Original Article

A new three-step heuristic for the Master Bay Plan Problem

Daniela Ambrosino[a,*], Davide Anghinolfi[b], Massimo Paolucci[b] and Anna Sciomachen[a]

[a]Department of Economics and Quantitative Methods (DiEM), University of Genova, Genova, Italy.
[b]Department of Communications, Computer and System Sciences (DIST), University of Genova, Genova, Italy.
E-mails: davide.anghinolfi@unige.it, paolucci@dist.unige.it

*Corresponding author.

Abstract In this work, we are looking at the problem of determining stowage plans for containerships. This problem, denoted in the literature as the Master Bay Plan Problem (MBPP), is computationally difficult to solve, that is NP-hard. We start from the optimal solution of subsets of bays related to independent portions of the ship, which are determined by a previously proposed decomposition approach for the MBPP; then, we look for the global ship stability of the overall stowage plan by using a tabu search (TS) metaheuristic approach. Note that at the same time the proposed TS algorithm allows us to further reduce the handling time of the containers to be loaded on the ship. The proposed heuristics has been implemented within a software support system that helps the planning management in the visualisation of the stowage plans of each bay of the ship. Preliminary computational experimentations performed on some real-life test cases related to a terminal located at the port of Genoa, Italy are provided.

Maritime Economics & Logistics (2009) **11,** 98–120. doi:10.1057/mel.2008.19

Keywords: maritime logistics; stowage plans; combinatorial optimisation; heuristics; tabu search

Introduction and Problem Definition

Maritime container terminals are very important logistic nodes in freight transportation networks, and the import and export container's flows, as well as

the transhipment one. They have to be managed to optimise their logistic costs and the terminal productivity (Peters, 2001). In this work we focus our attention on the quay and ship activities, having in mind the berthing time of a containership, which can be considered as consisting of different components, such as time for loading/unloading operations and waiting time before and after the operations. As our main goal is the minimisation of the berthing costs, we focus on the container loading process, which is very difficult and the one most affecting the efficiency of the terminal operations (Imai *et al*, 2002).

The stowage of a containership, that is the so-called Master Bay Plan Problem (MBPP), is carried out daily by each terminal management (Thomas and Roach, 1989). In the past period, stowage plans for containers were performed by the captain of the ship; today, the maritime terminal has to establish the master bay plan, in accordance with the stowage instructions of the ship coordinator representing the company holding the ship. In particular, inside each company for managing a terminal container there is a planning office where planners define the master bay plan for each ship approaching the terminal. The planning office knows the profile of the ship in relation to both structural and operational information of the ship and he communicates with the planning office of the last terminal visited by the ship, the port authority and the maritime agencies asking about containers' content.

The stowage of a containership involves different objectives; among others, it is required to optimise the available spaces, prevent damages to the goods, the containership, its crew and its equipment, and minimise the berthing time of the containership at the terminal.

Operations research techniques play a crucial role in the definition of stowage plans; an interesting review of the literature about such methods can be found in Steenken *et al* (2004), where the authors split the problem into a two-step process, concerning respectively, the shipping line and the terminal manager, providing for each of them the corresponding optimisation models. A literature update is provided in Stahlbock and Voss (2008).

As it has been already mentioned, here we consider MBPP mainly as a loading problem with the terminal manager as a decision maker. Thus, MBPP can be more formally defined as follows: given a set C of n containers of different types to be loaded on the ship and a set S of m available locations on the containership, we have to determine the assignment of each container to a location of the ship, in such a way, to satisfy all the given structural and operational constraints related to ship and containers, and to minimise the total stowage time.

Each location of the ship is addressed by three indices, namely i, j, k, with the following meaning: the bay (i), which gives its position related to the cross-section of the ship (counted from bow to stern), the row (j), which gives its

position related to the horizontal section of the corresponding bay (counted from centre to outside), and the tier (k) which gives its position related to the vertical section of the corresponding bay (counted from bottom to top).

Let I, J and K be, respectively, the set of bays, rows and tiers of the ship; K is split into subset K_H of tiers in the hold and subset K_D of tiers in the deck. Set C is given by the union of two subsets, T and F, consisting respectively, of 20- and 40-feet containers, which is $T \cup F \equiv C$. Standard locations are generally for dry 20-feet containers (denoted by 20′) and referred to as one Twenty Equivalent Unit (TEU) location. Containers of 40 feet (denoted by 40′) hence require two contiguous 20′ locations.

Note that, according to a practice adopted by the majority of maritime companies, bays with even number are used for stowing 40′ containers and correspond to two contiguous odd bays that are used for the stowage of 20′ containers. Thus, let us denote $E \subset I$ for the subset of even bay and $O \subset I$ for the subset of odd bays. Two partitions are assumed for the set S of the stowage locations $l = (i, j, k)$: $S = A \cup P$, where A includes the locations of the anterior bays (towards the bow of the ship) and P the ones in the posterior bays (towards the stern of the ship), and $S = L \cup R$, where L includes the locations for the rows in the left side (the berth side), whereas R the ones in the right side of the ship.

We assume that the container handling operations are performed by cranes that are positioned on the quayside of the ship (quay cranes); therefore, the stowage of containers in the rows nearest to the seaside will be more time consuming.

When facing the MBPP as a mathematical programming problem, we first have to consider its basic combinatorial optimisation constraints, like the assignment constraints (AC) and the knapsack constraints (KC), requiring respectively, that (AC); each container to be loaded can be assigned at most to one location and each location of the ship can receive at most one container; (KC): the total weight of the containers to be loaded on the ship cannot be greater than the known total capacity Q of the ship. Moreover, we have some constraints related to the size, weight and destination of the containers to be loaded, and to the stability of the ship. For a detailed description of such constraints, the reader is referred to Ambrosino *et al* (2004).

The stability conditions that must be satisfied are related to the following three types of equilibrium: (a) the *horizontal equilibrium,* requiring that the weight of the left size of the ship (that is, quayside) should be equal, within a given tolerance, to the weight of the right side (that is, seaside); (b) the *cross equilibrium*, requiring that the weight of the anterior part of the ship (that is, centre-bow) should be equal, within a given tolerance, to the weight of the posterior part of the ship (that is, centre-stern); this is to avoid bending ($\cap$) or

saddling ($\cup$) of the ship; (c) the *vertical equilibrium*, requiring that the weight of each tier must be greater than or equal to the weight of the tier immediately over it. Note that the vertical equilibrium is guaranteed by the weight constraints, forcing the weight of the container located in a given location to be less than the weight of the container stowed in the lower tier, in the same row and bay.

A 0/1 Linear Programming (LP) model for the MBPP is presented in Ambrosino *et al* (2004); it can be used only for solving, up to optimality, very small instances, while for real-life instances often it is not possible to find an integer feasible solution after hours of computation. In fact, the MBPP is NP-hard (Avriel *et al*, 2000). Other integer programming models for solving the MBPP have been proposed in literature, as in Botter and Brinati (1992); Avriel and Penn (1993) and Imai *et al* (2002); unfortunately these papers also deal with simplified version of the MBPP and the proposed models are not suitable for real-life large-scale applications. A simplified model for the stowage problem that also takes into a proper account the flow of containers to be loaded on board from the yard is proposed in Ambrosino and Sciomachen (2003).

