Journal of Intelligent Manufacturing, 12, 521–534, 2001

An optimization methodology for intermodal terminal management

L. M. GAMBARDELLA,* M. MASTROLILLI, A. E. RIZZOLI and M. ZAFFALON

IDSIA—Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Galleria 2, CH-6928 Manno, Switzerland
E-mail: luca@idsia.ch; monaldo@idsia.ch; andrea@idsia.ch; zaffalon@idsia.ch

A solution to the problems of resource allocation and scheduling of loading and unloading operations in a container terminal is presented. The two problems are formulated and solved hierarchically. First, the solution of the resource allocation problem returns, over a number of work shifts, a set of quay cranes used to load and unload containers from the moored ships and the set of yard cranes to store those containers on the yard. Then, a scheduling problem is formulated to compute the loading and unloading lists of containers for each allocated crane. The feasibility of the solution is verified against a detailed, discrete-event based, simulation model of the terminal. The simulation results show that the optimized resource allocation, which reduces the costs by $\frac{1}{3}$, can be effectively adopted in combination with the optimized loading and unloading list. Moreover, the simulation shows that the optimized lists reduce the number of crane conflicts on the yard and the average length of the truck queues in the terminal.

Keywords: Intermodal terminals, flexible job shop, resources allocation, containers optimization

1. Introduction

Intercontinental cargo transport has been continuously increasing since the advent of container shipping in the 1950s. Nowadays 95% of world cargo is moved by ship. Ports play the role of exchange hubs where containers are moved from ships to trains and trucks and their efficiency is fundamental to sustain the mounting cargo traffic. To cope with increased traffic demands and with decreasing profit margins the terminal operators need to improve the management of terminal processes such as ship berthing, ship loading and unloading, straddle crane routing, resource allocation, yard space management.

All of these problems are strictly interrelated; for instance, the choice of the position of the containers on the yard impacts on the decisions made regarding the allocation of resources, the allocation of resources constrains the scheduling of loading and unloading operations, and so on. The complexity of these processes makes it necessary to use efficient methods for the optimization of the overall system: operations research techniques and simulation have been proven particularly apt to solve these kinds of problems (see, for instance, Kozan and Preston, 1999; Kim and Bae, 1999; Kim and Kim, 1999; Magnanti and Wong, 1984).

The approach currently adopted by most terminal managers splits the problem by treating each ship as an independent entity: a ship planner is dedicated to plan loading and unloading operations for a single ship and to allocate the needed resources in terms of quay cranes, yard cranes, lifters and manpower. Since there is no cooperation among different ship planners, this method is a source of conflicts and performance degradation mainly in the case of resources that must be shared among parallel loading and unloading

*Corresponding author.

activities. The complexity of the process leads to the separation of management activities, but even from the mathematical point of view formulating a single problem is not viable, given the complexity of the resulting model.

This paper proposes a methodological approach (Section 3) to solve the problem which still considers the interactions among the ship planners, but reduces the problem complexity by iteratively modeling it at different levels of detail. At the highest level, container moves during loading and unloading operations are seen as a flow inside the terminal. The goal is to determine the best allocation of resources (RA) on the yard with the objective of minimizing the costs of the terminal. This is done very efficiently by modeling the problem as a network design problem and by using a mixed-integer linear program solved by a branch-and-bound search (Section 4). At the lowest level, given the exact position of containers in the yard, the goal is to schedule the sequence of loading/unloading operations (LUL) in order to optimize the use of the allocated resources and to make sure that the predicted flow is sustainable. The problem at this level is modeled as a scheduling problem and solved using a new neighborhood function and a new self-tuning tabu search (Section 5). Last, a simulation module is used as a test bench to evaluate the overall policies produced by the optimization modules (Section 6). The application of the computer-generated policies to the simulated model (Section 7) of La Spezia Container Terminal (LSCT) shows a noticeable increase in terminal performance: the same amount of work can be performed by only spending about 67% of the resource costs originally planned at the terminal, while still respecting the deadlines imposed on the load/unload operations (Section 2).

2. The intermodal container terminal

An intermodal container terminal is a place where containers enter and leave by multiple means of transport, as trucks, trains and vessels. In the following we consider the case of the LSCT, which is exemplified in Fig. 1.

The terminal is composed of the yard, where containers are stocked, and of the piers, where ships dock. Ships are served by quay cranes; quay cranes on the same pier can slide on rails to serve different

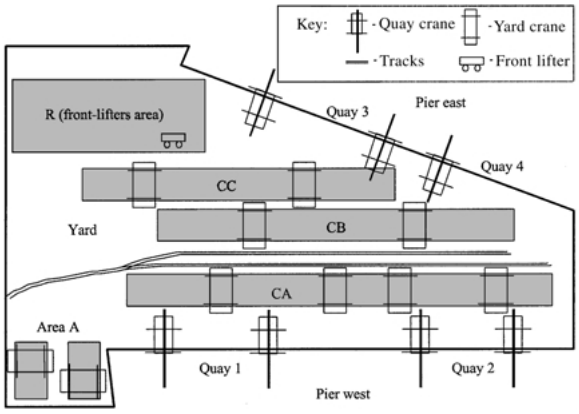

Fig. 1. A schematic view of the La Spezia container terminal.

sections of a ship, the bays. The position of a container (on ship and on yard) is identified by the triple bay, row (the second planar coordinate, besides the bay), and tier (the position in the container stack).

Shuttle trucks move containers from the quays to the yard areas and back. The terminal yard is divided into five areas: CA, CB, CC, A and R, which are used for temporary storage of containers. Containers in the R area are accessed by front lifters. The remaining areas are served by yard cranes. Yard cranes can only move over the related area. Their task is to load containers into the area from shuttle trucks (the cranes in CA and CB do it also from trains) and vice versa. In the LSCT there are ten front lifters and ten yard cranes (four on CA, two on CB and CC, two on A).

When a ship arrives in the terminal, it is first unloaded by a set of quay cranes. The unloaded containers will be stored in sub-regions of the yard areas, named import areas, where they can be stored by shuttle trucks, by yard cranes or by front lifters. When the unloading operations are completed the loading stage can start: containers which must be loaded on the ship are first discharged from the yard, loaded on shuttle trucks using yard cranes or front lifters, then each shuttle truck carries one container at a time to the ship where it will be loaded by using a quay crane.

3. The methodology

The problem of resource allocation is the most important since the terminal costs are a function of the resources (i.e., the quay cranes, the yard cranes,

the front lifters and the related operators) used to move the containers. The objective is to minimize the costs by properly allocating resources, while respecting the ships' deadlines. The solution to this problem must be found over a time horizon of one week without knowing in advance the exact positions of the containers in the yard.

