (a.k.a. *moves*). Thus, these generated neighbors are ranked according to the goodness of the solution, observing that the impact of a specific move on the current solution can be easily evaluated with respect to the deterministic terms of the objective function (21). The best neighbor is simulated: if the solution is feasible then it is a potential candidate to become the next current solution, otherwise the stopping conditions are checked. The potential candidate solution becomes the new current solution if and only if some predefined acceptance criteria are met (these criteria depend on the particular type of search algorithm). Whether or not the current solution is updated, a new iteration is executed if and only if the stopping criteria are not met.

Further details on the main stages of the SO scheme are given below.

4.1. Simulation of a solution of P^H

The simulation module of the SO framework is developed according to a *process interaction worldview* [30] centered around *resources* (yard storage blocks) and *model objects* (transfer vehicles). Each model object undergoes a

process, i.e. a sequence of activities (e.g. “container retrieval”, “container transfer” and “container storage”) in which every single activity is triggered and completed by specific events (e.g. “start container pick-up”, “end container pick-up”). For sake of completeness, a description of the processes is provided in the appendix. Here, further details on both the simulation and hard-coded constructs can be disregarded and one may picture the simulation module as a black box which, for a given input and scenario setting, provides as output an estimate of performance measures for a solution of P^H (see Figure 4).

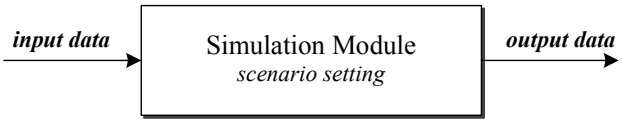


Figure 4: The black-box logic of the simulator.

The scenario setting is defined by:

- time horizon;
- berth length;
- yard layout and composition;
- number of available units per vehicle type (i.e. H-SCs and MTSs);
- location of vehicle depot.

The input data is given by:

- task list per vehicle type;
- container discharge/loading flow interfering with housekeeping operations (modeled by a *Poisson* distribution);
- container pick-up/set-down time (modeled by an *exponential* distribution);
- container shifting time (i.e. the time required to move other containers if the target container is not on top of the stack; this is modeled by a *triangular* distribution);
- travel speed per vehicle type (*constant*).

The output is estimated by confidence intervals generated on multiple simulation runs for the following performance measures:

- waiting time per vehicle type;
- number of completed tasks per vehicle type;
- distance covered per vehicle type.

4.2. Initial solution

The initial solution is generated by using an algorithm based on the company’s practice in the maritime container terminal of pure transshipment located in the port of Gioia Tauro, Italy. According to this practice, the containers requiring housekeeping must be processed before the end of the shift in which vessel discharge/loading is planned. To do so, the terminal planners adopt a greedy assignment policy: first they estimate the initial number of transfer vehicles (i.e. H-SCs and MTSs) and then they assign by hand the tasks to the transfer vehicles according to the distances to be covered and the estimated processing times. If the sum of these processing times goes beyond the available time horizon (a shift), the planners add an extra vehicle to the schedule under execution and repeat the assignment procedure.

The processing time p_i of task i is obtained by considering container pick-up, set-down, shifting and transfer times. Observe that the planners usually inflate the processing time by a percentage factor in order to account for waiting times experienced within a shift. However, the above factor is just a rough estimate, since the impact of congestion phenomena on the housekeeping process mainly depends on the schedule and the number of containers transferred to/from the berth.

Task assignment is based on a constructive approach according to which tasks are assigned to vehicles until vehicle capacity is fully utilized under the given time horizon. Specifically, if task i has been assigned to a vehicle (whether it be an H-SC or MTS), the next task to be processed, say j, will always be the one with the smallest t_{ij}, i.e. the time/cost required to move the transfer vehicle from the last position covered during the execution of task i to the first position to be covered to execute task j. If a task cannot be assigned to a vehicle, the schedule is incomplete. Hence, to complete the schedule, another vehicle is added to the pool of available transfer resources and the assignment procedure is reset and rerun under a smaller time horizon. As a result, a better load balancing on the equipment is also expected.

Obviously, in the initialization stage of the SO framework the initial solution is also the best solution.

4.3. Neighborhood generation

Neighbors are generated from the current solution by using two kinds of local search moves: *i*) *swap* and *ii*) *insertion*. Recalling that two separate couples of tasks and vehicles (i.e. *container–H-SC* and *batch of containers–MTS*) are defined in P^H, we observe that the moves defined in the following can be applied independently to either of the above couples.

In order to describe these moves, let us suppose for P^H that:

- i and j are two tasks;
- $v_i^{(b)}$ and $v_j^{(b)}$ are, respectively, the two vehicles to which tasks i and j are assigned *before* applying a move;
- $v_i^{(a)}$ and $v_j^{(a)}$ are, respectively, the two vehicles to which tasks i and j are assigned *after* applying a move;
- $s_i^{(b)}$, $s_j^{(b)}$, $s_i^{(a)}$ and $s_j^{(a)}$ are respectively the positions (i.e., sequence number in the vehicle schedule) of the tasks before and after applying a move.

Two different swap moves are embedded in the framework:

Swap1 Tasks i and j are assigned to the same transfer vehicle, i.e. $v_i^{(b)} = v_j^{(b)}$. Once the move is applied, $s_i^{(a)} = s_j^{(b)}$ and $s_j^{(a)} = s_i^{(b)}$.

Swap2 Tasks i and j are assigned to two different transfer vehicles of the same type (i.e. H-SC or MTS), i.e. $v_i \neq v_j$. This move provides for swapping $v_i^{(a)} = v_j^{(b)}$ and $v_j^{(a)} = v_i^{(b)}$ and the related positions $s_i^{(a)} = s_j^{(b)}$ and $s_j^{(a)} = s_i^{(b)}$.

There are also two different types of insertion moves:

Insertion1 A task i remains assigned to the same vehicle $v_i^{(b)} = v_i^{(a)}$, but it is moved from its current position to a different position, i.e. $s_i^{(b)} \neq s_i^{(a)}$.

Insertion2 A task i is moved from the current vehicle to which it is assigned to any other vehicle and in any other position, i.e. $v_i^{(b)} \neq v_i^{(a)}$, under the assumption that $v_i^{(b)}$ and $v_i^{(a)}$ are of the same type (i.e. H-SC or MTS).

Given a solution for P^H, according to the above moves, the size of the solution's neighborhood is

$$\Big[n_1 \cdot (n_1 - 1) + n_1 \cdot (n_1 + |K^{SC}| - 2)\Big] + \Big[n_2 \cdot (n_2 - 1) + n_2 \cdot (n_2 + |K^{MTS}| - 2)\Big], \tag{24}$$

where the terms in the two square brackets correspond, respectively, to the number of swap and insertion moves.

4.4. Identification of the candidate solution

The solutions generated within the neighborhood of the current solution are ranked with respect to a measurable index: vehicle travel time. Given a feasible schedule, the deterministic part of objective function (21) is immediately evaluated and this value represents the goodness of the solution. Among all neighbors, the one bearing the smallest goodness of the solution will be chosen as candidate solution. Of course, if only one neighbor is generated, it will automatically become the candidate solution.

4.5. Stopping criteria

Stopping criteria are often based on the comparison between the current best solution and a lower bound. Unfortunately, in our case a lower bound drawn from the IP formulation P^H is useless since it does not account for the congestion phenomena featured in the housekeeping process. Thus, here the stopping criteria are defined according to a given time budget and/or the number of iterations to be performed.

4.6. Algorithms in the framework

For the SO framework proposed in this paper, we decided to consider two well-known metaheuristic paradigms: *simulated annealing* (SA) [31] and *tabu search* (TS) [32]. The rationale of this choice is to investigate the role played by neighborhood generation and solution ranking.