Note that according to the improved typology of cutting and packing problems reported in Wäscher *et al* (2007), the MBPP is classified as a three-dimensional (Orthogonal) bin packing problem (BPP); Sciomachen and Tanfani (2007) presented a heuristic approach for MBPP based on an exact algorithm for the three-dimensional BPP .

Other heuristics have been proposed for stowage planning. Dubrovsky *et al* (2002) used a genetic algorithm for minimising the number of container movements, while being able to include with appropriate constraints some ship stability criteria; they significantly reduce the search space using a compact and efficient encoding scheme and they obtain good solution for instances of 1000 TEUs ships. Wilson and Roach (1999) and Wilson *et al* (2001) tested local search techniques based on combinatorial optimisation. In particular, the authors used branch and bound algorithms for solving the problem of assigning generalised containers to a bays' block, and successively they found a detailed plan that assigns locations in a block to containers by a tabu search (TS) algorithm. The computational experiments reported by the authors showed the good quality of the sub-optimal solutions obtained in about 90 min for a 688 TEU ship. Finally, Wilson and Roach (2000) explored the potential of applying the theory of artificial intelligence to cargo stowage problems. A TS approach is also presented in Alvarez (2006) for determining loading plans, taking into account both the container reshuffling and the movements of the reach stackers.

After preliminary studies, we have noted that very computationally difficult constraints seem to be related to the destination of containers, stating that we

have to load first, that is, into lower tiers, those containers having as destination the last port in the route of the ship; analogously, we have to load last those containers having as destination the first port in the route of the ship. Other computationally difficult constraints are those related to the weight of the containers.

Ambrosino *et al* (2006) presented a heuristic approach consisting of a three-phase algorithm. In particular, in the first phase the ship is split into different portions and containers are associated with different subsets of bays according to their destination; note that in this way it is possible to remove the destination constraints from the model thus solving different single-destination models. Then the optimal stowage plan related to each partition of the ship is determined by solving the resulting 0/1 LP sub-problem for single destination. Finally, the global solution is determined by the union of the optimal solutions obtained in the second phase; its possible unfeasibility, because of the cross and horizontal stability constraints violation, is removed by performing some local search exchanges. More precisely, the algorithm looks for a profitable exchange by performing some moves that concern either a single container or a stack of containers at a time.

In this paper we face the MBPP by a new three-phase heuristic approach which extend the one which is outlined in Ambrosino *et al* (2006) as (1) a new 0/1 LP model is adopted for the solution of the single-destination problems and (2) the last phase is based on a TS algorithm (Reeves, 1993; Glover and Laguna, 1997), which allows us not only to convert the starting global solution into a feasible one but also to improve its objective value. Note that the 0/1 LP model, which is introduced here, considers equivalent classes of containers instead of single elements of set C, thus it is able to face medium and real large-size instances of the problem.

Finally, we stress that the power of operations research models and methods can be enhanced by incorporating them within a decision support system, which takes advantage of modern information technology. In particular, a prototype of a stowage planning supporting system is presented along with some computational results related to real case medium size instances provided by an import/export maritime container terminal located in the port of Genoa, Italy.

The Proposed Three-Phase Heuristic for the Master Bay Plan Problem

The proposed heuristic method for the MBPP is based on the following three successive phases, which are detailed in the following:

1 Bay assignment to containers according to their destinations.

2 Determination of a trial master bay plan by optimally solving the MBPP for single destination with limited ship stability conditions.
3 Determination of a whole feasible plan and global optimisation.

The overall algorithm description

Phase 1. The first phase, which is performed according to the *bay assignment algorithm* presented in Ambrosino *et al* (2006), determines a partition of set S of the available stowage locations in the ship into p subsets, each one consisting of a subset of bays that have to be used to stow containers towards a given destination. Note that this algorithm aims also at homogeneously distributing the load throughout the ship and consequently at maximising the handling operations that can be performed concurrently by the quay cranes.

Phase 2. The second phase produces a 'good' trial solution for the MBPP by optimally solving p single-destination stowage problems, here denoted as SD-MBPPs, starting from the bay assignment determined in phase 1. Note that this trial solution is obtained disregarding the cross stability condition and imposing the horizontal stability condition only for the involved subset of bays. With such simplifying assumptions each SD-MBPP can be formalised with a 0/1 LP model, which is described in detail in the next section.

Phase 3. The union of the p solutions of SD-MBPP problems produces a trial solution for the whole MBPP, which may fail to satisfy the overall ship stability conditions, that is, the horizontal and cross equilibrium. Then, in the third phase a TS meta-heuristic algorithm is adopted to determine both a globally feasible solution (that is, a stowage plan satisfying the ship stability conditions) and possibly to improve it (that is, to reduce the total loading time). The TS algorithm has been designed as an extension of the exchange algorithm based on a set of neighbourhood searches proposed in Ambrosino *et al* (2006).

The 0/1 linear programming model based on classes of container weights

We propose a new 0/1 LP model to solve the SD-MBPP such that the location decisions are not associated with the single containers but with sets of containers characterised by a weight within a given range (*weight classes*) and a specified type and destination. We devise such a problem formulation to conspicuously reduce the number of variables and constraints with respect to the model in Ambrosino *et al* (2006) in order to deal with containerships of medium or large size. In addition, the new model takes into account the presence of hatches between hold and deck locations; hatches, in fact, allow to separate the constraints relevant to the weight of containers in a stack (that is, locations with

the same bay and row) in the hold from the one for the same stack on the deck. As a consequence, this new model better fits into the actual stowage plans because it allows empty locations between the top of a stack in the hold and the container in the first location of the same stack on the deck. Since a generally adopted 'good' practice is that of favouring the location of lighter containers on deck to improve the ship stability, we also impose that the total weight located on the deck cannot be greater than the one located in the hold.

Before reporting here the proposed mathematical formulation of the SD-MBBP we must introduce some further notation.

Sets

G set of weight classes (for example, 1 = light, 2 = medium, 3 = heavy)

D set of locations in even bays of the hold ($i \in E,\ k = 1, \ldots, |K_H| - 1$) under which a single available odd location exists (also denoted as *irregular* even bay locations)

Constants

t_{ijk} loading time $\forall (i, j, k) \in S$

n_{g20}, n_{g40} number of 20′ and 40′ containers in group g, respectively, to be loaded

M_{20}, M_{40} maximum cumulative weight for a triple of containers of type 20′ and 40′, respectively

w_g weight of a container in group g

Q_1 maximum horizontal equilibrium tolerance.

Variables

$x_{ijkg} \in \{0, 1\}\ \forall i \in I,\ j \in J,\ k \in K,\ g \in G;\ x_{ijkg} = 1$ denotes that location (i, j, k) is assigned to a container of class g. Note that, as 40′ containers can be located only in $i \in E$ whereas 20′ containers only in $i \in O$; the container type is univocally determined by the assigned bay.

