International Journal of Maritime Economics, 2002, 4, 126-148
© 2002 Palgrave. All rights reserved. 1388-1973/02 $15
www.palgrave-journals.com/ijme

The Containership Loading Problem

AKIO IMAI[1], ETSUKO NISHIMURA[1],
STRATOS PAPADIMITRIOU[2] & KAZUYA SASAKI[1]

[1]Department of Transportation and Information Systems Engineering, Kobe University of Mercantile Marine, Fukae, Higashinada, Kobe 658-0022 Japan; [2]Department of Maritime Studies, University of Piraeus, 40 Karaoli & Dimitriou Str. GR-185 32 Piraeus, Greece. E-mail: stratos@ath.forthnet.gr

The efficiency of a container terminal depends primarily on the smooth and orderly process of handling containers, especially during the ship's loading procedure. The loading plan is mainly determined by two considerations: ship stability and minimum number of rehandles required. These two basic considerations are often in conflict. Most containerships have a cellular structure, imposing a strong restriction on the order of the container loading sequence. To preserve a ship's stability, some containers may be stowed in a ship hold in middle vertical locations. A similar loading problem exists in the stacking of yard containers. If these containers are stacked in the yard under others which are to be picked up later, then the loading process requires a number of container rehandles. This paper is concerned with a ship's container load planning which satisfies these two considerations and minimises the number of rehandles.

International Journal of Maritime Economics (2002) **4**, 126-148.
doi:10.1057/palgrave.ijme.9100041

Keywords: Container transportation; load planning; ship stability; container rehandling.

INTRODUCTION

The overwhelming majority of general cargo is nowadays containerised. Given that the containerised transportation system is capital-intensive, fast turnaround times in a container terminal are essential for the economic performance of liner shipping companies. Shortening the transit time of containers in the yard results in faster turnaround times of ships and consequently reduces the overall transit time of the entire transport chain, which had prompted the introduction of the container transportation system in the first place. The turnaround time of a ship includes the time for berthing, unloading, loading, and departure. The major activities affecting the turnaround time are the unloading and loading processes. While there is a relationship between the two processes, they are substantially carried out as two independent tasks with loading being the more difficult and sensitive to the efficiency of the operation. This paper develops an algorithm for efficient container ship load planning.

The efficiency of the loading operation depends primarily on the loading sequence of the containers. Planning an efficient load sequence is not easy: Most container ships feature a cellular structure, designed for improving the container stowage function, which however imposes a strong restriction on the order of the loading sequence of the containers to be handled. If, for example, specific containers (referred to as target containers) must be stowed at vertically middle locations in a ship's hold for ship stability reasons, they have to be loaded in a loading sequence after the containers that are to be stowed under them and before the containers that are to be stowed above them. Concurrently, another restriction emerges during the picking of containers from a yard to be loaded onto the ship, since containers are piled up to form block formations in the yard for storage purposes. If the target containers are stacked in the yard under others which are to be picked up later, then the loading task requires the so-called 'rehandle' in order to remove them and reposition them. This is very likely to occur, for detailed information about the order of the loading sequence is not available when containers start to arrive at the terminal. Furthermore, even when the loading information is available, the ideal layout of export containers in the storage area of the yard is almost impossible to be achieved due to the random arrival of containers. Therefore, satisfying the constraints of good ship stability as well as the reduction in the number of rehandles is a difficult task, given that these constraints are often conflicting.

A way to avoid rehandles during a loading operation would be container shuffling in advance of loading. However, this necessitates additional workload for the handling equipment, which is usually more intense than the total workload of rehandling during the loading operation. This could be done only when the handling equipment is idle. In addition, smooth shuffling requires a buffer stacking area, where containers to be loaded are moved orderly from the storage area. However, such a buffer area is hardly practical or realistic for container terminals in land scarce countries.

Such a rehandle problem may not be as important in big container terminals where very large vessels call, since there is a large number of containers to be handled in terms of total number, type and weight. In this particular case, load masters may find the right containers to load without a prior rehandle and in the proper load sequence onto a ship at stable locations. However, the aforementioned rehandling problem is typical for a relatively small size container terminal with medium size calling vessels. In this case, the rehandle is unavoidable due to the small number and variety of containers. Furthermore, terminal space is generally limited and not enough for the establishment of shuffling procedures with a buffer area.

This paper focuses on loading operations in comparatively low volume handling terminals and aims at the development of an algorithm maximising ship stability while minimising the number of container rehandles. The paper is organised as follows. The next section reviews the related literature. In the third section the proposed algorithm is described. In the subsequent section, a variety of numerical experiments are carried out and presented, and the final section reports the paper's conclusions.