In the SA-based framework, neighborhood generation is simplified: at each iteration, SA generates only one solution from the neighborhood of the current solution. This single solution is then simulated and, if feasible, it becomes a candidate solution. The candidate solution is surely accepted as new current solution if it improves the current solution; otherwise, it is accepted with a probability of acceptance p. At iteration h, $p = e^{-\Delta f/T_h}$, where Δf is the difference between the value of the current solution and the value of the candidate solution, and T_h is the temperature at iteration h. The temperature is updated at iteration h according to a cooling scheme based on a cooling rate α. Therefore, $T_{h+1} = T_h \cdot \alpha$. The cooling rate α and the initial temperature T_0 were chosen according to initial computational experiments.

As for the TS-based framework, neighborhood generation returns multiple candidate solutions from which the best is selected after a proper ranking and then simulated. If feasible, it becomes the new current solution. Observe that during neighborhood generation, any kind of move may be applied except for those contained in the tabu list. The tabu list is updated with those moves that, if reversed, can lead the search process towards solutions recently visited. The swap moves are stored in the triplets $(i, v_i^{(b)}, s_i^{(b)})$ and $(j, v_j^{(b)}, s_j^{(b)})$, while the insertion moves are stored in 5-tuples $(i, v_i^{(b)}, s_i^{(b)}, v_i^{(a)}, s_i^{(a)})$.

5. Numerical Experiments

The efficiency of the SO framework devised to support the planning of housekeeping operations has been tested in a real-life context. Comparisons between SA and TS have been carried out by considering 50 instances divided into 5 groups (A–E). In each group: the number of housekeeping tasks to be performed is generated according to the number of vehicles and to a uniform distribution ranging in $[50, 55]$ for each H-SCs and in $[6, 8]$ for each MTSs (e.g., given an instance with two MTSs, the number of tasks for the MTSs is the sum of two outcomes from $U(6, 8)$); the number of transfer vehicles remains constant within each specific group of instances (the number of tasks and vehicles are obviously correlated). The size of the neighborhood of a solution to P^H (see Section 4.3 for the moves used in neighborhood generation) depends on the number of tasks and vehicles in the problem and it is given by eq. (24). For each group of instances, the above information is provided in Table 2.

As previously specified, experiments are based on the real-life maritime container terminal located in Gioia Tauro (Italy). Therefore, all scenarios feature a 3.1km berth, vessel loading/discharge operations based on distributions fitted using historical data and a yard divided into 80 blocks which, in turn, are organized in four parallel areas located behind the berth. The first area is the closest to the berth and it is used to stack the so-called A-type containers, i.e. 1-TEU operative (non-empty) containers; the second area is mostly dedicated to B-type containers, i.e. 2-TEU operative containers; the third area is normally smaller than the others and it is used to store C-type containers, i.e. refrigerator containers or *reefers*, whatever be their size; the last area is the most distant from the berth and it is dedicated to the

n° of groups: 5 — n° of instances per group: 10					
Instance set	*A*	*B*	*C*	*D*	*E*
n° of H-SCs	4	8	12	16	20
n° of MTSs	2	3	4	5	6
Avg. n° of tasks for H-SCs	102	204	307	408	510
Avg. n° of tasks for MTSs	7	15	22	30	38
Avg. neighborhood size for H-SCs	21,075	84,267	191,028	338,568	528,059
Avg. neighborhood size for MTSs	84	435	995	1,879	2,980
Time budget per instance (min)	2	4	6	8	10

Table 2: Description of the instance sets.

storage of empty containers. Every instance refers to a 6-hour planning horizon ($T = 6$), which is also the duration of a work shift in the terminal. For each task i, $r_i = 0$ and $d_i = T$ have been set.

Comparisons between the two algorithms have been carried under the same execution time for all the instances belonging to the same group. Furthermore, the execution time is set to a value proportional to the average number of tasks to be carried for that group. As reported in Table 2, it ranges from 2 minutes for instances in group A to 10 minutes for instances in group E. To account for the non-deterministic effects related to the execution of both algorithms under comparison, the solution of each instance of whatever group has been repeated ten times, thus providing average results with associated standard deviations. Furthermore, for each run of the two algorithms on whatever instance, the required estimation via simulation of the objective function value has been carried out through a sample of fifteen replications. According to preliminary numerical experience, the above sample size should guarantee a high probability of avoiding to update the current solution with a worse neighbor solution due to random deviations of the sample estimates from the expected values.

As for algorithm details, in both the TS and SA, neighborhood generation occurs by applying an insertion move or a swap move to the current solution. In the TS algorithm: the tabu tenure is set equal to 10; at each iteration, 10% is sampled from the current solution's neighborhood; the tabu list is implemented by keeping track of the moves that have led the search process to the previous solutions. In the SA algorithm, the initial temperature is set equal to 100; the cooling rate depends on the time budget set for each group of instances (see Table 2) and it is .9934, .9969, .9980, .9985, and .9990 for instances of group A, B, C, D and E, respectively; according to the classic SA scheme, at each iteration only one neighbor is generated from the current solution.

The framework has been implemented in Java 7 and all experiments have been carried out on a 3.5 GHz Intel i7-3770K desktop computer with 8Gb of RAM.

5.1. Comparison

The numerical results returned by the experiments performed with the TS and SA algorithms on the instances from groups A to E are reported in Tables 3 through 7. In particular, for any given instance of a specific group, a table contains: the value of the total travel time (TT), the total waiting time (WT) and the sum of the two ($TT + WT$) for the initial solution and the best solution returned by the TS and SA algorithms, respectively. The decomposition of the cumulative value of the objective function (21) into the two terms TT and WT, respectively defined in eqs. (22) and (23), allows the reader to appreciate the separate contribution of the solution algorithms to the two quantities.

For convenience, the average results for the instances of each group are listed in Table 8. Observe that the significant reduction of the waiting time achieved throughout the group of instances can be seen as the result of the integration of simulation within both the SA and TS algorithms.

According to the results reported in Tables 3 through 7, the major conclusion to draw is that, under the same time budget, the TS algorithm is more effective than the SA algorithm. In particular, the average improvement for the instances of group A is 5.6% returned by the SA and 26.5% by the TS; for the instances of group B 9.9% by the SA and 28.3% by the TS; for the instances of group C 4.7% by the SA and 21.9% by the TS; for the instances of group D 5.9% by the SA and 21.7% by the TS; and for the instances of group E 6.9% by the SA and 21.7% by the TS. In summary, the average improvement of the initial solution over all the instances is 5.9% for the SA and 22.3% for the TS. In

particular, the significant improvement achieved by the TS is on both of the components of the objective function (i.e. TT and WT), whereas the SA only improves the WT component at the cost of sacrificing the TT component. This is actually due to the fact that the TS takes benefit of the neighborhood search process, where the neighbors are screened according to their deterministic TT value and only after selected on the basis of the WT value. Vice versa, in the SA-guided search, no preliminary screening is performed on the TT value and, thus, solutions are chosen according to the cumulative value of the objective function, whatever be the contribution of the single components.

A further explanation for the significant difference in the quality of the solutions returned by the two algorithms lies in the average number of infeasible solutions explored during the corresponding search processes: 2900 per instance for the SA against 1250 per instance for the TS. This is due to the fact that in the SA algorithm: *i*) the candidate solution is randomly chosen from a large neighborhood, rather than from a sample of the neighborhood; *ii*) the search process in the initial stage is likely to accept solutions that do not improve the value of the objective function and that are far from the initial solution. Therefore, the probability of selecting unpromising candidate solutions and, thus, searching in a portion of the search space where solutions are barely feasible increases. On the other hand, although the TS algorithm bases its search process on a small sample of the neighborhood (i.e. 10%), it allows to pre-screen promising candidate solutions and, thus, move slowly, but effectively towards more profitable sub-regions.