The SD-MBPP problem is formulated as follows:

$$\min L(x) = \sum_{ijkg} t_{ijk} x_{ijkg} \tag{1}$$

subject to

$$\sum_{ijk:i \in E} x_{ijkg} = n_{g40}, \quad \forall g \in G \tag{2}$$

$$\sum_{ijk:i\in O} x_{ijkg} = n_{g20}, \quad \forall g \in G \tag{3}$$

$$\sum_{g} x_{ijkg} \leqslant 1, \quad \forall i \in I, j \in J, k \in K \tag{4}$$

$$\sum_{g} (x_{i-1jkg} + x_{ijkg}) \leqslant 1, \quad \forall i \in E, j \in J, k \in K \tag{5}$$

$$\sum_{g} (x_{i+1jkg} + x_{ijkg}) \leqslant 1, \quad \forall i \in E, j \in J, k \in K \tag{6}$$

$$x_{ijk+1g} \leqslant \sum_{\substack{h\in G \\ h\geq g}} x_{ijkh}, \quad \forall i \in O,\ j,\ k = 1, \ldots, |K_H| - 1, \forall g \tag{7}$$

$$x_{ijkg} \leqslant \sum_{\substack{h\in G \\ h\geq g}} x_{ojk-1h}, \quad \forall (i,j,k) \in D, \forall g, o \in \{i-1, i+1\} : (o,j,k-1) \in S \tag{8}$$

$$x_{ijk+1g} \leqslant \sum_{\substack{h\in G \\ h\geq g}} x_{ijkh}, \quad \forall i \in E, j, k = 1, \ldots, |K_H| - 1, (i,j,k) \notin D, \forall g \tag{9}$$

$$x_{ijkg} \leqslant \sum_{\substack{h\in G \\ h\geq g}} x_{ijk-1h}, \quad \forall i, j, k = |K_H| + 2, \ldots, |K|, \forall g \tag{10}$$

$$\sum_{g} w_g (x_{ijkg} + x_{ijk+1g} + x_{ijk+2g}) \leqslant M_{20}, \quad \forall i \in O, j, k = 1 \tag{11}$$

$$\sum_{g} w_g (x_{ijkg} + x_{ijk+1g} + x_{ijk+2g}) \leqslant M_{40}, \quad \forall i \in E, j, k = 1 \tag{12}$$

$$\sum_{g} w_g (x_{ijk-2g} + x_{ijk-1g} + x_{ijkg}) \leqslant M_{20}, \quad \forall i \in O, j, k = |K_H| + 3 \tag{13}$$

$$\sum_{g} w_g (x_{ijk-2g} + x_{ijk-1g} + x_{ijkg}) \leqslant M_{40}, \quad \forall i \in E, j, k = |K_H| + 3 \tag{14}$$

$$\sum_{\substack{ijkg \\ k\notin K_H}} w_g x_{ijkg} \leqslant \sum_{\substack{ijkg \\ k\in K_H}} w_g x_{ijkg} \tag{15}$$

$$\left| \sum_{\substack{ijkg \\ j\in L}} w_g x_{ijkg} - \sum_{\substack{ijkg \\ j\in R}} w_g x_{ijkg} \right| \leqslant Q_1 \tag{16}$$

$$x_{ijkg} \in \{0, 1\}, \quad \forall i \in I, j \in J, k \in K, g \in G \tag{17}$$

Equation (1) defines the objective function as the total loading time. Constraints (2) and (3) impose that the required number of ship locations is assigned to each weight class and container type. Assignment constraints (4) ensure that any single ship slot is assigned at most once. Constraints (5) and (6) impose that if a

location in an even bay i is assigned to a 40′ container, then the two contiguous odd bay, $i-1$ and $i+1$, cannot be used or *vice versa*. The four sets of inequalities (7)–(10) are introduced to avoid the situation in which a container in a heavier class of weight is located directly over another belonging to a lighter class of weight (*pair weight condition*). Constraints (7), (8) and (9) ensure that this condition is satisfied in the locations in the hold taking into account for possible irregularity in the hold because of the shape of the keel of the ship, whenever below a location in an even bay there is only one of the two locations in the corresponding odd bays. In particular, (7) and (9) impose the pair weight, respectively, for odd bays (that is, for 20′ containers) and even bays (that is, for 40′ containers) in the hold, simply stating that an upper location can be assigned to a weight class only if the lower location has not been assigned to a lighter weight class, whereas inequalities (8) imposes that whenever an irregular even bay location is assigned to a 40′ container the single location in an odd bay below it is in turn assigned to a 20′ container with an appropriate weight class. As no irregularity is assumed for the locations on the deck, the single set of constraints (10) is used to impose the pair weight condition for them. Finally, note that the four sets of constraints (7)–(10) introduce a separation between hold and deck. Constraints (11)–(14) impose a maximum weight for any stack of three containers (*stack weight condition*). Noting that the pair weight condition ensures that the weight of container in any stack is non-increasing, the stack weight condition is satisfied by imposing it for any fixed bay and row only for the first stacks of three containers in the hold (from tier $k=1$ to $k=3$) and on the deck (from $k=|K_H|+1$ to $k=|K_H|+3$); in particular, (11) and (12) take into account stacks, respectively, of 20′ and 40′ container in the hold, whereas (13) and (14) on the deck. The separation between hold and deck locations because of the presence of hatches (stated by both pair and stack weight conditions) allows us not to fill the bays in the hold before assigning the corresponding bays on the deck; however, taking into account that loading times associated with deck locations are usually smaller than the ones for the hold, this fact could lead us to generate plans with an unacceptable mass distribution. In order to prevent such a drawback, we introduce inequality (15) to impose that the total weight located on the deck cannot be greater than the one located in the hold. Constraint (16) states the horizontal equilibrium stability condition for the subset of containers with the considered destination, and finally (17) is the definition of the variables.

The solution of a SD-MBPP for a destination d is then converted into an actual single-destination stowage plan by a straightforward container assignment procedure: first the subset of container directed to d is clustered into the considered classes of weight, and then the containers in each cluster are located into the stowage locations assigned to the corresponding class of weight by the

SD-MBPP solution. Finally, the union of the single-destination plans so thus determined produces the whole trial solution for the MBPP used as starting solution in the last phase of the heuristic algorithm.

The tabu search algorithm

The third phase of the proposed heuristic consists of a TS procedure to iteratively modify the starting MBPP solution by applying a sequence of *moves* in order both to determine a solution satisfying global stability conditions and to possibly improve its objective value. The TS algorithm (Reeves, 1993; Glover and Laguna, 1997) extends the Local Search (LS) algorithm for combinatorial problems. Differently from LS, which iterates the generation of a sequence of modified solutions until no more moves improving the problem objective can be found, TS accepts also worsening moves to avoid a too early termination in a local optimum. In particular, TS uses a *first-in-first-out* list of forbidden moves, called *tabu list* (TL), which provides the algorithm with a short-term memory to avoid cycling during the search process; the number of positions of the TL, called *tabu tenure*, specifies the number of iterations that must be executed since the last time a move has been used to modify the solution, before that move can be applied again. This rule, however, can be overridden by specifying a so-called *aspiration criterion* that, for example, accepts tabu moves producing an improved solution.