The terminal is modeled as a network of flow (Papadimitriou and Steiglitz, 1982), the nodes of which represent yard areas, cranes, or ships. These are connected by arcs whose capacities depend on the allocated resources, which are decision variables. The containers to be moved are flows in the network. The graph of the network extends over all the work shifts in order to represent the model overtime. Deciding which and how many resources to allocate means finding the right balance between moving containers while saving resource costs. This corresponds to finding the best set of capacities of the arcs, which is referred to in literature as network design (Ahuja *et al*., 1993; Magnanti and Wong, 1984). We formulate the problem as a mixed-integer linear program. This is approximately solved in a short time by a branch-and-bound search (Papadimitriou and Steiglitz, 1982).

The scheduling of ship loading and unloading deals with the same problem but at a different level of detail. A few hours before the arrival of an incoming ship, the terminal receives detailed information about its stowage, i.e., containers that are to be stored in the yard (import containers), and about the list of containers which are to be loaded on the ship (export containers). This information allows the ship planners to generate the unloading list and the loading list. The unloading list specifies the order by which containers are to be unloaded from the ship in order to guarantee the ship stability (Sha, 1985). There is an unloading list for each quay crane serving the ship.

Thus, the goal is to move the containers to and from the ship, but at this stage the resources have been already allocated and the container positions are finally known. Containers are usually stacked up in piles, so that computing the right sequence of operations is crucial to increase performances and to avoid deadlocks among resources. This optimization problem has been modeled as an extension of the flexible job shop problem (Mastrolilli and Gambardella, 2000) where groups of containers are jobs, single containers are operations and resources are machines. The objective is to execute the loading and unloading operations with the given resources so that the makespan is minimized. The scheduling phase does not take into account economic factors, since the expenses have already been taken into account in the resource allocation stage. On the other hand, solving the scheduling problem with the allocated resources guarantees that the hypotesized container flows are sustainable in practice, when we consider single container moves and not flows. The problem of scheduling loading and unloading operations is heuristically solved by local search techniques (Aarts and Lenstra, 1995) and, in particular, by the tabu search paradigm (Glover, 1989). In this paper a deep investigation of our extended neighborhood function for the flexible job shop and our new self-tuning tabu search is presented. Using the neighborhood function and the tabu search in succession, it is possible to quickly (a few minutes on a Pentium 266 MHz) compute scheduling policies that increase the terminal performance. Notice that being able to obtain a good solution in a very short time makes the algorithm very attractive for terminal operators because it means that it is usable in real time.

Coupling simulation with optimization techniques plays a fundamental role both in testing the impact of proposed policies and to convince the terminal operators of their validity (Gambardella *et al*., 1998): in fact the policies are produced under deterministic assumptions, while a terminal is clearly a stochastic environment. A simulator can act as a test bench to show that the policies are sustainable and robust. Different authors have developed discrete-event simulation models of port terminals (Hayuth *et al*., 1994; Young and Seok, 1999). We also have developed an object-oriented, process-oriented, discrete-event simulator of the container terminal of La Spezia in Italy to which our study is applied.

The simulation model is therefore the place where the two optimization modules are used together (Fig. 2) to provide a solution to the integrated management of the terminal (Gambardella *et al*., 1998). The resource allocation module provides resources over an horizon of one week ($RA([D_i, D_i + 6])$, where D_i is day number i), but it is re-run at the end of every day, with the updated terminal state (current state) produced by the simulation of one day of work, that is, of four work shifts (I, II, III, IV). The allocated resources (Resources of Day i) are then used as input by the scheduling module (LUL) that computes the loading and unloading lists (L/U list) for each crane for the next work shift. It is then the simulator (SIM)

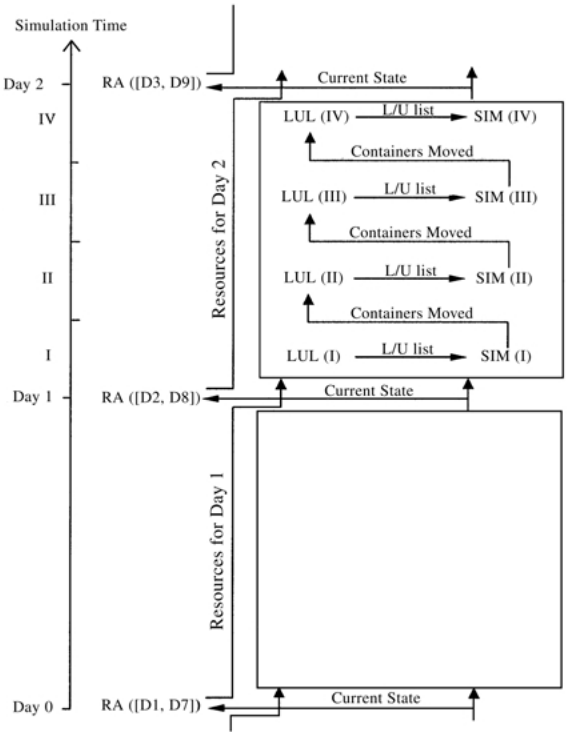

Fig. 2. Schema of the simulation-optimization loop; the modules are executed in a hierarchical way, from resource allocation to scheduling and simulation.

that uses these lists to perform terminal operations (containers moved), updating the terminal state. By repeating this ''allocate–schedule–simulate'' loop it is possible to assess the performance of the optimized policies in the integrated management of the terminal.

4. Resource allocation

Resource allocation is made on the basis of a workplan, like that presented in Table 1. Each row of Table 1 refers to a ship.

The ''Eta'' column reports the time at which the ship is expected. This is expressed as a shift number and the hour and minutes in such a shift. (A day is conventionally divided into four shifts of six hours.) The ''Deadline'' column reports the shift by which a ship must have been served. The last five columns display the yard distribution of all the containers to be moved for a ship. We do not distinguish between the inflow and the outflow because the allocation of resources is not concerned with the direction of moves.

A standard workplan describes a temporal horizon of one week (28 shifts). The workplan is a forecast that involves a degree of uncertainty that increases with time. For this reason, the resources are allocated only for the following day and the 1-week time window is updated daily. The time window is needed to balance the short-term and the mid-term information.

An algorithm solves the resource allocation problem by providing a set of resources for each shift in the workplan, as in Table 2. Now each row represents a shift: e.g., the first row suggests to allocate 1 quay crane at Pier West, 1 yard crane in CA, 1 in CB, 1 in CC and 2 front lifters (area R).

4.1. *The resource allocation model*

We model the terminal as a network of flow (Papadimitriou and Steiglitz, 1982). Consider, for the moment, the first shift in Table 1: a total of 100 containers must transit from the areas to the ship through the quay cranes at Quay 1. The graph in Fig. 3 describes this process.