LITERATURE REVIEW

Studies on container terminal efficiency can be classified in three distinct albeit interlinked categories. One category deals with the scheduling problem for handling equipment such as quay cranes, transfer cranes (or transtainers), and straddle carriers. Another category concerns the analysis of trade-offs between yard storage area and container handling efficiency. The third category includes more comprehensive studies integrating the main aspects of the previous two categories.

Different types of handling equipment are involved in container terminals. Although more automated equipment has been recently introduced in many terminals due to higher efficiency requirements, the majority of terminals still employ conventional handling systems that can be basically grouped into two categories: transtainer and straddle carrier systems. Detailed equipment characteristics of these two systems are provided in Taleb-Ibrahimi *et al.* (1993). In addition to these stacking-based systems, there is also one known as 'chassis system', which is suitable for terminals with ample land.

Optimal scheduling of handling equipment is important not only because of its high cost but, more importantly, due to the need for fast vessel turnaround times. Although integrated scheduling of all equipment involved is thus in principle necessary, the complexity of such a problem, also due to its combinatorial nature, will often necessitate the scheduling of each equipment to be solved independently.

Martin Jr. *et al.* (1988) addressed the container ship load planning problem for the transtainer system. Transtainer operations were the bottleneck in the loading process. A heuristic algorithm was developed based on rules of thumb prevalent in the terminals. The objectives of the heuristic algorithm were the minimisation of both the transtainer movement time and the number of containers that had to be rehandled for the specific voyage.[1] Kim and Kim (1999) also dealt with the transtainer scheduling problem. They formulated a Mixed Integer Program and solved it by using Dynamic Programming. However, it is doubtful whether this method can be of help in improving transtainer operations, due to its heavy computational requirements.

Steenken *et al.* (1993) examined straddle carrier operations in a terminal, defining the straddle carrier routing problem as a Multiple Travelling Salesman one. Kim and Kim (1999) treated a single straddle carrier routing problem for export tasks only, and solved it by the dynamic programming. Their solution procedure does not seem practical though due to its heavy computational demands. Evers and Koppers (1996) considered the traffic control of automated guided vehicles (AGV) in a container terminal. Daganzo (1989) investigated the quay crane scheduling problem by effectively assigning a set of quay cranes to a set of ships (actually ship holds). He assumed that cranes can move freely from hold to hold (maybe to another ship too); a restrictive and unrealistic assumption for rail mounted quay cranes of modern terminals that obviously cannot cross over each other. Peterkofsky and Daganzo (1990) developed an exact solution method for a class of problems considered by Daganzo, while Daganzo solved the problem by a heuristic algorithm.

The relationship between storage space utilisation and complexity of handling operations raises a crucial problem, especially for the ship loading sequence. Taleb-Ibrahimi *et al.* (1993) tackled this problem by using an analytical model. Kim and Kim (1994) treated a similar problem but with a Mixed Integer Program.

As mentioned previously, container rehandle may occur during the ship-loading process. It also arises when an import container is retrieved from a block of container stack in the storage area in order to be placed on a road truck. Rehandle of import containers is inevitable due to the random retrieval of a container.

Castilho and Daganzo (1993) investigated handling strategies for import containers by estimating the expected number of container rehandles. Kim (1994) and Kim (1997) developed a formula for estimating the number of rehandles. Kim and Bae (1998) considered the re-marshalling of export containers that had to be moved from one container-stacking block to another, where they are stacked in the order of loading. The overall problem was decomposed into three sub-problems solved through mathematical programming techniques. Chen *et al.* (2000) carried out statistical analyses on the relationship between container rehandles, container handling volume, and storage density of a yard.