We consider for the TS the objective function $Z(x) = M(\sigma_1(x) + \sigma_2(x)) + L(x)$, which includes not only the horizontal and cross equilibrium stability violation functions, respectively $\sigma_1(x)$, $\sigma_2(x)$, introduced in Ambrosino *et al* (2006), but also the total loading time $L(x)$ of a solution x. Note that we introduce in $Z(x)$ the M coefficient, such that $M \gg 0$, to strongly penalise the stability violation functions, so giving priority to the generation of feasible solutions. We use a general set of moves (that is, neighbourhood structures) in order to modify a solution generating neighbouring solutions. In particular, we define seven classes of moves, combining three kinds of items that can be moved, that is, a single container, a stack of containers and a bay, with three kinds of position exchanges, that is, anterior–posterior location exchange, left–right side exchange and cross exchange. Note that one of the two items involved in a move can correspond to one or more empty locations.
The seven classes of moves are detailed as follows:

1 *Anterior–Posterior exchange of containers*: This kind of move exchanges the current positions $l \in A \cap X$ and $l' \in P \cap X$ of two containers c and c', X being a fixed side of the ship; note that this change may affect the cross equilibrium but does not modify the horizontal equilibrium of the ship.

2 *Anterior–Posterior exchange of stacks*: This move exchanges the positions of two whole stacks of containers, s and s', where $s = \{l\colon l \in A \cap X,\ l = (i, j, k)$ with i and j fixed$\}$ and $s = \{l\colon l \in P \cap X,\ l = (i', j', k)$ with i' and j' fixed$\}$, X being a fixed side of the ship, that is, two stacks currently located one in an anterior bay and the other in a posterior bay, but both in the same left or right side of the ship; again these moves may affect only the cross equilibrium.
3 *Left–Right side exchange of containers*: Two containers c and c' located, respectively, in $l \in L$ and $l' \in R$ are exchanged; this move may affect the horizontal equilibrium whereas it does not modify the cross one.
4 *Left–Right side exchange of stack*: As for move 2, two stacks of containers s and s' made of location, respectively, belonging to the left and right sides of the ship are exchanged; analogously to the above move only the horizontal equilibrium may be affected.
5 *Cross exchange of containers*: This kind of move exchanges current positions $l \in A \cap L$ (or $l \in A \cap R$) and $l' \in P \cap R$ (or $l' \in P \cap L$) of two containers c and c', hence affecting both the horizontal and the cross equilibrium.
6 *Cross exchange of stack*: Similarly to the previous moves, in this case two stacks s and s' of containers are exchanged, whose locations are, respectively, in $A \cap L$ (or $A \cap R$) and in $P \cap R$ (or $P \cap L$).
7 *Anterior–Posterior exchange of bays*: This move exchanges all the containers in two bays i and i', respectively, located in A (or P) and P (or A) without changing the original row and tier positions of the containers; for this reason, these moves may affect only the cross equilibrium.

Note that any of the above moves always involves containers with the same destination. Three different TLs are initialised; in particular, a separate TL is considered for each type of items affected by the seven classes of moves previously described, that is, single containers, stacks and bays. Note that each TL can have a different *tabu tenure*, which is an input parameter of the algorithm. The TS algorithm executes a cycle of *external* iterations until a termination condition, corresponding to a maximum number of external iterations or a maximum number of external iterations without improvement, is met. Each external iteration is in turn made of a sequence of neighbourhood explorations that are executed on the different neighbourhood structures defined by the seven classes of moves introduced. This kind of behaviour tries to exploit the variable neighbourhood search proposed in Mladenovic and Hansen (1997) rationale, according to which local optimisers may depend on the explored neighbourhood structure, in order to help the TS to escape from local optima and to diversify its exploration. Even in this case the search corresponds to an *internal* iteration cycle performed until a maximum number of iterations or

iterations without improvement is reached. During an internal search iteration, a sequence of feasible candidate solutions is generated in the neighbourhood $N(x)$ of the current solution x (note that feasibility tests are used to select moves not violating both destination and weight constraints); a candidate solution x' is allowed either if it is produced by a move not included in a TL (tabu move) or if x' satisfy the adopted aspiration criterion. In particular, we adopt the so-called *best objective* criterion that is satisfied by any candidate solution x' generated by a tabu move such that $Z(x') < Z(x^{\circ})$, where x° is the current best solution found by the TS algorithm. Note that a candidate solution is feasible if it satisfies all the assignment and weight constraints (2)–(10) in the SD-MBPP problem formulation.

The exploration of the neighbourhood $N(x)$ of the current solution during an internal iteration is completed according to the *first improvement* selection criterion: as soon as an allowed candidate solution x' is found, such that $Z(x') < Z(x)$, this candidate becomes the new current solution; alternatively, if none of the allowed candidate solutions generated improves the current one, the candidate x'' such that

$$x'' = \arg\min_{x' \in N(x) \setminus \{x\}} Z(x') \tag{18}$$

is chosen as new current solution. Note that the neighbourhood exploration is not complete but the candidate solutions are generated by a random extraction of the moves. Besides, the TL is updated removing the move, if any, in the last position of the list, then increasing by one the positions of the other moves currently in the list and inserting the move selected to update the current solution in the first position of the list. Finally, after a given number of iterations without improvements, a diversification procedure tries to escape from a local minimum performing a number of feasible swap moves, disregarding the cost that they produce. After this procedure the TS exploration restarts normally.

A Master Bay Planning Support System

In order to provide the terminal operator with a tool allowing an easy and friendly inspection of the stowage plan generated by our three-step heuristics, we developed a support system for the master bay planning activity, which is based on an interactive graphic user interface. In Figures 1 and 2 we provide an example of the kind of information that the developed *master bay planning support system* (MBPSS) is able to provide. Figure 1 shows the main stowage