The square nodes, which are sources of flow, represent the yard areas; the circle node stands for the quay cranes to be assigned to S1 and the rectangular node represents the ship (the sink of flow). The graph represents the paths that containers can follow to reach the ship. Arc labels denote arc capacities, i.e., the maximum number of containers that can flow through an arc during one shift. Capacities are proportional to the number of resources allocated for the related node. Deciding which and how many resources to allocate determines the structure of the

Table 1. An example workplan that represents the arrival of two ships

Name	*Quay*	*Eta*	*Deadline*	*CA*	*CB*	*CC*	*A*	*R*
S1	1	Shift 1 at 1:0	3	10	15	30	0	45
S2	2	Shift 2 at 7:0	3	20	5	14	10	9

Table 2. An example of allocation of resources for the workplan in Table 1

Shift	*Pier west*	*Pier east*	*CA*	*CB*	*CC*	*A*	*R*
1	1	0	1	1	1	0	2
2	1	0	1	0	0	0	1
3	1	0	0	1	1	1	0

flow network. This problem is known in literature as network design (Ahuja *et al.*, 1993; Magnanti and Wong, 1984).

Figure 4 reports the flow network extended to all the shifts (arc capacities are not shown).

Observe that the structure of the subnetwork over the upper dashed line is the same graph as that in Fig. 1. The nodes are renamed by adding superscripts for shift numbers and subscripts for ship names: e.g., CA^1_{S1} is the subset of containers stored in CA, at the start of shift 1, for ship S1. In this way, we regard each area of the yard as a set of logical sub-areas related to the shift and to the ship under consideration.

The vertical arcs are temporal arcs that lead to the next shift. For instance, the arc $CA^1_{S1} \rightarrow CA^2_{S1}$ gives the chance of delaying part of the moves of the 10 containers in CA to shift 2. Time arcs are uncapacitated because any number of containers can be postponed to the next shift. Observe that the graph does not extend beyond the third shift, thus forcing the service of S1 to be completed by its deadline.

The presence of two separate graphs in shift 2 is due to the presence of two ships at the terminal; the right side of the graph is used for S2. The two graphs are related by the resources shared between the ships: e.g., the cranes allocated in CA must be shared in order to direct the containers both to QC^2_{S1} and to QC^2_{S2}. This connection is made by using capacity constraints: in the preceding example we must require that the sum of the two flows does not exceed the overall capacity of the allocated area cranes.

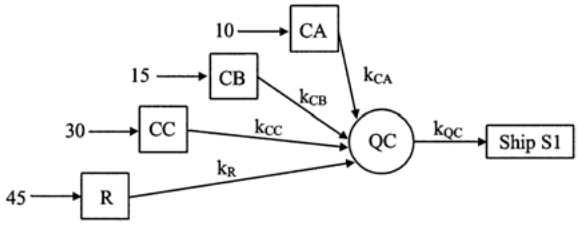

Fig. 3. The network of flow for the first shift in Table 1.

We want to compute the cheapest allocation of resources. This corresponds to optimally dimensioning the network to sustain the container flows. The amount of flow on the arcs is a by-product of the solution; in particular, the information about the load state of each ship at each shift is also available. This, together with the allocation plan, is used later by the scheduler (Section 5).

The graph model, further enhanced to include more details (such as the treatment of trains and trucks, see Zaffalon and Gambardella, 1998), is formalized as a mixed-integer linear program (Papadimitriou and Steiglitz, 1982), whose objective is the minimization of resource costs and solved using the branch-and-bound algorithm by the commercial solver Cplex 6 (Ilog, 1998).

5. Scheduling of loading/unloading operations

We formalize the unloading/loading problem as follows. A set $\mathbf{V} = \{v_1, v_2, \ldots, v_n\}$ of n ships is given and every ship $v_j \in \mathbf{V}$ has a known arrival time $r_j \geq 0$. We define a set of q quay cranes $QC = \{qc_1, qc_2, \ldots, qc_q\}$ and a set $YC = \{yc_1, yc_2, \ldots, yc_m\}$ of m yard cranes and front lifters. (In the following we include also front lifters in the yard crane set since in our model the only difference is the time that they need to move containers.) The sets QC and YC are determined by the resource allocation module (see Section 4). When a ship v_j arrives we assume, without loss of generality, that there is only one quay crane qc_j that works for v_j, for $j = 1, \ldots, n$ (the general case where more than one quay crane works for v_j is immediately obtained by considering one fictitious ship for every other quay crane available for v_j). The set of containers is partitioned into classes according to their features: in our example, we consider weight, destination and size as distinctive features. Each class has a finite set of positions where containers of the given class can be stored or picked up. We assume that each yard position can be accessed by a unique yard crane. We also assume that for each quay crane there is one truck that can carry containers from (to) quay crane to (from) location (in Remark 5.1 we show how the case in which there are more than one truck for each quay crane can be handled by using our model). We distinguish between the unloading and the loading stage.

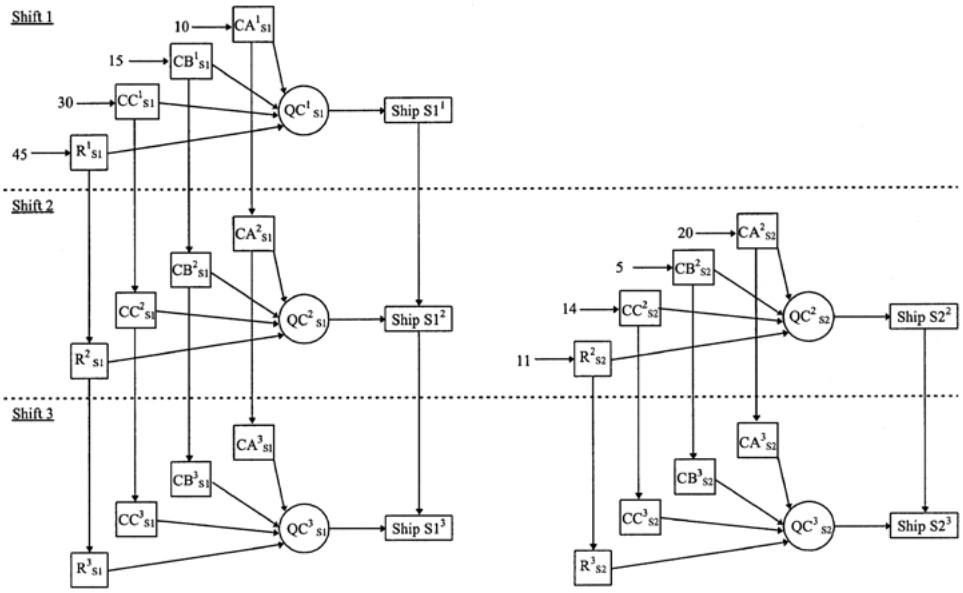

Fig. 4. The flow network for the 2 ships described in Table 1.