The above explanation is even clearer in Figures 5 through 9. These graphs compare, second-by-second, the average trends of the SA and TS algorithms with respect to the average value of the objective function of the current best solution for each instance in groups A through E. Both algorithms improve in the initial stage of the search process when they stay in the vicinity of the initial solution. Observing Figures 5-9, one may recognize that the SA realizes almost all the improvements at the very beginning of the search process. Vice-versa, although a steeper descent may be observed at the initial stage of the search process, the improvements achieved by the TS algorithm are spread over the whole search process.

	Value of initial solution			AVG (STD) value of best SA solution			AVG (STD) value of best TS solution		
Instance	*TT*	*WT*	*TT+WT*	*TT*	*WT*	*TT+WT*	*TT*	*WT*	*TT+WT*
A1	103	122	224	109 (8.9)	90 (0.3)	198 (1.7)	93 (3.0)	76 (3.2)	169 (31.6)
A2	88	120	207	98 (6.3)	92 (14.1)	190 (14.1)	76 (4.5)	73 (4.5)	149 (54.8)
A3	107	117	224	116 (20.0)	94 (14.1)	211 (14.1)	95 (5.5)	76 (5.5)	171 (30.0)
A4	112	104	216	113 (14.1)	98 (10.0)	211 (14.1)	88 (1.7)	73 (5.5)	161 (31.6)
A5	115	112	226	122 (24.5)	100 (8.4)	223 (24.5)	100 (1.4)	74 (6.3)	175 (26.5)
A6	79	115	193	86 (6.3)	101 (10.0)	187 (14.1)	65 (4.5)	68 (5.5)	134 (31.6)
A7	113	117	230	118 (20.0)	95 (10.0)	213 (17.3)	99 (2.0)	69 (4.5)	168 (31.6)
A8	81	113	194	87 (7.1)	101 (14.1)	187 (20.0)	70 (2.0)	70 (4.5)	140 (44.7)
A9	113	117	230	120 (22.4)	100 (17.3)	220 (22.4)	99 (1.4)	68 (4.5)	167 (31.6)
A10	89	115	204	102 (9.5)	92 (14.1)	194 (14.1)	78 (2.0)	64 (4.5)	143 (44.7)

Table 3: Computational results for instance set A.

5.2. Test on the moves

To complete the analysis of the heuristic search processes, we have investigated the effect of the various types of moves which have been implemented within the SA and TS algorithms for each group of instances. By measuring the above effect in terms of *i*) percentage of infeasible solutions returned during their application, *ii*) percentage of solutions that are accepted as the new best and, finally, *iii*) percentage of improvement achieved by the objective function, we have reported in Table 10 numerical results for both the insertion and swap moves under the two types of transfer vehicles. Some considerations follows here.

The percentage of infeasibility increases with the size of the instances. In particular: the *insertion2* type moves are those which suffer the largest growth, with a worst case value of about 86%; the *swap2* type moves are those where the growth is limited within small values for both transfer types (max 12%); the *insertion1* type move for the H-SC transfer vehicles almost always provides feasible solutions, despite the increasing size of the instances.

	Value of initial solution			AVG (STD) value of best SA solution			AVG (STD) value of best TS solution		
Instance	*TT*	*WT*	*TT+WT*	*TT*	*WT*	*TT+WT*	*TT*	*WT*	*TT+WT*
B1	164	246	410	173 (28.3)	229 (31.6)	402 (14.1)	138 (31.6)	177 (24.5)	315 (17.3)
B2	175	248	423	179 (22.4)	232 (44.7)	410 (22.4)	142 (44.7)	180 (22.4)	322 (24.5)
B3	188	270	457	199 (14.1)	247 (54.8)	446 (54.8)	162 (44.7)	181 (22.4)	343 (44.7)
B4	191	264	455	211 (31.6)	235 (44.7)	446 (44.7)	167 (31.6)	179 (22.4)	346 (54.8)
B5	198	260	459	214 (20)	214 (24.5)	428 (30)	172 (28.3)	183 (20)	355 (44.7)
B6	198	280	478	214 (20)	234 (44.7)	448 (54.8)	166 (31.6)	184 (22.4)	351 (54.8)
B7	198	263	462	220 (26.5)	224 (30)	444 (44.7)	176 (26.5)	187 (17.3)	363 (44.7)
B8	224	277	501	242 (44.7)	220 (26.5)	462 (44.7)	197 (5.5)	189 (14.1)	386 (17.3)
B9	230	254	484	256 (54.8)	201 (17.3)	457 (44.7)	187 (14.1)	182 (22.4)	369 (31.6)
B10	163	278	441	181 (24.5)	235 (44.7)	416 (24.5)	131 (31.6)	185 (17.3)	317 (20)

Table 4: Computational results for instance set *B*.

	Value of initial solution			AVG (STD) value of best SA solution			AVG (STD) value of best TS solution		
Instance	*TT*	*WT*	*TT+WT*	*TT*	*WT*	*TT+WT*	*TT*	*WT*	*TT+WT*
C1	327	430	757	353 (54.8)	375 (44.7)	728 (44.7)	273 (30)	319 (26.5)	592 (17.3)
C2	272	520	792	304 (28.3)	440 (54.8)	744 (54.8)	238 (44.7)	324 (31.6)	562 (44.7)
C3	263	416	680	284 (22.4)	372 (44.7)	655 (54.8)	216 (17.3)	320 (28.3)	537 (44.7)
C4	243	410	653	274 (44.7)	353 (54.8)	627 (31.6)	218 (20)	289 (22.4)	507 (20)
C5	268	466	734	294 (20)	397 (44.7)	691 (30)	231 (31.6)	324 (28.3)	555 (54.8)
C6	271	418	689	311 (28.3)	322 (44.7)	633 (44.7)	234 (31.6)	310 (14.1)	545 (44.7)
C7	284	422	706	305 (17.3)	366 (54.8)	670 (44.7)	239 (44.7)	318 (22.4)	556 (44.7)
C8	277	410	687	295 (20)	375 (44.7)	670 (31.6)	237 (44.7)	302 (28.3)	539 (44.7)
C9	327	425	752	348 (54.8)	369 (44.7)	717 (24.5)	267 (31.6)	337 (44.7)	605 (20)
C10	284	424	708	310 (20)	376 (31.6)	687 (26.5)	253 (54.8)	343 (44.7)	596 (14.1)

Table 5: Computational results for instance set *C*.

	Value of initial solution			AVG (STD) value of best SA solution			AVG (STD) value of best TS solution		
Instance	*TT*	*WT*	*TT+WT*	*TT*	*WT*	*TT+WT*	*TT*	*WT*	*TT+WT*
D1	369	654	1023	394 (20)	552 (70.7)	946 (70.7)	311 (14.1)	485 (24.5)	797 (17.3)
D2	376	776	1152	418 (30)	614 (44.7)	1032 (44.7)	322 (24.5)	550 (63.2)	872 (44.7)
D3	372	578	950	388 (20)	540 (54.8)	928 (44.7)	295 (5.5)	464 (44.7)	760 (44.7)
D4	358	614	973	399 (22.4)	502 (31.6)	901 (20)	296 (14.1)	426 (31.6)	722 (28.3)
D5	360	608	968	394 (28.3)	557 (63.2)	951 (54.8)	293 (10)	477 (30)	770 (31.6)
D6	353	620	973	377 (31.6)	537 (44.7)	915 (31.6)	308 (10)	468 (54.8)	775 (44.7)
D7	354	649	1003	385 (17.3)	567 (44.7)	952 (54.8)	308 (14.1)	487 (28.3)	795 (24.5)
D8	425	613	1039	453 (54.8)	542 (63.2)	996 (31.6)	356 (44.7)	468 (44.7)	824 (31.6)
D9	332	671	1003	382 (24.5)	547 (54.8)	928 (44.7)	281 (20)	496 (26.5)	777 (31.6)
D10	475	682	1157	514 (30)	575 (54.8)	1089 (94.9)	425 (26.5)	503 (31.6)	928 (44.7)

Table 6: Computational results for instance set *D*.