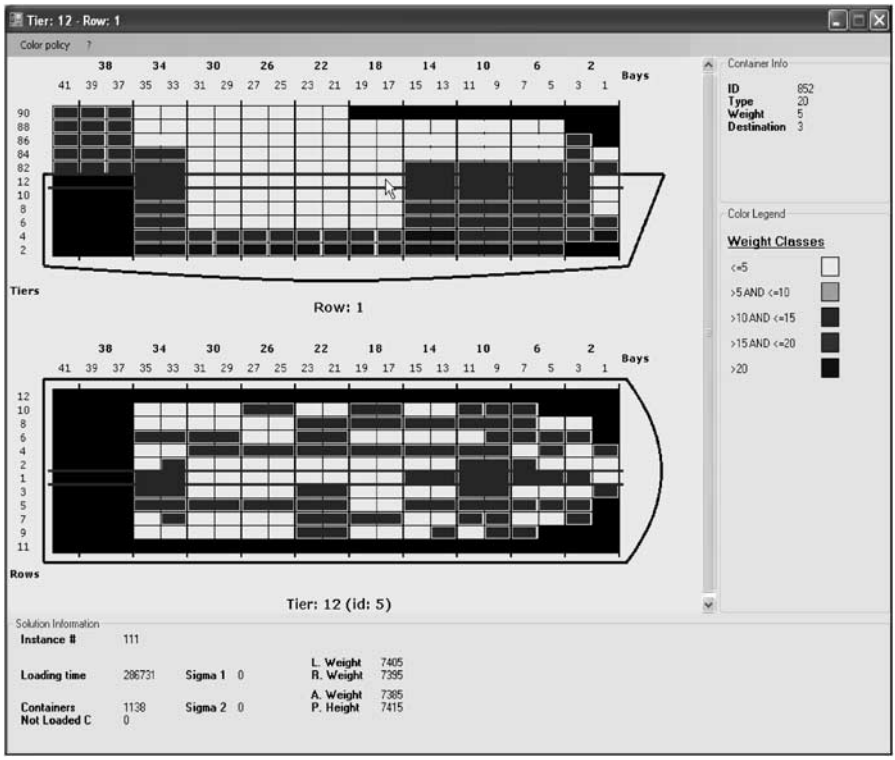

Figure 1: An example of the container weight view of a master bay plan for the European Senator containership provided by the MBPSS.

views produced by the system, which is the row view in the upper part of the interface and the tier view in the lower one; the terminal operator can inspect a plan as follows:

1 the operator can select the desired row and tier with the mouse or keyboard; for example, Figure 1 shows the containers located in row 1 (row view) and in tier 12 (tier view) for all the bays, whereas different combinations of row and tier are selected in the two images of Figure 2;
2 the operator can observe the container distribution according to three alternative views, that is, destinations (Figure 2a), container types (Figure 2b), classes of weight (Figure 1); note that the meaning of the used colour scheme is highlighted in the *Colour legend* box of the interface;
3 the operator can retrieve information for the container pointed by the mouse; in images of Figures 1 and 2 the data for the selected container are reported in the upper-right box, *Container info*, of the interface.

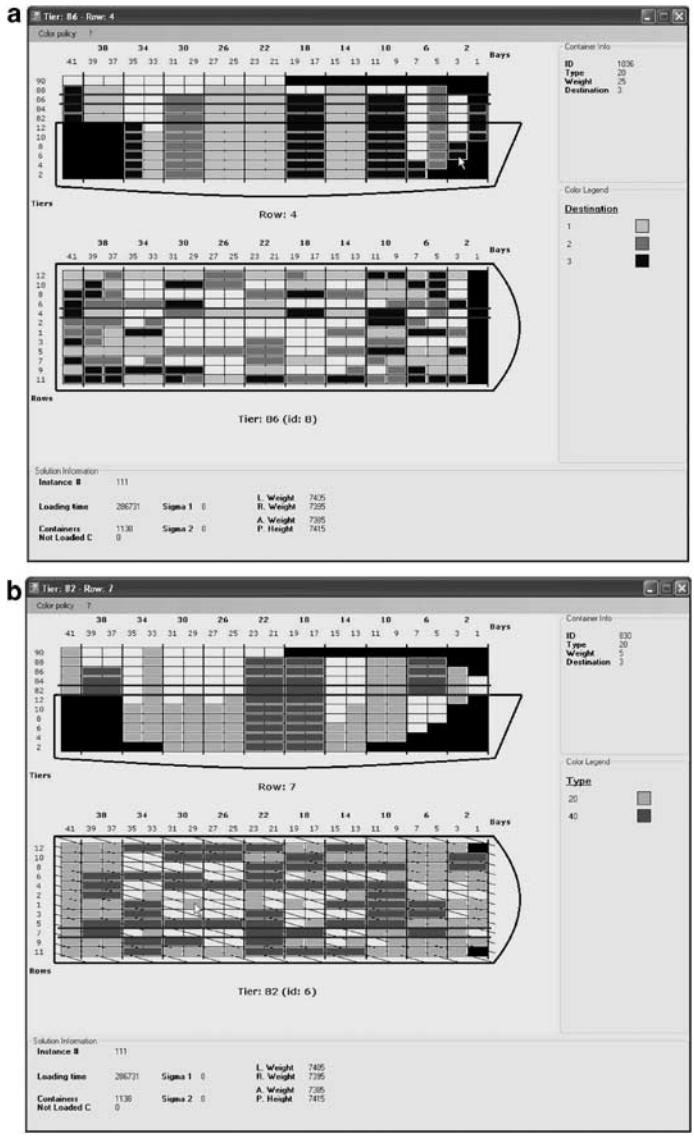

Figure 2: An example of the master bay plan inspection facilities for the European Senator containership provided by the MBPSS: (a) the destination view of a master bay plan and (b) the container type view of a master bay plan.

Finally, in the lower box of the interface, *Solution information*, we provide aggregate data characterising the showed plan, as the loading time and the total weight for the left (L), right (R), bow (A) and stern (P) portions of the ship.

Experimental Results

The proposed MBPP algorithm was coded in C++, using the commercial Cplex 9.0 as 0/1 LP solver, and tested on a 1.5GHz, Intel Celeron PC with 1Gb RAM. In particular, the tests were related to two containerships of different size whose data have been provided by the SECH Terminal of Genova, Italy. A first set of problem instances concerns a small size containership, named Chiwawa, with a maximum capacity of 198 TEU, composed of 11 odd bays, 4 rows and 5 tiers (3 in the hold and 2 in the upper deck, respectively). Table 1 reports the characteristics of the considered 10 small size test instances, showing the total number of containers, both in *TEU* and absolute number (#), the number of 20′ and 40′ containers, the number of containers for three classes of weight (l = low, m = medium, h = high), the partition of containers for each destination and finally the level of occupancy of the ship (*Full*). Table 2 shows the loading times for the Chiwawa containership; in this case loading times, depending on the row and tier and growing from left side rows to the right ones and from highest tiers on the deck to the lowest ones in the hold, have been assumed.