5.1. *The unloading stage*

For each shift, according to the resource allocation module and the unloading list, every ship v_j has a list $U(v_j) = (c_{j,1}, c_{j,2}, \ldots, c_{j,k_j})$ of k_j containers (k_j is computed according to the resource allocation module, see Section 4) to be unloaded in the given order, i.e. container $c_{j,i}$ is unloaded before $c_{j,i+1}$, for $j = 1, \ldots, k_j - 1$. It is out of the scope of this paper to discuss how this list is determined, we only say that it is computed to ensure ship stability. The set $U(v_j)$ of containers is partitioned into classes according to their characteristics. Each class has a finite set of potential positions in the yard where containers belonging to that class can be stored. The unloading stage problem is to assign each container to a unique free yard position (hence to a unique yard crane according to our assumption) and to define the processing order of each yard crane such that the maximal finishing time (makespan) is minimized. More precisely, each container $c_{j,i}$ has to be: (Step 1) unloaded from the ship v_j by using quay crane qc_j, loaded on a truck and moved to the chosen yard position; (Step 2) loaded in the selected yard position by using the associated yard crane. We assume that Step 1 can start only when a truck is available. Moreover Step 2 can start only after step 1 is completed. We represent the described steps as a chain of operations (see Fig. 5).

In Fig. 5 the following notation is adopted: operations 0 and * are dummy and are used to represent the start and the end; operations of ship v_j can start after r_j time units (release time); C_{ji1} and C_{ji2} denote Step 1 and Step 2 for container c_{ji}; p_{ji1} denotes the required time for Step 1; p_{ji2} denotes the required time by the yard crane to make the truck free plus the time required by the truck to come back from the yard area where c_{ji} is stored to qc_j. Let p_{ji3} denote the time required by a yard crane to move container c_{ji}. For instance, assume that a yard crane processes, in the order, containers c_x, c_{j2} and c_y. That situation can be represented by using a graph as in Fig. 6.

Note that the values of p_{ji1}, p_{ji2} and p_{ji3} depend on the chosen position for container c_{ji}. Since the yard crane that moves the considered container is close to

Fig. 5. Step order.

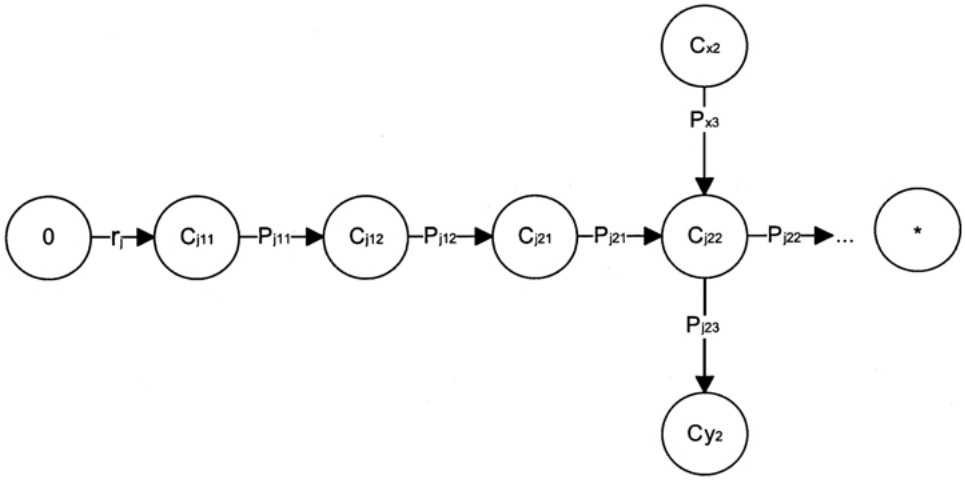

Fig. 6. Graph model.

the chosen position we approximately assume that p_{ji1}, p_{ji2} and p_{ji3} depend on the yard crane associated with the chosen position. We claim that this assumption can be easily removed at the cost of a more computational expensive model. In our example we observed that $p_{ji2} \geq p_{ji3}$, and we will assume this in the following.

5.2. *The loading stage*

Every ship v_j has a fixed set of containers to be loaded and its loading list $L(v_j) = (c_{jk_j+1}, c_{jk_j+2}, \ldots, c_{j\mu_j})$ specifies that $\mu_j - k_j$ containers have to be loaded on ship v_j and the ith loaded container belongs to class c_{ji}, for $i = k_j + 1, \ldots, \mu_j$. Note that here, unlike the unloading stage, c_{ji} does not represent a container, but a class; we will see that this will allow us to describe the whole problem without distinguishing between the unloading and loading stage. The order in the loading list is computed according to ship stability and containers destination. For each class there is a finite set of yard positions where containers of that class can be retrieved. This set of alternatives is an input of our algorithm that depends on where the containers of that class are placed during the considered shift.

The loading stage problem is to determine for each class c_{ji} the yard position from which a container of class c_{ji} can be retrieved and to define the processing order of each yard crane such that the maximal finishing time (makespan) is minimized. When a yard position for c_{ji} is chosen, the ith loaded container on ship v_j is the one stored in that position, for $i = k_j + 1, \ldots, \mu_j$. For each chosen container: (Step 1) a truck has to go from ship v_j to the chosen yard position; (Step 2) the container has to be moved from the position where it is stored and loaded on the truck that carries it to ship v_j. Interestingly, the described steps can be represented in the same way as in the unloading stage, see Fig. 5, when the following notation is adopted: C_{ji1} and C_{ji2} denotes Step 1 and Step 2 for the chosen container of class c_{ji}; p_{ji1} denotes the time required for Step 1; p_{ji2} denotes the time required by the yard crane to move the container of class c_{ji} in the chosen position to the truck plus the time required by truck to come back from the yard area to qc_j. In the remainder c_{ji} denotes also the corresponding container when a yard position is chosen for class c_{ji}. Let p_{ji3} denote the time required by yard crane to move container c_{ij}. Suppose for instance, that a yard crane processes, in the order, containers c_x, c_{j2} and c_y. By using the loading stage symbol interpretation that situation can be represented as in Fig. 6. Again, like for the unloading stage, we approximately assume that p_{ji1}, p_{ji2} and p_{ji3} depend on the chosen yard crane for container c_{ji}. Note that $p_{ji2} \geq p_{ji3}$ by definition.

5.3. *A comprehensive model*

In the remainder we present a graph-based model of the problem and we will see that by using the adopted

model there is no distinction between the unloading and the loading stage.

We call each pair (C_{ji1}, C_{ji2}) operation $O_{ji}, 1 \leq j \leq n$ and $1 \leq i \leq \mu_j$. For each ship v_j there is a corresponding job J_j that is a sequence of operations to be performed in the order $J_i = (O_{j1}, O_{j2}, \ldots, O_{j\mu_j})$. For each operation O_{ji} there is a non-empty set of machines (yard cranes) $M_{ji} \subseteq YC$ that can perform that operation. For each $y \in M_{ji}$ three processing times $(p^y_{ji1}, p^y_{ji2}, p^y_{ji3})$ are given (they are retrieved from LSCT data). A schedule is a pair (μ, s) that defines for each operation O_{ji} a unique machine $y = \mu(O_{ji})$, where $y \in M_{ji}$, on which O_{ji} is processed and a pair of starting times $s_{ji} = (s_{ji1}, s_{ji2})$ of Steps 1 and 2, respectively. When a machine assignment is defined, the position where a container is stored is the first available that is served by the chosen machine; while for the loading stage the picked up container is the first available of the required class in the area that is served by the chosen machine. A machine assignment μ is feasible: (Condition 1) if $\mu(O_{ji}) \in M_{ji}$, for each O_{ji}; (Condition 2 for the unloading stage) if there exists an allowed free position for each container that is processed according to μ; (Condition 2 for the loading stage) if for each operation O_{ji} assigned to an allowed machine $y \in M_{ji}$ there exists a container of class c_{ji} that can be moved by y. A schedule is feasible if the machine assignment is feasible and the starting times for each operation satisfy the described precedences. The length $C_{\max}$ (or makespan) is the earliest time at which all operations are completed. The problem is to find an optimal schedule, i.e., a feasible schedule of minimum makespan.