	Value of initial solution			AVG (STD) value of best SA solution			AVG (STD) value of best TS solution		
Instance	*TT*	*WT*	*TT+WT*	*TT*	*WT*	*TT+WT*	*TT*	*WT*	*TT+WT*
E1	442	1041	1484	519 (54.8)	816 (77.5)	1335 (316.2)	383 (20)	709 (54.8)	1092 (100)
E2	459	855	1315	491 (22.4)	767 (44.7)	1258 (282.8)	399 (9.5)	654 (70.7)	1053 (70.7)
E3	508	914	1422	534 (44.7)	816 (70.7)	1351 (316.2)	435 (31.6)	643 (63.2)	1077 (83.7)
E4	448	950	1399	489 (22.4)	779 (54.8)	1268 (282.8)	389 (14.1)	700 (14.1)	1089 (94.9)
E5	510	908	1418	616 (94.9)	739 (141.4)	1355 (316.2)	426 (28.3)	728 (44.7)	1154 (173.2)
E6	439	1026	1465	529 (70.7)	811 (54.8)	1340 (316.2)	367 (31.6)	709 (31.6)	1075 (83.7)
E7	522	851	1372	559 (44.7)	707 (44.7)	1265 (282.8)	455 (44.7)	634 (44.7)	1089 (94.9)
E8	541	906	1446	600 (31.6)	785 (54.8)	1386 (447.2)	477 (26.5)	712 (31.6)	1189 (200)
E9	521	889	1410	558 (54.8)	773 (63.2)	1331 (316.2)	446 (44.7)	689 (44.7)	1135 (141.4)
E10	398	884	1282	441 (44.7)	709 (54.8)	1149 (173.2)	354 (44.7)	659 (63.2)	1013 (44.7)

Table 7: Computational results for instance set *E*.

	AVG (STD) value of initial solution			AVG (STD) value of best SA solution			AVG (STD) value of best TS solution		
Instance	*TT*	*WT*	*TT+WT*	*TT*	*WT*	*TT+WT*	*TT*	*WT*	*TT+WT*
A	100 (14.3)	115 (4.9)	215 (14.3)	107 (13.3)	96 (4.2)	203 (13.8)	86 (13)	71 (3.9)	158 (14.8)
B	205 (45.7)	279 (51.4)	484 (94.1)	209 (27)	227 (12.9)	436 (20.6)	164 (21.3)	183 (3.7)	347 (23.3)
C	282 (26.6)	434 (34.1)	716 (42.3)	308 (25.3)	375 (30)	682 (39.1)	241 (18.8)	319 (15.7)	559 (30.5)
D	377 (41.9)	647 (55.5)	1024 (73.8)	410 (42.6)	553 (29.1)	964 (58.6)	320 (42.4)	482 (31.9)	802 (59.5)
E	479 (47.3)	922 (65.3)	1401 (63.4)	534 (52.8)	770 (40.7)	1304 (69.5)	413 (40.6)	684 (33.3)	1097 (50.8)
avg	*289*	*479*	*768*	*314*	*404*	*718*	*245*	*348*	*593*

Table 8: Average computational results for instance sets *A*–*E*.

	AVG (STD) number of evaluated solutions		AVG (STD) percentage of infeasible solutions	
Instance	SA	TS	SA	TS
A	2029 (80.7)	1121 (70.6)	17 (0.5)	2 (0.7)
B	2924 (137.8)	1792 (110.2)	32 (1.3)	11 (0.5)
C	3960 (174.3)	2725 (203.8)	51 (1.9)	48 (1.2)
D	5676 (111.2)	3076 (132.0)	70 (3.2)	63 (2.0)
E	6920 (132.0)	3916 (221.8)	74 (3.5)	69 (1.5)

Table 9: Infeasible solutions results for instance sets *A*–*E*.

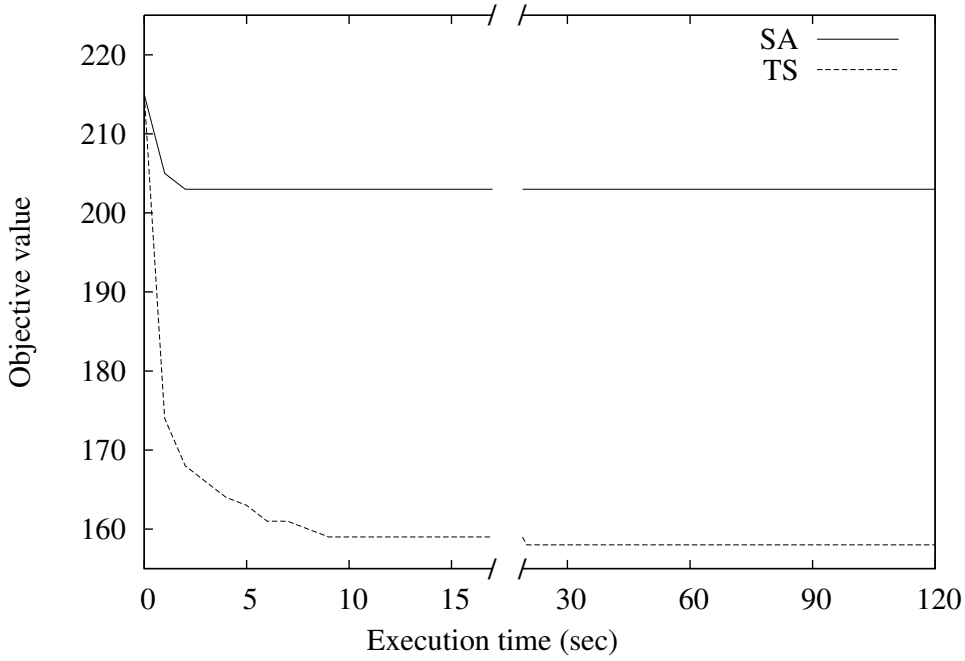


Figure 5: Average trend of SA vs average trend of TS for instances of group *A*.

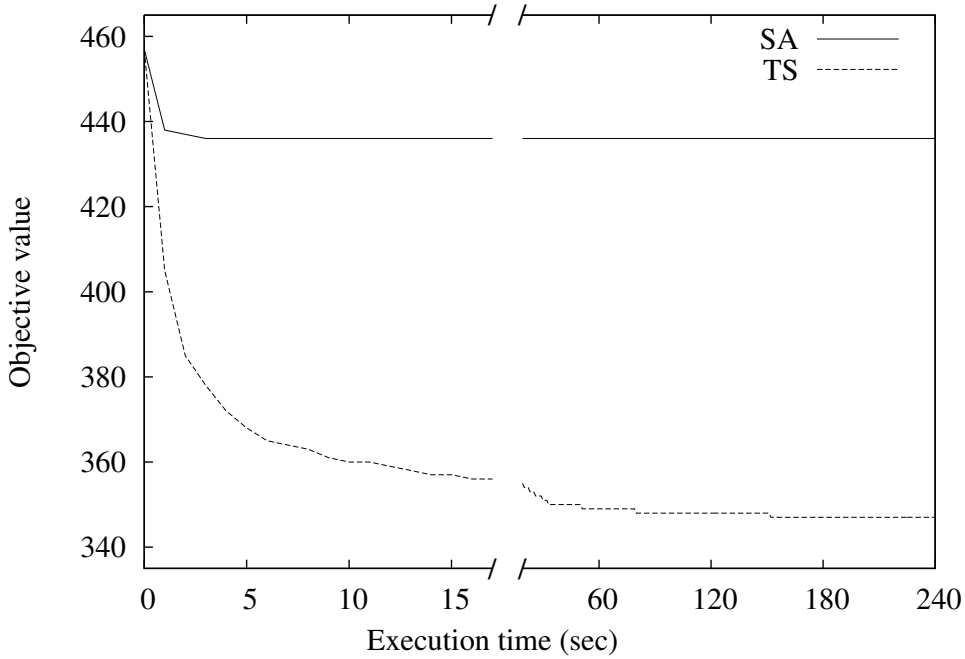


Figure 6: Average trend of SA vs average trend of TS for instances of group *B*.