Table 1: The small size test instances

Instance	*TEU*	#	*Type* (#)		*Weight* (#)			*Destination* (*TEU*)						*Full* (%)
			20′	*40′*	*L*	*M*	*H*	*1*	%	*2*	%	*3*	%	
1	138	100	62	38	46	50	4	47	47.00	53	53.00	—	—	73.4
2	165	120	75	45	52	64	4	55	45.83	65	54.17	—	—	87.7
3	170	130	90	40	60	66	4	62	47.69	68	52.31	—	—	90.0
4	175	130	85	45	58	68	4	62	47.69	68	52.31	—	—	93.0
5	180	140	100	40	62	74	4	61	43.57	79	56.43	—	—	98.4
6	180	150	120	30	70	76	4	65	43.33	85	56.67	—	—	98.9
7	185	130	75	55	60	66	4	62	47.69	68	52.31	—	—	95.7
8	185	140	95	45	58	78	4	65	46.43	75	53.57	—	—	98.4
9	185	140	95	45	62	73	5	50	35.71	40	28.57	50	35.71	100
10	188	148	108	40	68	76	4	50	33.78	50	33.78	48	32.43	95.7

Table 2: Loading times for the Chiwawa containership (times are in 1/100 of minute)

Tier	*Row*			
	3	*1*	*2*	*4*
2	240	250	260	270
4	230	240	250	260
6	220	230	240	250
82	210	220	230	240
84	200	210	220	230

We show the results obtained for the set of small size instances in Table 3, where objective values are given in 1/100 of minute and CPU times in seconds; the results are divided into three groups of columns relevant to the solutions found, respectively, by the exact 0/1 LP model after 1 hour of computation, by the SD-MBPP 0/1 LP model in Ambrosino *et al* (2006) with an absolute gap (that is, the difference between the incumbent integer solution and the lower bound) equal to 1 min as termination condition, and by the TS algorithm. Note that, as we adopted a stochastic TS, the *Avg obj* column reports the average results obtained over five independent runs of our TS. The *%dev* from (a) and *%dev* from (b) columns quantify the relative variation of the average TS results, respectively, from the exact 0/1 LP solutions and the SD-MBPP ones; the last column (*Cumul Time*) indicates the total (average) CPU time needed by the SD-MBPP and TS phases, which apparently are the most time demanding phases of the proposed approach. Table 3 highlights the effectiveness of the approach, showing that globally feasible solutions can be produced in a very short computational time (24.5 seconds on the average) with a very reduced average optimality gap (0.17 per cent), corresponding to the average percentage difference from the solution produced after 1 hour of computation by the exact complete model for the MBPP, which includes both destination and global stability constraints.

However, we performed a further analysis relaxing the termination condition for the 0/1 LP solver; this test revealed that the exact 0/1 LP model needed only 26.1 seconds average CPU time to produce solutions with, on the average, a 0.08 per cent gap from the ones obtained after 1 hour of computation. This fact pointed out that actually very small size containerships do not represent a challenging test for the proposed approach.

Then, a second set of 14 instances was considered for a medium size containership, the European Senator, with a 2124 TEU capacity, composed of 17 odd bays, 10 rows and 6 tiers in the hold and 21 odd bays, 12 rows and 5 tiers in the upper deck.

We report in Table 4 the characteristics of the medium size instances and in Table 5 the corresponding loading times as done for Tables 1 and 2. Note that this second test highlighted the need of considering weight groups instead of single containers because in the latter case the 0/1 LP solver was not able to find any integer solution after hours of computation. Nevertheless, as we highlight in the following, the medium size instances remain quite hard. We fixed termination condition as an absolute gap $= 5Nd$ min, where Nd is the number of destinations of the considered instance, and a maximum time limit of 1 hour. Then, we found that for all the four instances with three destinations no integer solution was found in 1 hour of computation, for five instances over ten with two destinations the solver stopped after reaching the maximum time

Table 3: The comparison of the results for the small size instances

Instance	*Exact 0/1 LP (max time=1hour)*		*SD-MBPP (absolute gap=1 min)*					*Tabu search (average over five runs)*				
	Obj (a)	*Time*	*Obj (b)*	*%dev*	σ_1	σ_2	*Time*	*Avg obj*	*%dev from (a)*	*%dev from (b)*	*Avg time*	*Cumul. time*
1	22 270	3602.2	22 340	0.31	0	180	3.6	22 320	0.22	−0.09	12.30	15.9
2	27 030	3602.7	27 100	0.26	0	130	5.0	27 110	0.30	0.04	36.24	41.3
3	29 320	3603.0	29 400	0.27	0	75	6.4	29 360	0.14	−0.14	14.56	21.0
4	29 370	3603.0	29 480	0.37	0	205	5.8	29 390	0.07	−0.31	11.29	17.1
5	31 720	3603.4	31 770	0.16	15	260	8.6	31 750	0.09	−0.06	15.58	24.2
6	34 060	3603.9	34 150	0.26	0	55	8.4	34 150	0.26	0.00	11.82	20.2
7	29 480	3602.8	29 550	0.24	0	275	6.3	29 530	0.17	−0.07	12.17	18.5
8	31 790	3603.3	31 930	0.44	0	0	6.5	31 840	0.16	−0.28	11.64	18.1
9	31 790	3603.3	31 910	0.38	0	25	3.5	31 810	0.06	−0.31	14.78	18.3
10	33 670	3603.6	33 860	0.56	10	5	8.1	33 740	0.21	−0.35	42.17	50.3
			Averages	0.33			6.2		0.17	−0.16	18.3	24.5

Table 4: The medium size test instances

Instance	*TEU*	*#*	*Type (#)*		*Weight (#)*			*Destination (TEU)*						*Full (%)*
			20′	*40′*	*L*	*M*	*H*	*1*	*%*	*2*	*%*	*3*	*%*	
1	945	715	485	230	215	429	71	463	48.99	482	51.01	—	—	52.50
2	1022	762	502	260	228	458	76	501	49.02	521	50.98	—	—	56.78
3	1120	820	520	300	246	492	82	549	49.02	571	50.98	—	—	62.22
4	1218	898	578	320	270	541	87	600	49.26	618	50.74	—	—	67.67
5	1320	980	640	340	295	589	96	450	34.09	514	38.94	356	26.97	73.33
6	1380	1090	800	290	327	654	109	676	48.99	704	51.01	—	—	76.67
7	1386	984	582	402	296	590	98	680	49.06	706	50.94	—	—	77.00
8	1415	1069	723	346	321	642	106	481	33.99	524	37.03	410	28.98	78.61
9	1420	1060	700	360	318	636	106	696	49.01	724	50.99	—	—	78.89
10	1528	1202	876	326	361	721	120	749	49.02	779	50.98	—	—	84.89
11	1522	1138	754	384	341	683	114	524	34.43	586	38.50	412	27.07	84.56
12	1627	1215	803	412	365	729	121	804	49.42	823	50.58	—	—	90.39
13	1724	1331	938	393	400	799	132	588	34.11	669	38.81	467	27.09	95.78
14	1800	1413	1026	387	424	848	141	882	49.00	918	51.00	—	—	100.00

limit, whereas for the remaining five instances with two destinations the solver required on the average 16 min and 47 seconds to terminate; it is important to underline that trying to find out an exact solution for the instances with three destinations appeared significantly harder than solving the ones with two destinations, so this supports the need of an effective heuristic approach for medium or large containerships. We can analyse the behaviour of the 0/1 LP solver with the exact MBPP model considering the two plots in Figure 3, where we report the relative gap (that is, the percentage difference between the best integer solution and the lower bound) variation during the solution process, having fixed 3 hours as maximum time limit and not imposing any gap stopping criterion. Figure 3a shows the medium-sized instance 4 with two destinations, which in our tests was solved after 26 min 25 seconds because of the absolute gap criterion: the figure points out that after an extension of computation time of 2 hours 33 min 35 seconds the absolute gap is not significantly reduced and the relative percentage improvement of the first integer solution found (−0.20per cent) is negligible. In Figure 3b we can observe the solver behaviour, for instance 5 with three destinations not solved within the 1 hour time limit: for such an instance, the first integer solution was found after 1 hour 18 min 14 seconds, but even after 3 hours the absolute gap (25 min 33 seconds) would not be sufficient to stop the solver (note that also in this case a negligible −0.34 per cent relative percentage improvement was measured after 3 hours).