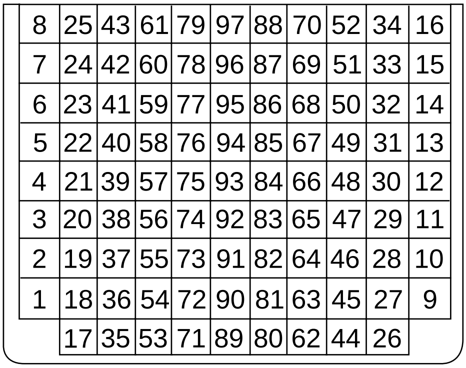
Due to difficulties in decision processes of container terminals, Hee *et al.* (1988) developed a decision support system for port operations, integrating different models for specific activities involved in terminal operations. Subsequently, Hee and Wijbrands (1988) modified the system for container terminal planning. They assumed reshuffling of export containers ahead of loading, thus eliminating rehandle problems. Chung *et al.* (1988) recognised that transtainer operations can be a bottleneck in the loading process due to the frequent export container rehandles. They thus proposed a buffer area where rehandled export containers are temporarily stored until they are called for by the loading process. Through a simulation model, they analysed the effectiveness of the buffer area under various loading plans and fleet sizes of handling equipment. In addition to the above two container related studies, some researchers (Lai and Lam, 1994; Ballis and Abacoumkin, 1996; Ballis *et al.*, 1997; Gambardella *et al.*, 1998; and Merkuryev *et al.*, 1998) carried out simulations to see how various equipment allocation strategies could affect terminal efficiency.

With regard to ship stability, Haghani and Kaisar (2001) developed a heuristic algorithm for ship stowage planning, minimising container rehandles during unloading at destination ports, while keeping ship stability acceptable. The rehandle in their study is the same as the one in Martin Jr. *et al.* (1988). However, this problem is unlikely to arise since, in most cases, one ship hold (called a ship bay in practice) is dedicated to containers destined for a certain port, and there is no destination mix in a specific hold. Studies on aircraft stability also exist. Martin-Vega (1985) and Zhang *et al.* (1992) dealt with the problem of assigning cargo to airplanes, but no rehandle problems arise in aircraft loading.

All in all, no research has been conducted as yet on the relationship between ship stability and container rehandles that take place in a yard during the ship-loading process, which is the objective of this study. Some of the above studies assume advanced reshuffling or temporary buffer areas, to avoid the export container rehandling issue. However, such methods cannot be adopted in busy and land-scarce container terminals, as for instance in Japan, and the rehandling problem remains an important issue in ship load planning.

PROBLEM DEFINITION AND SOLUTION

This section presents a formulation of the ship loading sequence and describes its solution method. Although a straddle carrier system is considered throughout this section, the model is easily adaptable to the transtainer system without any major changes. Furthermore, a cellular (or LOLO) containership is assumed here. Figure 1 shows a typical cross-sectional view (or a ship bay) of such a ship. Each cell in Figure 1 represents a container slot and the number in a cell implies the typical



8	25	43	61	79	97	88	70	52	34	16
7	24	42	60	78	96	87	69	51	33	15
6	23	41	59	77	95	86	68	50	32	14
5	22	40	58	76	94	85	67	49	31	13
4	21	39	57	75	93	84	66	48	30	12
3	20	38	56	74	92	83	65	47	29	11
2	19	37	55	73	91	82	64	46	28	10
1	18	36	54	72	90	81	63	45	27	9
	17	35	53	71	89	80	62	44	26	

Figure 1: Cross section of cellular container ship

order of the loading process. Thus, the order of the loading process defines the vertical location of a container stored in the ship hold. Of course, this also defines the container's horizontal location, but this is not used in this study.

The straddle carrier (hereafter referred to as *carrier*) is a versatile piece of equipment used to stack containers, load or unload them onto road trucks and move them inside the terminal. Containers are placed end to end in long rows, one container wide and up to three containers high, as shown in Figure 2, which can be straddled by the carriers. Major container terminals employing the carrier system use high performance carriers that can straddle and hold a container up to 'four high' and move over a row freely. In the stack layout of Figure 2, picking up a target container (blacked box) results in four container rehandles (hatched boxes). Normally a carrier moves over the row towards the dockside in order to carry a container to a quay crane. To reduce the number of rehandles, a low performance carrier (three high) can move towards the land side when fewer rehandles are expected; however, this is not always the case in practical operations because of longer carrier movement. This study assumes only dockside movement.

Ship stability is evaluated by three factors: metacentric height or GM (the distance between the centre of Gravity and the Metacenter, shown in Figure 3), trim and heel. Among those, we use GM as measure of ship stability. Stability issues raised by the other two factors are tractable by the use of ballast tanks. In this study, we examine the loading sequence of only one ship bay. In practice, GM ought to be assessed by taking into account all bays, but this leads to considerable complexity of the problem. Therefore, we restrict the problem definition in order to facilitate its solution. Imbalance of GM across bays raises longitudinal bend and heel problems but, as mentioned above, these can be practically solved by proper use of ballast.