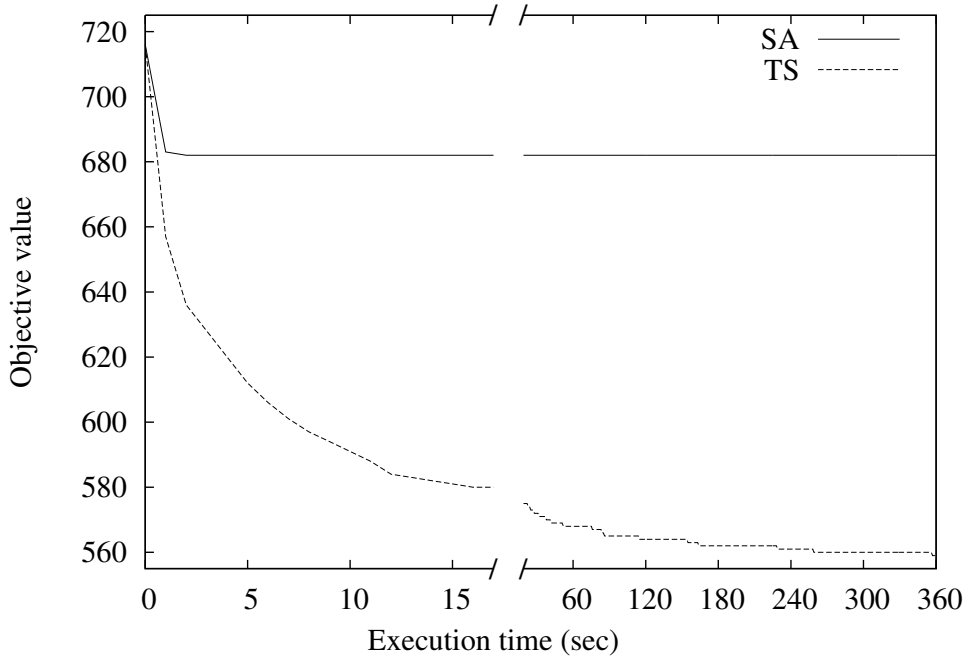


Figure 7: Average trend of SA vs average trend of TS for instances of group *C*.

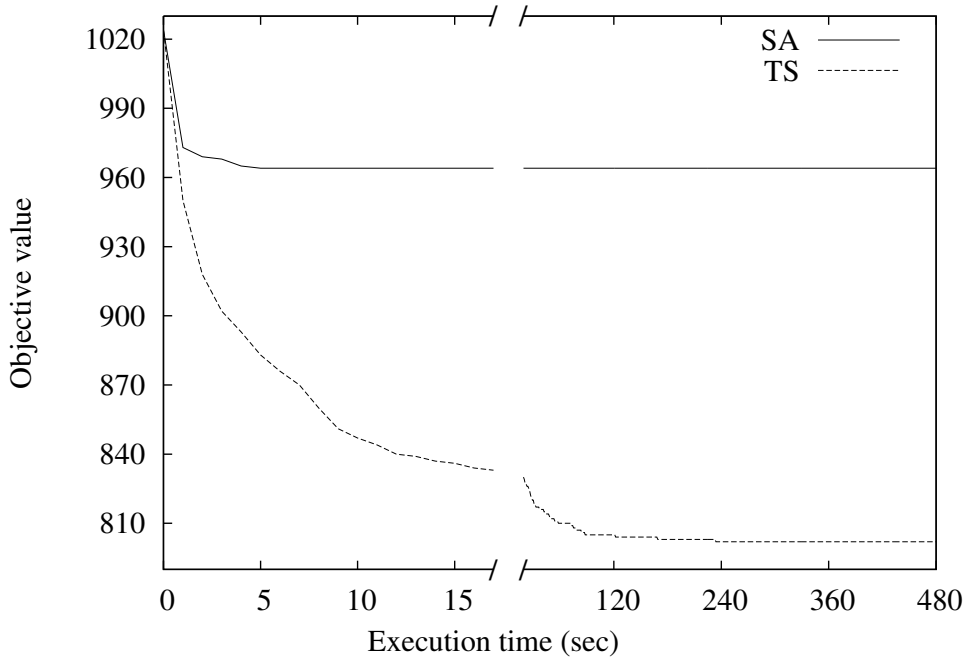


Figure 8: Average trend of SA vs average trend of TS for instances of group *D*.

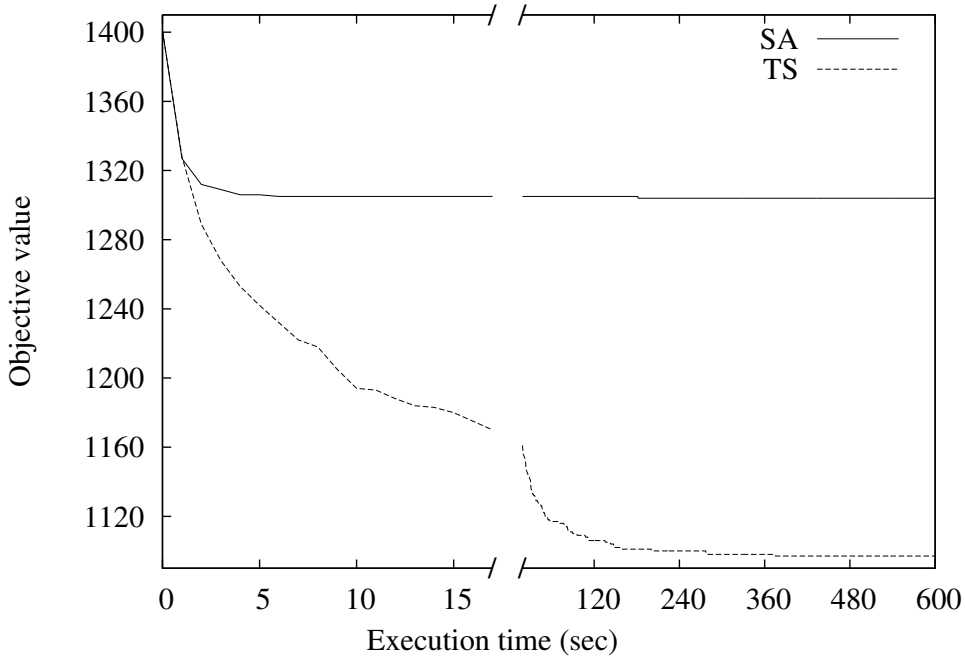


Figure 9: Average trend of SA vs average trend of TS for instances of group *E*.