The comparison of the results produced by the exact model and the solution of the SD-MBBP problems, that is, after the first and second phases of the proposed approach, is reported in the first nine columns of Table 6. An absolute

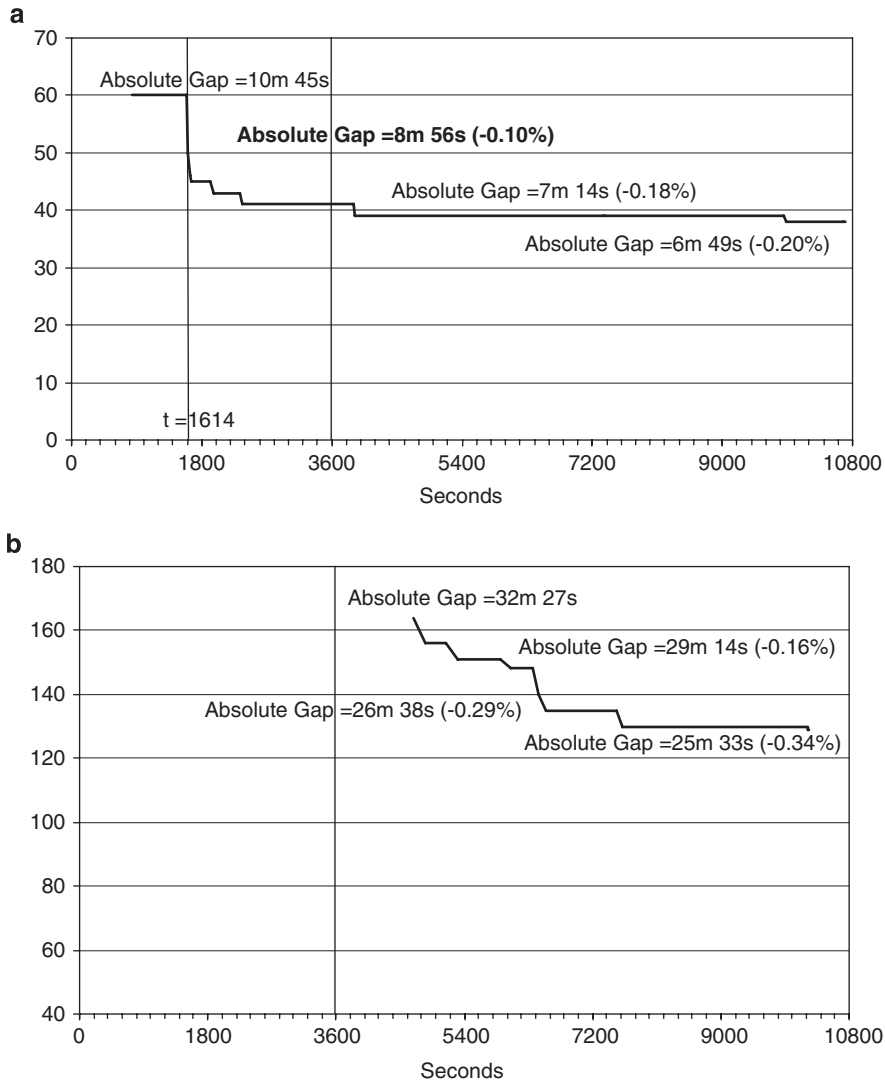

Figure 3: The variation of the relative gap for two medium size instances provided by the 0/1 LP solver with the exact MBPP model: (a) medium size instance 4 (two destinations) and (b) medium size instance 5 (three destinations).

gap = 5 min was fixed as termination condition for the SD-MBPP model and the loading times shown in the columns *Obj* are expressed as 1/100 of minute. Table 6 indicates that the solutions obtained by the SD-MBPP are worse but not very far from the ones yielded by the exact 0/1 LP model (the average

Table 5: Loading times for the European Senator containership (times are in 1/100 of minute)

Tier	*Row*											
	11	*9*	*7*	*5*	*3*	*1*	*2*	*4*	*6*	*8*	*10*	*12*
2	220	230	240	250	260	270	280	290	300	310	320	330
4	210	220	230	240	250	260	270	280	290	300	310	320
6	200	210	220	230	240	250	260	270	280	290	300	310
8	190	200	210	220	230	240	250	260	270	280	290	300
10	180	190	200	210	220	230	240	250	260	270	280	290
12	170	180	190	200	210	220	230	240	250	260	270	280
82	160	170	180	190	200	210	220	230	240	250	260	270
84	150	160	170	180	190	200	210	220	230	240	250	260
86	140	150	160	170	180	190	200	210	220	230	240	250
88	130	140	150	160	170	180	190	200	210	220	230	240
90	120	130	140	150	160	170	180	190	200	210	220	230

percentage deviation is 3.41 per cent, which corresponds to an average difference in the total loading time = 1 hour 2 min 21 seconds) and the average CPU time needed is very short (37.9 seconds); however, the stability conditions (especially the horizontal one) are never satisfied. Table 6 also reports that in the last five columns the results are obtained after the execution of the TS algorithm from the starting solutions provided by the SD-MBPP.

We performed some preliminary tests to select suitable values for the TS parameters, finally determining the following configuration: tenure = 80, diversification after 30 non-improving iterations, diversification length = 35 iterations, termination condition corresponding to maximum number of iterations = 500 and maximum not improving iterations = 50. As random choices are used in the proposed TS, five independent runs were performed for each instance and then the average results were considered. In particular, Table 6 shows the average total loading time (*Avg obj*), as well as the percentage deviations from the objective produced by the exact 0/1 LP model (%*dev* from (a)) and by the SD-MBPP (%*dev* from (b)), the CPU time for the TS phase and the cumulative one (SD-MBPP + TS time). Note that the TS was able to obtain solutions satisfying both the cross and the horizontal stability conditions on every run for each instance. This second set of results clearly highlights the quality of the proposed approach for the MBPP: the solutions provided in 74.7 seconds cumulative average CPU time are not only always globally feasible but they improve on the average ones yielded by the SD-MBPP (−1.90 per cent) so that they are only 1.33 per cent worse than the ones because of the exact 0/1 LP model after a much longer computation (corresponding to an average difference in the total loading time = 24 min 49 seconds). We also tested the TS algorithm