It is possible to represent each feasible schedule (μ, s) by using a directed acyclic graph G (μ, s), as described above. The makespan of (μ, s) is equal to the length of a longest (also called critical) path from 0 to * in $G(\mu, s)$. Note that the s_{ji1} and s_{ji2} correspond to the length of the longest path from 0 to C_{ji1} and C_{ji2} in $G(\mu, s)$, respectively. In the remainder $p^y_{ji1}, p^y_{ji2}, p^y_{ji3}$ will be written without superscript y, if it is clear what yard crane y processes operation O_{ji}. In addition, tail times t_{ji1} and t_{ji2} are defined. It corresponds to the length of a longest path from C_{ji1} and C_{ji2} to node *, respectively. Denoting the value of one of the longest paths from node i to j in $G(\mu, s)$ by $l(i,j)$, it is, $s_{ji1} = l(0, C_{ji1}), s_{ji2} = l(0, C_{ji2}), t_{ji1} = l(C_{ji1}, *)$ and $t_{ji2} = l(C_{ji2}, *)$. An operation O_{ji} is critical if, and only if, $s_{ji2} + t_{ji2} = C_{\max}$. For any given feasible schedule, the labeling algorithm of Bellman can be used in order to compute the makespan and all values of $s_{ji1}, s_{ji2}, t_{ji1}$ and t_{ji2} in time $O(N)$, where N is the number of operations.

Remark 5.1 It was assumed in Section 5 that there is only one truck for each quay crane. This assumption can be avoided to some extent by adding a new job for each new truck and by partitioning the set of operations of each ship among trucks. We remark that the presented model can deal with that extension if quay cranes can be considered as non-bottleneck. An examination of real data supported our hypthesis. Another way to deal with that situation is to make a proportional increase in trucks speed.

5.4. *Solving the scheduling problem*

Our approach to solve the LUL is based on local search (Aarts and Lenstra, 1997). Consider the minimization problem $\min\{f(x) \mid x \in X\}$, where f is the objective function and X is the search space, i.e. the set of feasible solutions of the problem. The neighbourhood function is a mapping $\mathcal{N} : X \to 2^X$, which defines for each solution $x \in X$ a subset $\mathcal{N}(x)$ of X, called neighborhood. Each solution in $\mathcal{N}(x)$ is called a neighbor of x. A local search algorithm starts from some initial solution and moves from neighbor to neighbor as long as possible while decreasing the objective function value.

In the following we relax Condition 2 of the machine assignment feasibility. We will consider this condition in the search procedure (see Section 5.4.3). Note that when $r_j = p^y_{ji1} = 0$ and $p^y_{ji2} = p^y_{ji3} (1 \leq j \leq n, 1 \leq i \leq \mu_j, y \in M_{ji})$ the problem is equivalent to the flexible job shop problem (Mastrolilli and Gambardella, 2000). Therefore, the loading/unloading problem is NP-hard since it is a generalization of the classical job shop problem (Garey *et al.*, 1976). In the following we propose a new neighborhood function that is able to handle this more general problem. The neighbor of a solution (s, μ) in the proposed approach, is based on moving an operation. Suppose that a feasible solution of the LUL is given and let G be the corresponding solution graph. Moving an operation O_{ji} is done in two steps: (Step 1) delete O_{ji} from its current machine sequence by removing all its machine edges; (Step 2) Assign O_{ji} to a machine $k \in M_{ji}$ and choose the position of O_{ji} in the processing order of k. Let G^- be the graph

obtained from G at the end of step 1 and let O_{ji} denote the operation to be moved. Notice that, for each node x in G^-, it is $l(0,x)^- \leq l(0,x)$ and $l(x,*)^- \leq l(x,*)$, since the removal of an operation from a machine cannot increase the starting and tail times.

5.4.1. *Feasible move*

A move is feasible if it does not create a cycle in the resulting graph at the end of Step 2. Clearly G^- is acyclic since G is already acyclic. We denote the new machine of O_{ji} by k, where $k \in M_{ji}$. Now, let Q_k be the set containing all the operations processed on k, in $G^-(O_{ji} \notin Q_k)$. If $Q_k = \emptyset$ then no task is performed by machine k and no cycle is created at the end of Step 2. Otherwise, inserting operation O_{ji} in machine k means deciding what is the subset of Q_k that is to be processed before O_{ji}. Hence, any partition of Q_k into disjoint sets of predecessors P_k and successors $S_k = Q_k \backslash P_k$ define an insertion of O_{ji} in k. Let us denote such a reinsertion by (O_{ji}, P_k). The corresponding solution graph G^{P_k} is obtained from G^- by adding arcs (u, O_{ji}) for all $u \in P_k$, and arcs (O_{ji}, w) for all $w \in S_k$. If a path from x to O_{ji} exists in G^-, then x must be scheduled before O_{ji} in order to obtain a feasible solution, otherwise a cycle is produced. Similarly, if a path from O_{ji} to x exists in G^-, then x must be scheduled after O_{ji}. Finally, if no path between x and O_{ji} exists in G^-, then every insertion of O_{ji} just after or before x leads to a feasible solution. In the following sufficient conditions for feasibility are presented. By using these conditions we can compute a set of feasible moves which define the neighborhood of the current solution.