The percentage of acceptance is generally under a threshold of about 3% and there is no evidence of the superiority of a move type w.r.t. the others. It is worth mentioning that all the move types applied to the MTS schedules register the lowest common value of percentage of acceptance (i.e. 0.2%) for group *A* instances, due to the limited number of both MTS tasks and MTS vehicles.

The percentage of improvement is generally significant, i.e. it is on average 20.4% and 19.3% on instances from *B* to *E*, respectively, for MTS and H-SC transfer vehicles. Moreover, these improvements appear commonly achieved whatever be the move type. The improvement achieved on the MTS schedules for instances belonging to group *A* is very small (below 3%): again, this is explained by the small number of both tasks and vehicles. All the moves for the H-SC transfer vehicles exhibit slightly better results for smaller instances (groups *A* and *B*) and, at a finer look, *insertion2* and *swap2* type moves seem to achieve one more percentage point of improvement on all the group of instances (*A* to *E*).

6. Conclusions

This paper has introduced a simulation-based optimization (SO) approach for the housekeeping problem in a transshipment container terminal with the overall objective of finding the vehicle schedule that minimizes both vehicle travel times and waiting times. The embedded heuristics rely on simple local search moves, but they use a detailed simulation model to evaluate the aforesaid vehicle travel times as well as waiting times due to congestion phenomena that arise in non conflict-free simultaneous vehicle access to shared yard rows in the terminal's container storage area.

Numerical results have shown how the performances of a tabu search heuristic and a simulated annealing heuristic are quite dissimilar: in 98% of the time the former, which features a multiple generation and sampling mechanism, outperforms the latter, which features a single generation and sampling mechanism. This most likely occurs because of the particular problem structure characterized by a very low probability of generating and selecting better neighboring solutions than the current solution. Nevertheless, although the neighborhood of a solution cannot be explored in full, experiments have also shown that sampling as little as 10% of this neighborhood at each iteration within the tabu search procedure yields significant improvements with respect to the solutions produced by the simulated annealing

AVG (STD) percentage of infeasibility						
		A	*B*	*C*	*D*	*E*
MTS	Insertion1	0.0 (0.0)	0.0 (0.0)	0.1 (0.2)	11.3 (1.3)	37.3 (3.2)
	Insertion2	0.0 (0.0)	12.4 (2.5)	61.0 (6.4)	72.5 (5.6)	85.9 (4.1)
	Swap1	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	2.1 (0.3)	6.7 (0.9)
	Swap2	0.0 (0.0)	0.0 (0.0)	0.3 (0.1)	1.4 (0.2)	12.2 (1.1)
SC	Insertion1	0.0 (0.0)	0.0 (0.0)	0.2 (0.0)	3.9 (0.3)	0.0 (0.0)
	Insertion2	4.0 (0.8)	10.1 (1.0)	18.1 (2.1)	24.6 (2.5)	29.2 (3.8)
	Swap1	0.0 (0.0)	0.1 (0.0)	0.0 (0.0)	18.2 (1.4)	37.6 (3.2)
	Swap2	0.0 (0.0)	0.0 (0.0)	0.1 (0.0)	1.4 (0.2)	5.7 (0.9)
AVG (STD) percentage of acceptance						
		A	*B*	*C*	*D*	*E*
MTS	Insertion1	0.2 (0.0)	0.9 (0.0)	1.2 (0.0)	1.5 (0.0)	1.4 (0.1)
	Insertion2	0.2 (0.0)	1.3 (0.0)	1.3 (0.0)	1.2 (0.0)	1.2 (0.0)
	Swap1	0.2 (0.0)	1.6 (0.1)	1.8 (0.1)	2.2 (0.1)	2.2 (0.1)
	Swap2	0.2 (0.0)	1.3 (0.0)	2.1 (0.1)	1.9 (0.1)	1.5 (0.0)
SC	Insertion1	2.1 (0.1)	1.5 (0.0)	2.1 (0.1)	2.2 (0.1)	1.9 (0.0)
	Insertion2	2.2 (0.1)	2.4 (0.1)	2.0 (0.1)	1.6 (0.0)	2.2 (0.1)
	Swap1	2.2 (0.1)	2.1 (0.1)	2.3 (0.1)	1.5 (0.1)	1.7 (0.0)
	Swap2	2.8 (0.2)	1.8 (0.1)	2.0 (0.1)	2.6 (0.1)	2.4 (0.2)
AVG (STD) percentage of improvement						
		A	*B*	*C*	*D*	*E*
MTS	Insertion1	3.1 (0.1)	12.1 (1.0)	13.1 (0.7)	11.9 (0.6)	14.6 (1.1)
	Insertion2	2.8 (0.2)	17.9 (1.1)	15.0 (0.8)	12.0 (0.8)	12.8 (0.9)
	Swap1	2.9 (0.1)	14.2 (0.9)	13.2 (0.8)	14.5 (1.0)	14.9 (0.9)
	Swap2	3.0 (0.1)	15.9 (0.8)	14.9 (0.7)	14.9 (0.9)	14.8 (0.8)
SC	Insertion1	21.9 (0.9)	19.5 (0.9)	19.7 (1.1)	18.5 (1.1)	14.9 (0.9)
	Insertion2	24.6 (1.2)	20.7 (1.2)	18.6 (0.9)	17.7 (0.9)	20.0 (1.1)
	Swap1	23.0 (1.0)	21.0 (1.0)	19.1 (1.0)	19.0 (0.9)	17.9 (1.0)
	Swap2	27.4 (1.3)	22.2 (1.1)	19.5 (1.0)	21.2 (1.3)	19.7 (1.2)

Table 10: Move contribution for instance sets *A*–*E*.

procedure or by applying simple rules such as those used in practice in the real container terminal of reference. As a matter of fact, when these rules are used to produce the initial solution for our meta-heuristics, the improvements achieved by the tabu search procedure throughout all the instances examined range from 20% to 27%.

Future work will be focused on improving the compromise between computational time and solution quality. More powerful local search operators will be designed within the current tabu search heuristic to perform a more thorough exploration of the solution space and further effort will be directed to developing a more sophisticated search paradigm based on a large neighborhood search (LNS) logic according to which different strategies will be investigated to gradually improve the initial solution by alternately destroying and repairing it.

Appendix A. Simulation details

This section provides details on the terminal resources and processes undergone by the model objects of the housekeeping simulation module.

The resources are the terminal *yard* and *berth*. The yard is organized in parallel areas, an area is divided into blocks and a block is organized in rows where each row is defined by a sequence of slots. Both blocks and rows are preceded by queuing spaces where MTSs and H-SCs, respectively, wait their turn to perform housekeeping operations. The berth is located along the same longitudinal direction of the yard. It generates the flow of containers transferred between the quay and the yard by means of SCs that are assigned to cranes performing container discharge/loading operations.

As for processes, there are three types, one for each of the three model objects defined in the simulator: D-SCs, H-SCs and MTSs. The related descriptions are given by the flowcharts provided in the following.

The process for the D-SCs is illustrated in Figure A.10. This process models vessel discharge/loading operations by using a probability distribution to determine both the frequency of container retrieval/delivery operations between the quay and the yard and the slot on the yard in which container pick-up/set-down must occur. As a result of every retrieval/delivery request generated during the fixed time horizon, an instance of the D-SC model object is generated with the purpose of *i*) moving towards the target yard block; *ii*) performing container pick-up/set-down from/in the proper slot; *iii*) returning to the quayside. Once these three steps are performed, the object is disposed.