Table 6: The comparison of the results for the medium size instances

Instance	*Nd*	*Exact 0/1 LP (max time=1hour, absolute gap=5Nd min)*		*SD-MBPP (absolute gap=5 min)*					*Tabu search (average over five runs)*				
		Obj (a)	*Time*	*Obj (b)*	*%dev*	σ_1	σ_2	*Time*	*Avg obj*	*%dev from (a)*	*%dev from (b)*	*Avg Time*	*Cumul. Time*
1	2	139 530	1300.8	150 570	7.91	15	5380	68.4	142 990	2.48	−5.03	32.7	101.1
2	2	149 440	1606.4	155 100	3.79	20	1065	11.6	151 700	1.51	−2.19	32.2	43.8
3	2	161 670	1876.7	168 310	4.11	0	1225	7.4	164 444	1.72	−2.30	33.1	40.5
4	2	178 320	1613.3	186 310	4.48	0	1435	21.8	180 906	1.45	−2.90	32.5	54.3
5	3	—	3600.0	206 590	—	20	5685	65.0	200 140	—	−3.12	54.6	119.6
6	2	219 650	3698.6	225 510	2.67	0	1115	17.6	222 600	1.34	−1.29	40.5	58.1
7	2	197 410	1999.8	203 040	2.85	0	1555	18.0	200 072	1.35	−1.46	31.7	49.8
8	3	—	3600.0	220 210	—	15	1300	7.5	217 660	—	−1.16	38.0	45.5
9	2	213 520	3698.5	219 330	2.72	0	1235	12.1	216 036	1.18	−1.50	39.9	52.0
10	2	244 660	3698.4	252 640	3.26	0	1495	21.3	247 810	1.29	−1.91	42.2	63.5
11	3	—	3600.0	237 050	—	0	1625	19.8	233 608	—	−1.45	46.1	65.9
12	2	248 050	3295.0	253 640	2.25	5	1010	212.3	250 128	0.84	−1.38	31.1	243.4
13	3	—	3600.0	278 780	—	0	470	30.8	275 964	—	−1.01	35.9	66.7
14	2	293 150	3704.7	293 310	0.05	0	1960	17.0	293 530	0.13	0.08	24.5	41.5
		Averages	2649.2		3.41			37.9		1.33	−1.90	36.8	74.7

in order to analyse its behaviour with both a faster and a slower alternative configurations. In particular, we performed five TS runs with maximum number of iterations = 10 (without diversification), obtaining the results in an average CPU time = 0.9 seconds but with a loading times 3.10 per cent worse on the average than the ones of the exact 0/1 LP model (corresponding to an increase in loading time of 58 min 6 seconds); finally, five TS runs with maximum number of iterations = 1000, non-improving iterations = 200 and diversification after 100 non-improving iterations, produced in an average CPU time = 84.1 seconds results 1.17 per cent worse (that is, an average increase in loading time = 21 min 42 seconds) than the ones of the exact 0/1 LP model.

Conclusions

In this paper we have proposed a three-step heuristic for solving the MBPP for containership of medium-large size instances, which includes a new 0/1 LP model for the single-destination problem.

The proposed three-phase approach is able to obtain feasible master bay plans for the challenging medium size containership in a short average CPU time (74.7 seconds), which require on an average only 24 min 49 seconds more than the 'almost' optimal loading time. In addition, the NP-hardness of the problem as well as the observed increase in the computation time required by the exact model, when compared with the one needed by the proposed approach, appears to indicate this latter as appropriate to face the MBPP for containership of medium-large size.

The effectiveness of the proposed optimisation method is enhanced by a decision support system for stowage planning, which includes the implemented algorithms and displays the obtained solutions for each bay, row and tier of the ship.

Acknowledgement

This work has been developed within the research project 'Modelli di Programmazione lineare intera e tecniche Euristiche per problemi inerenti alla gestione di un terminal container' of the University of Genoa, Italy.

References

Alvarez, J.K. (2006) A heuristic for vessel planning in a reach stacker terminal. *Journal of Maritime Research* 3(1): 3–16.

Ambrosino, D. and Sciomachen, A. (2003) Impact of a yard organisation on the Master Bay Planning Problem. *Maritime Economics & Logistics* 5: 285–300.
Ambrosino, D., Sciomachen, A. and Tanfani, E. (2004) Stowing a containership: The Master Bay Plan problem. *Transportation Research A* 38: 81–99.
Ambrosino, D., Sciomachen, A. and Tanfani, E. (2006) A decomposition heuristics for the container ship stowage problem. *Journal of Heuristics* 12(3): 211–233.
Avriel, M. and Penn, M. (1993) Exact and approximate solutions of the container ship stowage problem. *Computer and Industrial Engineering* 25(1–4): 271–274.
Avriel, M., Penn, M. and Shpirer, N. (2000) Container ship stowage problem: Complexity and connection to the colouring of circle graphs. *Discrete Applied Mathematics* 103: 271–279.
Botter, R.C. and Brinati, M.A. (1992) Stowage container planning: A model for getting an optimal solution. *IFIP Transactions B* B-5: 217–229.
Dubrovsky, O., Levitin, G. and Penn, M. (2002) A genetic algorithm with compact solution encoding for the container ship stowage problem. *Journal of Heuristics* 8: 585–599.
Glover, F. and Laguna, M. (1997) *Tabu Search*. London: Kluwer Academic Publishers.
Imai, A., Nishimura, E., Papadimitriu, S. and Sasaki, K. (2002) The containership loading problem. *International Journal of Maritime Economics* 4: 126–148.
Mladenovic, N. and Hansen, P. (1997) Variable neighbourhood search. *Computers & Operations Research* 24: 1097–1100.
Peters, J.H. (2001) Developments in global seatrade and container shipping markets: Their effect on port industry and private sector involvement. *International Journal of Maritime Economics* 3: 3–26.
Reeves, C.R. (1993) *Modern Heuristic Techniques for Combinatorial Problems*. New York: John Wiley & Sons.
Sciomachen, A. and Tanfani, E. (2007) A 3DD packing approach for optimising stowage plans and terminal productivity. *European Journal of Operational Research* 83(3): 1433–1446.
Stahlbock, R. and Voss, S. (2008) Operations research at container terminal: A literature update. *OR Spectrum* 30: 1–52.
Steenken, D., Voss, S. and Stahlbock, R. (2004) Container terminal operation and operations research – A classification and literature review. *OR Spectrum* 26: 3–49.
Thomas, B.J. and Roach, D.K. (1989) *Management of Port Maintenance: A Review of Current Problems and Practices*. London: H.M.S.O.
Wäscher, G., Haussner, H. and Schumann, H. (2007) An improved typology of cutting and packing problems. *European Journal of Operational Research* 83(3): 1109–1130.
Wilson, I.D. and Roach, P. (1999) Principles of combinatorial optimisation applied to container-ship stowage planning. *Journal of Heuristics* 5: 403–418.
Wilson, I.D. and Roach, P. (2000) Container stowage planning: A methodology for generating computerised solutions. *Journal of the Operational Research Society* 51(11): 248–255.
Wilson, I.D., Roach, P. and Ware, J.A. (2001) Container stowage pre-planning: Using search to generate solutions, a case study. *Knowledge-Based Systems* 14(3): 137–145.