Let us define two subsets of Q_k as follows, $R_k = \{x \in Q_k \mid s_{x2} + p_{x3} > s_{ji1}\}$ and $L_k = \{x \in Q_k \mid t_{x2} > t_{j(i+1)1}\}$. By definition of R_k there is no path from any task of R_k to O_{ji} in G^-. If x is a task of $Q_k - R_k$ then $s_{x2} + p_{x3} \leq s_{ji1}$. It follows $s_{x2} < s_{ji1} + p_{ji1}$. The latter entails that there is no path from O_{ji} to any operation of $Q_k - R_k$ in G (therefore neither in G^-). Analogously, by definition of L_k there is no path from O_{ji} to any task of L_k in G^-. If x is a task of $Q_k - L_k$ then $t_{x2} \leq t_{j(i+1)1}$. The latter entails that there is no path from any operation of $Q_k - L_k$ to O_{ji} in G (therefore neither in G^-). We distinguish between two cases, i.e., $L_k \cap R_k \neq \emptyset$ and $L_k \cap R_k = \emptyset$. If $L_k \cap R_k \neq \emptyset$ then $\forall x \in L_k \cap R_k$ no path from O_{ji} to x to O_{ji} exists in G^-. If $L_k \cap R_k = \emptyset$, $\forall x \in Q_k - L_k - R_k$, no path from O_{ji} to x and from x to O_{ji} exists in G^-. We only consider sets P_k and S_k for which

$$L_k - R_k \subseteq P_k \tag{5.1}$$

$$R_k - L_k \subseteq S_k \tag{5.2}$$

Theorem 5.2
Any insertion (O_{ji}, P_k) that satisfies conditions (5.1) and (5.2) is a feasible insertion.

5.4.2. *Neighborhood function*

In this section we present a neighborhood function $\mathcal{N}(S)$ which can be used in local search methods for the LUL.

Moving an operation produces a neighbor of a current solution S. In order to minimize the makespan, it may be profitable to consider only neighbors which are obtained by reinserting operations that belong to some critical path P in the solution graph of the current schedule. Consider reinserting an operation that does not belong to a longest path in the solution graph of the current schedule. This produces a neighbor whose path is at least as long as the longest path in the solution graph of the current schedule. The reason for this is that the longest path of the current schedule remains present in the solution graph of such a neighbor.

Now, $\forall v \in P$ consider every machine $k \in M_v$ and the set F_{vk} of feasible insertions of v in k, that satisfy that conditions of Theorem 5.2. Our new neighborhood function $\mathcal{N}(S)$ is defined as the set of every F_{vk}, for each operation $v \in P$ and each machine $k \in M_v$.

5.4.3. *Tabu search procedure*

Our search procedure is based on local search implemented by using a tabu search mechanism (Glover, 1989). Tabu search allows the search to explore solutions that do not decrease the objective function value only in case when these solutions are not forbidden. This is usually obtained by keeping track of the last T (where T is a parameter) solutions in term of action used to transform one solution to the next. This list of actions represents the tabu list of size T associated with the search mechanism. A solution is forbidden if it is obtained by applying a tabu action to the current solution. In the case of the present work, an action is composed by the couple (O_{ji}, k) where O_{ji} is the task that has been moved and k is its original yard crane before the move. In order to keep track of the actions performed, we use a $N \times m$ matrix TM. Each time, action (O_{ji}, k) is performed, $TM(O_{ji}, k)$ is set to iter $+ T$, where iter is the current iteration number and T is defined in the following. An action (O_{ji}, k) is tabu if $TM(O_{ji}, k) >$ iter. The tabu status length T is crucial

to the success of the tabu search procedure, and we propose a self-tuning procedure based on empirical evidence. T is dynamically defined for each operation v and each solution. It is equal to the number of operations of the current critical path P plus the number of alternative machines available for operation O_{ji}, i.e., $T := |P| + |M_{ji}|$. We choose this empirical formula since it summarizes, to some extent, the features of the given problem instance and those of the current solution. For instance, there is a certain relationship between $|P|$ and the instance size, between $|P|$ and the quality of the current solution. In order to diversify the search it may be unprofitable to repeat the same action often if the number of candidate actions is ''big'' or the solution quality is low, in some sense, when $|P|$ is ''big''. Furthermore, the tabu status length of O_{ji} is augmented by $|M_{ji}|$ in order to diversify the machine assignment of O_{ji}.

We denote as best move the one with the smallest estimated length of the new longest path passing through the moved task. If some non-tabu moves exist, the next one is stochastically chosen between the best two non-tabu moves. This method is useful to decrease the probability to generate cycles. In order to explore the search space in a more efficient way, tabu search is usually augmented with some aspiration criteria. The latter are used to accept a move even if it has been marked tabu. In the present case a move of task O_{ji} is always accepted if the estimated length of the new longest path passing through O_{ji} decreases the best found makespan (when several moves satisfy the previous condition, the best one is chosen). The last situation happens when only tabu solutions are available. In this case, the chosen solution is the first introduced in the tabu list.

We have relaxed Condition 2 of the machine assignment feasibility in Section 5.4, in order to take into consideration also this condition we modify the search procedure as follows. Each time the machine assignment feasibility is not satisfied, a penalty is added to the objective function value. In this way the search procedure moves from solution to solution which could be feasible or unfeasible.

6. Simulation

The solution of the resource allocation and scheduling problems produced some very promising results, as outlined in the previous sections. These need to be verified in a complex stochastic environment: we designed and implemented a simulator to be used as a test bench to evaluate the management policies produced by the optimization modules (Gambardella *et al*., 1998).

6.1. *The simulation model*

The simulation model has been designed according to the object-oriented analysis and design paradigm (Booch, 1994): simulation agents and components are modeled as objects that store and exchange information on terminal inputs, states and outputs and which perform actions according to their local behavior (Zeigler, 1984, 1990).

There is a hierarchy of simulation objects according to the importance of the decisions they take. Planners, such as yard and ship planners are at the top, since they take decisions on resource and space allocations. Crane operators (yard and quay) and shuttle truck drivers occupy the middle layer, and at the bottom, there are the terminal components, such as yard areas, containers, and other agents such as ships, trains and trucks, whose behavior is imposed as an external constraint and it is not directly controllable by the terminal operator.

The simulation module has been implemented in Modsim III (CACI, 1996), a specialized object-oriented simulation language, which supports the process-oriented view of discrete-event simulation (Erard and Déguénan, 1996). A detailed description of the simulation model we have implemented can be found in Gambardella *et al*. (1998). In the remainder we summarize the main characteristics of the model, with respect to its use as a tool to assess the performance of the optimized terminal management policies.

External input events, such as ship, truck, and train arrivals, generate responses by the simulation agents which in turn operate on simulation components. These events are generated by a specialized module, the arrival generator, which reads ship and train arrivals from the database, since they are known in advance, while truck arrivals are generated according to known statistical distributions. We are therefore performing a ''trace-driven'' simulation.

The responses of simulation agents are determined according to the policies which can either be generated by the optimization modules or by an

algorithmic representation of the experience of terminal operators. We have therefore identified two principal usage modes of the simulator: model calibration and validation and policy evaluation.

In the model calibration mode, the simulation model has been designed to replicate the actual behavior of the ship and yard planners. Their workflow has been studied and transferred into a computer model. In this model a grand total of 2950 containers is moved over 10 work shifts and a set of 11 experiments (simulations) was performed. Each experiment is identified by a different combination of the calibration parameters. In order to find the best combination of parameters among the different experiments, the sum of the absolute weighted errors over the examined work shifts for each quay crane and yard crane is computed. Aggregating the performance indicators over the quay and yard cranes, we obtained a ''global'' simulation error equal to 12% in calibration and to 18% in validation. More information on the model calibration and validation can again be found in Gambardella *et al*. (1998).