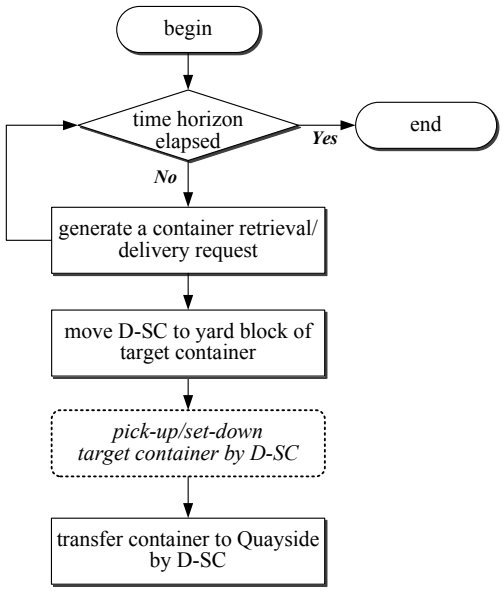


Figure A.10: The process for the quayside model object.

The dotted-line block in Figure A.10 is a macro-activity and it represents the container pick-up/set-down operation in the yard carried out by a generic SC, whether the latter be assigned to container discharge/loading or to container

housekeeping. A zoom on this macro-activity is provided in Figure A.11. Specifically, when an SC reaches the row that hosts the target container, it must first verify if another transfer vehicle is already working in that row or in an adjacent one. If so, the SC must wait its turn until the row becomes available after which it accesses the row to perform the container pick-up/set-down. Once operations are completed, the SC releases the row and leaves the block.

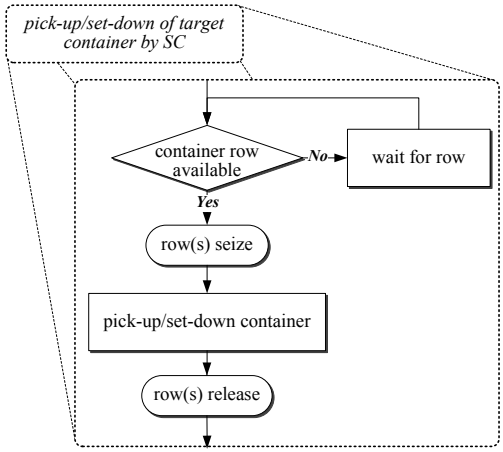


Figure A.11: The pick-up/set-down operations performed by an SC.

The process for the H-SCs is illustrated in Figure A.12. This process starts when the shift starts and ends when the sequence of tasks to be performed (i.e. single containers) is completed. An H-SC first moves from its depot and then, for each task, it *i*) approaches the yard block in which the target container is located; *ii*) picks up the target container; *iii*) transfers the container to the destination block; *iv*) sets down the container in the destination block. Obviously, both container pick-up and set-down macro-activities require seizing the target row as illustrated by Figure A.11. Once the entire schedule is completed the H-SC returns to the assigned depot.

The process for the MTS model object is the most complex since it models the operations of three vehicles (2 M-SCs and a tractor). This process is illustrated in Figure A.13. Similar to the H-SC model object, this process runs during the fixed time horizon until the tasks are finished. The MTS moves from the depot to which it is initially assigned to perform a set of tasks. Each task requires transferring a batch of containers from a source block to a destination block. To do so, the related process first moves an empty multi-trailer, its M-SC for container loading and a tractor near the source block. Then the multi-trailer is loaded with the scheduled containers. Once loading operations are completed, the multi-trailer is hauled by the tractor until the destination block is reached. Here the multi-trailer is unloaded.

The two dotted-line blocks in Figure A.13 are macro-activities for which further details are provided in Figure A.14 and explained below. When an MTS reaches the block in which the batch of containers to be housekept is located, it must first verify whether another MTS has already seized the block for its own purposes. If so, the MTS must queue before the block. As soon as the block is available, the MTS initiates container loading/discharge on/from the multi-trailer. In particular, for each container that belongs to the target batch, if the operation to be performed consists in loading the multi-trailer, then the container is retrieved from the source block by a H-SC and set down on one of the multi-trailer's free wagons; vice versa, if the operation to be performed consists in discharging the multi-trailer, then the container is transferred by a H-SC from the multi-trailer's wagon to the target slot in the row yard. Note that the dotted-line macro-activities in Figure A.14 used to represent the pick-up/set-down operations performed by the H-SCs are carried out according to the steps described in Figure A.11. Once the entire schedule is completed, the MTS returns to the assigned depot.

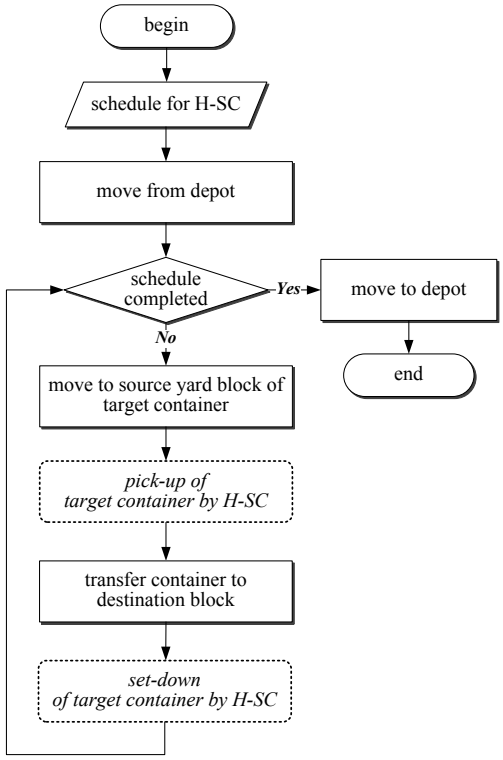


Figure A.12: The process for the H-SC model object.

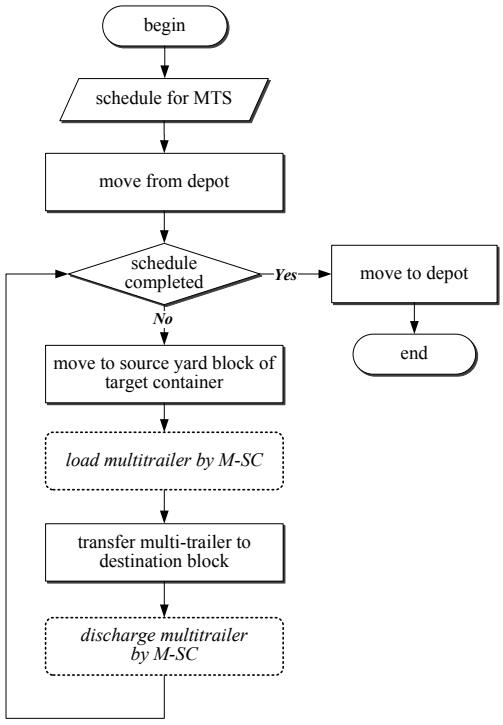


Figure A.13: The process for the MTS model object.

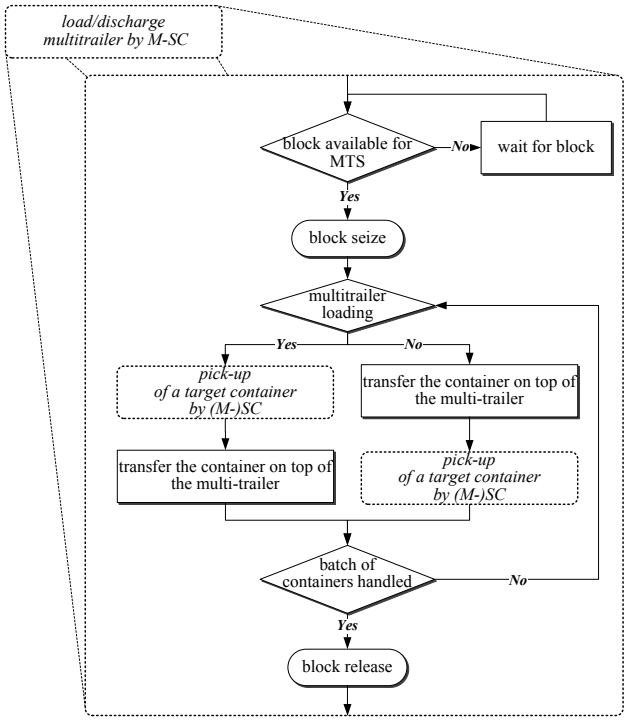


Figure A.14: The loading/discharge operations performed by an MTS.

Acknowledgements

This work was partly supported by the Canadian Natural Sciences and Engineering Research Council under grant 227837-09 and by the Italian National Operational Programme for Research and Competitiveness 2007-2013 under grant PON01_01936. This support is gratefully acknowledged. The authors are also thankful to the anonymous referees for their valuable comments.

References

[1] International Monetary Fund, World Economic Outlook, October 2012: Coping with High Debt and Sluggish Growth, World Economic and Financial Surveys, International Monetary Fund, www.imf.org/external/pubs/ft/weo/2012/02/pdf/text.pdf, 2012.

[2] H. Meersman, E. Van De Voorde, T. Vanelslander, Port congestion and implications to maritime logistics, in: D.-W. Song, P. Panayides (Eds.), Maritime Logistics: Contemporary Issues, Emerald Group Publishing Limited, Bingley, UK, 2012, pp. 49–68.

[3] P. Legato, R. Mazza, R. Trunfio, Simulation for performance evaluation of the housekeeping process, in: C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, A. Uhrmacher (Eds.), Proceedings of the Winter Simulation Conference, Winter Simulation Conference, Berlin (Germany), 2012, pp. 127:1–127:12.

[4] P. Legato, R. Mazza, R. Trunfio, Medcenter container terminal spa uses simulation in operations planning, Interfaces 43 (4) (2013) 313–324.

[5] T. Chen, Yard operations in the container terminal - A study in the "unproductive moves", Maritime Policy & Management: The Flagship Journal of International Shipping and Port Research 26 (1) (1999) 27–38.

[6] Y. Saanen, R. Dekker, Intelligent stacking as way out of congested yards? Part 1, Port Technology International 31 (2007) 80–85.

[7] Y. Saanen, R. Dekker, Intelligent stacking as way out of congested yards? Part 2, Port Technology International 31 (2007) 87–92.

[8] J. Klaws, R. Stahlbock, S. Voß, Container terminal yard operations-simulation of a side-loaded container block served by triple rail mounted gantry cranes, in: J. Böse, H. Hu, C. Jahn, X. Shi, R. Stahlbock, S. Voß (Eds.), Computational Logistics, Vol. 6971 of Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, 2011, pp. 243–255.

[9] L. Lee, E. Chew, K. Tan, H. Huang, W. Lin, H. Y., T. Chan, A simulation study on the uses of shuttle carriers in the container yard, in: S. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. Tew, R. Barton (Eds.), Proceeding of the 2007 Winter Simulation Conference, 2007, pp. 1973–1981.

[10] B. Lee, K. Kim, Comparison and evaluation of various cycle-time models for yard cranes in container terminals, International Journal of Production Economics 126 (2010) 350–360.

[11] M. Petering, Effect of width and storage yard layout on marine container terminal performance, Transportation Research Part E 45 (4) (2009) 591–610.

[12] M. Petering, K. Murty, Effect of block length and yard crane deployment systems on overall performance at a seaport container transshipment terminal, Computers and Operations Research 36 (5) (2009) 1711–1725.
[13] R. Dekker, P. Voogd, E. van Esperen, Advanced methods for container stacking, OR Spectrum 28 (2006) 563–586.
[14] Y. Hirashima, K. Takeda, S. Harada, M. Deng, A. Inoue, A q-learning for group-based plan of container transfer scheduling, The Japan Society of Mechanical Engineers Series C 49 (2) (2006) 473–479.
[15] X. Guo, S. Huang, W. Hsu, M. Low, Yard crane dispatching based on real time data driven simulation for container terminals, in: S. Mason, R. Hill, L. Mnch, O. Rose, T. Jefferson, J. Fowler (Eds.), Proceeding of the 2008 Winter Simulation Conference, 2008, pp. 2648–2655.
[16] X. Guo, S. Huang, W. Hsu, M. Low, A simulation based hybrid algorithm for yard crane dispatching in container terminals, in: M. Rossetti, R. Hill, B. Johansson, A. Dunkin, R. Ingalls (Eds.), Proceeding of the 2009 Winter Simulation Conference, 2009, pp. 2230–2241.
[17] P. Legato, P. Canonaco, R. Mazza, Yard crane management by simulation and optimization, Maritime Economics and Logistics 11 (1) (2009) 36–57.
[18] B. Li, W.-F. Li, Y. Zhang, Y.-H. Ge, H. Chen, X.-L. Liang, Modeling and simulation of yard trailer dispatching at container terminals, in: Proceedings of the 2009 IEEE International Conference on Automation and Logistics, 2009, pp. 29–34.
[19] M. Petering, Y. Wu, W. Li, M. Goh, R. de Souza, Development and simulation analysis of real-time yard crane control systems for seaport container transshipment terminals, OR Spectrum 31 (4) (2009) 801–835.
[20] I. Vis, A comparative analysis of storage and retrieval equipment at a container terminal, International Journal of Production Economics 103 (2006) 680–693.
[21] S. Andradóttir, Simulation optimization, in: J. Banks (Ed.), Handbook of simulation: principles, methodology, advances, applications, and practice, John Wiley & Sons, Inc., Hoboken, NJ, USA, 2007, pp. 307–333.
[22] L.-A. Tuan, R. B. de Kosterb, Y. Yub, Performance evaluation of dynamic scheduling approaches in vehicle-based internal transport systems, International Journal of Production Research 48 (24) (2010) 7219–7242.
[23] B. Skinner, S. Yuan, S. Huang, D. Liu, B. Cai, G. Dissanayake, H. Lau, A. Bott, D. Pagac, Optimisation for job scheduling at automated container terminals using genetic algorithm, Computers & Industrial Engineering doi:10.1016/j.cie.2012.08.012.
[24] J. A. Ferland, P. Michelon, The vehicle scheduling problem with multiple vehicle types, The Journal of the Operational Research Society 39 (6) (1988) 577–583.
[25] A. Bertossi, P. Carraresi, G. Gallo, On some matching problems arising in vehicle scheduling models, Networks 17 (3) (1987) 271–281.
[26] G. Carpaneto, M. Dell'amico, M. Fischetti, P. Toth, A branch and bound algorithm for the multiple depot vehicle scheduling problem, Networks 19 (5) (1989) 531–548.
[27] M. Dell'Amico, M. Fischetti, P. Toth, Heuristic algorithms for the multiple depot vehicle scheduling problem, Management Science 39 (1) (1993) 115–125.
[28] G. Desaulniers, J. Lavigne, F. Soumis, Multi-depot vehicle scheduling problems with time windows and waiting costs, European Journal of Operational Research 111 (3) (1998) 479–494.
[29] A. Hadjar, F. Soumis, Dynamic window reduction for the multiple depot vehicle scheduling problem with time windows, Computers & Operations Research 36 (7) (2009) 2160 – 2172.
[30] J. Banks, J. Carson, B. Nelson, D. Nicol, Discrete-Event System Simulation, 3rd Edition, Prentice-Hall, Upper Saddle River (NJ, USA), 2001.
[31] S. Kirkpatrick, C. D. Gelatt J., M. P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[32] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, Norwell, MA (USA), 1997.