In the policy evaluation mode (Section 7), some procedures in the models of the ship and yard planners have been substituted by the optimized policies we have obtained in the resource allocation and the operation scheduling phases. In particular, the ship planner uses the scheduling algorithm to compute the loading and unloading lists that are then given for execution to all the cranes, both yard and quay ones.

7. Experimental analysis

We tested our methodological approach against the complex stochastic environment provided by the simulated terminal, by using two weeks of real data provided by LSCT. These data describe the activity of LSCT in great detail, in such a way that every container movement and the time required to move it was logged. Furthermore, the data set tracks the activity of every transport means entering and leaving the terminal (ships, trucks and trains). For the considered period we also obtained data describing the real resources allocated, the time of each loading/unloading operation, the origin, the destination and the transport means of each container movement.

The economic parameters are the costs originated by the resources and they are usually the sum of a number of factors, like the salary of the human operators and the cost for maintaining the resource active. Table 3 summarizes the unitary costs (in fictitious units) for each type of resource. Notice that the costs are different for day or night shifts, the latter (from 7 pm to 7 am) being more expensive.

As for the workplan, this consists of 11 ships to be served in the first week and about 12,400 containers to be moved.

To test the effectiveness of our procedure we consider two types of experiments: in the first experiment, for each considered shift we use the same initial status and resources allocated by LSCT (Table 4). The goal is to test the effectiveness of our scheduling policy under the same conditions and the same containers to be moved. In Fig. 7, the original scheduling by LSCT is reported and displayed by the graphical user interface of our optimization software. The yard cranes are listed on the *y*-axis, while time (in seconds) on the *x*-axis. Each box represents an operation (a container move), its width being the time taken to complete the operation. The numbers associated with each box identify the quay crane that has performed that operation. In Fig. 8, the same representation displays the optimized schedule for the same set of cranes in the same shift, where the makespan is reduced from 20,640 s to 17,980 s. On average we observed that all the tasks are completed about 1 h before the end of each considered shift.

In the second experiment, we test whether our scheduling policies are able to move the same

Table 3. Unitary resource cost per shift

	Cranes			*Front lifters*
	CA, *CB*, *CC*	*A*	*QC*	
Daily	45,177	48,901	14,566	17,057
Night	52,344	56,067	18,827	59,651

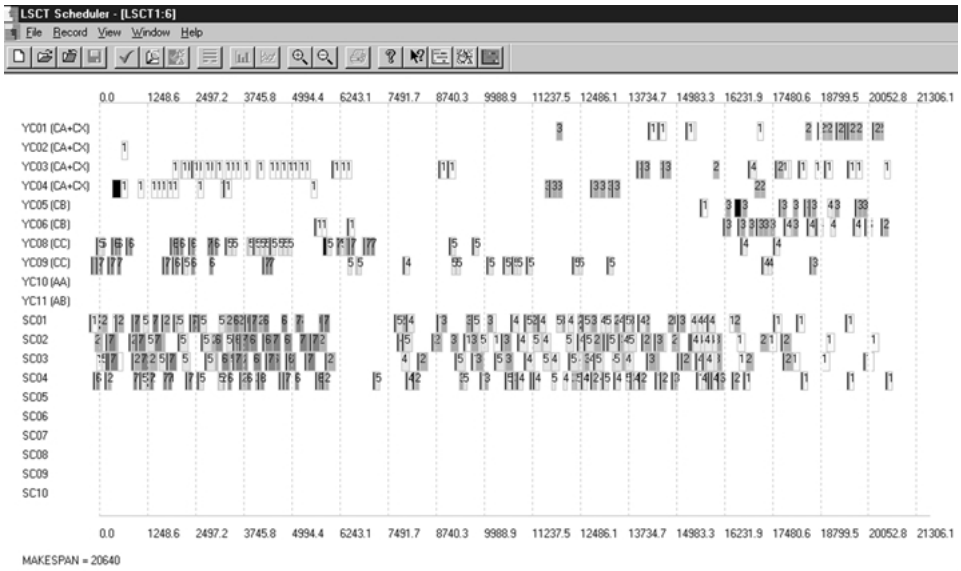

Fig. 7. The Gantt chart represents the scheduling problem adopted by LSCT.

containers by considering the same initial status but a reduced set of resources, as computed by the resource allocation module. The resources are computed by a large mixed-integer linear program (e.g., about 2000 variables, 2000 constraints), solved by the commercial solver Cplex 6 (Ilog, 1998). Cplex turns out to be very effective in firstly reducing the size of the problem by default pre-processing by an average factor 0.05. The search space is then explored by the method of branch and bound. We found the following search strategy to be particularly effective: exploring the tree by depth-first search; choosing a branch related to the ceiling of a fractional resource variable first.

Cplex is used as a way to get approximate solutions by stopping the search after a time limit, because a

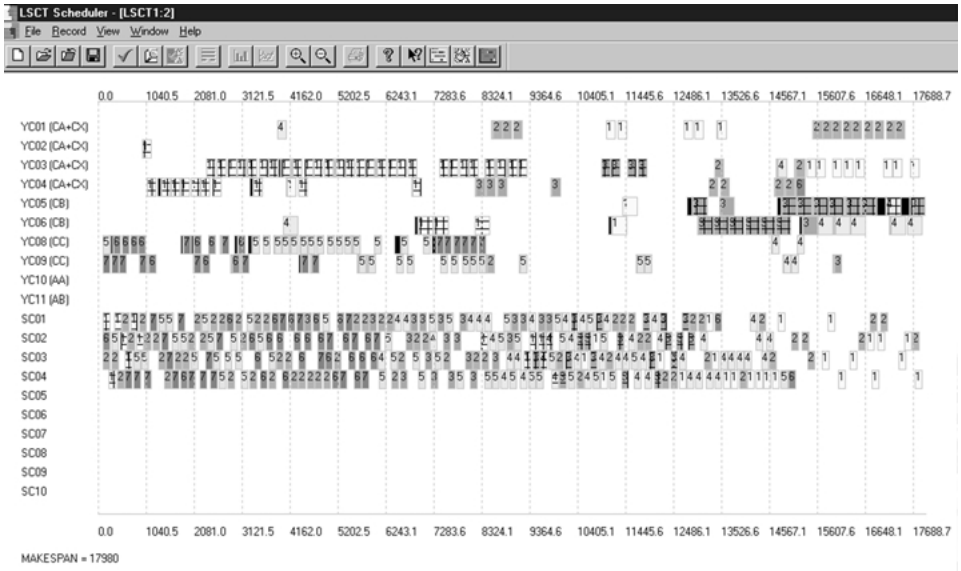

Fig. 8. The Gantt chart represents the optimized schedule, proposed by the optimization program.

Table 4. Summary of the computer-generated plan of resource allocations

	Cranes						*Front lifters*	
	CA + CB + CC		*A*		*QC*			
	LSCT	*RA*	*LSCT*	*RA*	*LSCT*	*RA*	*LSCT*	*RA*
Daily	96	63	11	14	36	36	63	46
Night	72	38	2	3	31	14	45	12
Total	**168**	**101**	**13**	**17**	**67**	**60**	**108**	**58**

complete search in the tree does not seem viable. The approximate solutions are very satisfactory: 1 min computation on a pentium 133 MHz is sufficient to produce integer feasible solutions whose average relative distance from the continuous bound is about 6.7%; and they outperform the real policies implemented at the terminal by allowing, on average, a reduction of costs of about $\frac{1}{3}$.

Table 4 summarizes the LSCT and the computer-generated resource allocations.

The comparison of the two allocation plans outlines the effectiveness of the computed allocations. The overall number of resources is much lower than that of the real allocation. Also, there is a more convenient partition of the resource between daily and night shifts, with the computed plan that better exploits the cheaper shifts. Therefore, by having the global view of the terminal, the module shares the resources among different tasks in a more convenient way than the real terminal does, where the decisions are taken independently by operators related to different ships.

Starting from the resources allocated by the RA module, loading and unloading sequences have been computed by LUL. These solutions show notable advantages over the real solutions implemented at the terminal. The computer-generated allocations do not impose a quay crane intensity (number of container moved per hour) higher than the one normally attained by the real world terminal. We observed an average crane intensity of 20.4 containers/h in the real world, against a crane intensity of 21.2 in the simulation with the optimized policies. We found that the improvement is instead due to a better organization of the yard crane work. In fact, it turns out that optimized loading/unloading lists allow a more efficient use of yard cranes, thus increasing their intensity from 6.7 containers/h to 9.9. This increase of the intensity is actually possible since the yard crane ''conflicts'' (i.e. when two yard cranes try to access two containers which are very close at the same time) are dramatically reduced.

Another desirable side effect of optimization is the average reduction of queue lengths under yard cranes. This leads to faster round-trip times for shuttle trucks, thus increasing the container flows from quay to yard. This effect has been observed using the detailed discrete-event based simulation model. Such a model allowed to evaluate and assess the effects of the designed policies on a more accurate model of the terminal, enhancing the managers confidence in the goodness of the computer-generated policies. The simulator has been therefore used not only as a workbench to evaluate the policies in front of stochastic unexpected events, but also as a communication tool to interact with the terminal managers.

8. Conclusions

The paper presents a methodological approach to the optimization of an intermodal terminal. The problem of resources allocation and scheduling of containers is solved by using two different but strictly interconnected modules. The resources allocation module remarkably reduces the number of resources typically required by the terminal. The optimized plans tend to exploit the resources to their limit, so that only an effective scheduler, as the one presented, can cope with the increased complexity of this task. The scheduler achieves such a result by reducing the conflicts of the yard cranes, allowing their throughput to be dramatically increased.

The complexity of the scheduling problem required state-of-the-art local search techniques and tabu search to be used. The local search implemented is based on a set of neighborhood solutions that can be

computed very efficiently by using the formal properties provided in this paper. We have also implemented an innovative ''self-tuning'' tabu search where the only parameter of the search algorithm, i.e. the tabu status length, is automatically tuned according to the problem instance under consideration.

The feasibility of the overall optimized management of the terminal has been assessed thanks to a discrete-events based simulator, which has been previously validated and calibrated on a large set of real data. The simulator showed all the computer-generated policies to be sustainable in such a complex stochastic environment, thus emphasizing the robustness of the proposed approach.

Acknowledgments

The authors would like to thank the management group of the La Spezia Container Terminal for their effective collaboration in this research project. Thanks also to Data and System Planning SA, in Manno (CH), for support in the software development. This project was funded by the Swiss commission for technology and innovation under the project #3128.1.

References

Aarts, E. H. and Lenstra, J. K. (1997) Introduction. *Local Search in Combinatorial Optimization*, John Wiley & Sons, Chichester, pp. 1–17.

Ahuja, R., Magnanti, T. and Orlin, J. (1993) *Network Flows*, Prentice Hall, New Jersey.

Booch, G. (1994) *Object-oriented Analysis and Design: with Applications*, 2nd edn, Rational, Santa Clara.

CACI Products Company (1996) *Modsim III, the Language for Object-oriented Programming. Users' Manual*, La Jolla, CA.

Gambardella, L. M., Rizzoli, A. E. and Zaffalon, M. (1998) Simulation and planning of an intermodal container terminal. *Simulation*, **71**, 107–116.

Erard, P.-J. and Déguénon, P. (1996) *Simulation par éveénements discrets*, Lausanne Presses Polytechniques et Universitaires Romandes.

Garey, M. R., Johnson, D. S. and Sethi, R. (1976) The complexity of flowshop and jobshop scheduling. *Mathematical Operators Research*, **1**, 117–129.

Glover, F. (1989) Tabu search—Part I. *ORSA Journal on Computing*, **1**, 190–206.

Hayuth, Y., Pollatschek, M. A. and Roll, Y. (1994) Building a port simulator. *Simulation*, **63**, 179–189.

Ilog (1998) *Cplex 6*, www.ilog.com.

Kozan, E. and Preston, P. (1999) Genetic algorithms to schedule container transfers at multimodal terminals. *International Transactions in Operational Research*, **6**, 311–329.

Kim, H. K. and Bae K. H. (1999) Segregating space allocation models for container inventories in port container terminals. *International Journal of Production Economics*, **59**, 415–423.

Kim, H. K. and Kim Y. K. (1999) Routing straddle carriers for the loading operation of containers using a beam search algorithm. *Computers and Industrial Engineering*, **36**, 109–136.

Magnanti, T. L. and Wong, R. T. (1984) Network design and transportation planning: models and algorithms. *Transportation Science*, **18**, 1–56.

Mastrolilli, M. and Gambardella, L. M. (2000) Effective neighborhood functions for the flexible job shop problem. *Journal of Scheduling*, **3**, 3–20.

Papadimitriou, H. and Steiglitz, K. (1982) *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, New York.

Sha, O. P. (1985) Computer aided on board management. *Computer Applications in the Automation of Shipyard Operation and Ship Design V*, P. Banda, C. Kuo, (eds), Elsevier, Amsterdam, pp. 177–187.

Young, Y. W. and Seok, C. Y. (1999) A simulation model for container-terminal operation analysis using an object-oriented approach. *International Journal of Production Economics*, **59**, 221–230.

Zaffalon, M. and Gambardella, L. M. (1998) Optimization of resources in an intermodal terminal, *IDSIA Technical Report n. IDSIA-20-98*.

Zeigler, B. P. (1984) *Mutifacetted Modelling and Discrete Event Simulation*, Academic Press.

Zeigler, B. P. (1990) *Object-oriented Simulation with Hierarchial Modular Models: Intelligent Agents and Endomorphic Systems*, Academic Press.