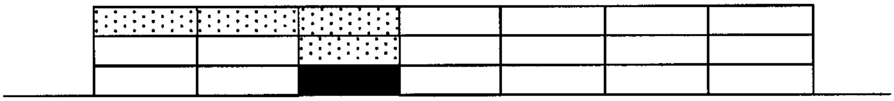


Figure 2: Container stack in a yard

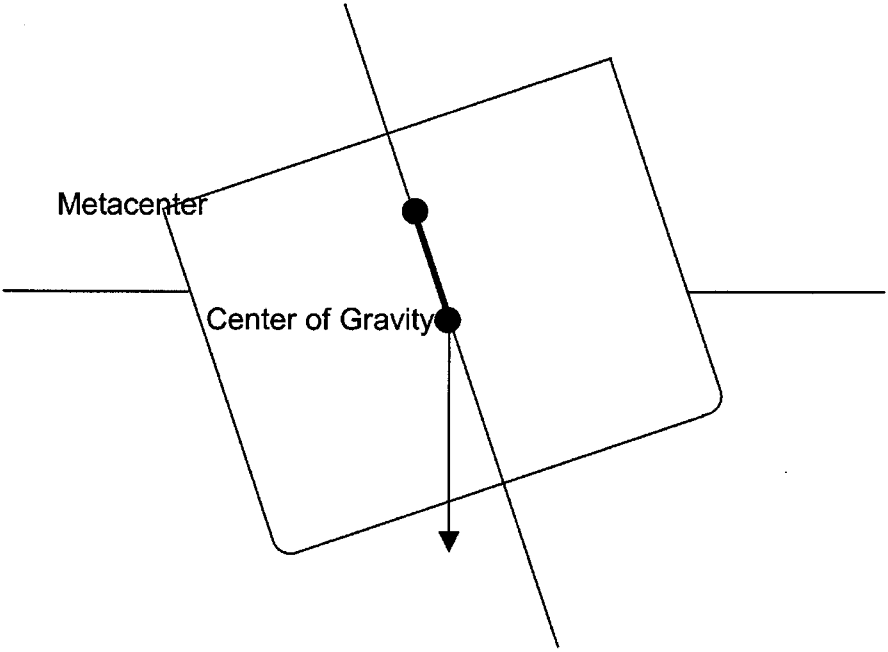


Figure 3: Metacenter and centre of gravity

Container rehandle estimation

As described in the relevant literature of container rehandling (Castilho and Daganzo, 1993; Kim, 1994; and Kim, 1997), the difficulty of estimating the number of container rehandles is caused by the random retrieve. This is typical for import container distribution, but it is also the case in export container loading. In container loading, the loading sequence is obviously predetermined and this implies that container retrieve is not random but scheduled. Consequently, the number of rehandles can be exactly calculated. Our purpose here, however, is to determine the loading sequence while the criteria of ship stability and container rehandles are simultaneously satisfied. Therefore, at the planning stage, the randomness of export container retrieve still holds true.

Assuming that a container stack row in a yard is dedicated to all containers stored in a particular ship bay (which is a row of containers onboard a ship), we consider a loading sequence of a set of containers in that row. Normally, rehandled containers are not moved back to the original location after a specific target container is retrieved from a row, and are temporarily stacked nearby till they are loaded. However, as this practice makes problem modelling more difficult, the rehandled containers are assumed to be moved back to their original locations.

Assuming that container locations in a yard row are given serial numbers, let S_{ij} be the expected number of rehandles to pick up a container at location i in the row as the j-th container (which is placed in its corresponding position in the ship bay). When picking up a target container in Figure 2, we obtain the expected number of the hatched containers to be rehandled. Letting N be the number of containers in the row, any j-1 containers are retrieved with the probability of equation (1) before another is loaded as the j-th one.

$$1 - \frac{j-1}{N-1} \tag{1}$$

Therefore, the number of containers remaining among the hatched ones is defined as:

$$S_{ij} = \left(1 - \frac{j-1}{N-1}\right) B_i \tag{2}$$

where B_i is the number of containers to be rehandled when a container at location i (black box in Figure 2) is picked up as the first container in the loading sequence.

As mentioned before, the value of B_i for the low performance carrier is higher than that for the high performance carrier.

GM estimation

The GM is the distance between the metacenter and the centre of gravity, showing, among other things, how easy or difficult is for the ship to capsize. A low GM endangers stability while a high one involves more rolling that can cause serious cargo damage. The desirable GM is often said to be one meter, but this can change depending of course on ship design and cargo conditions.

To formulate the loading sequence, we define the GM contribution ratio, G_{ij}, for a container at location i picked up as the j-th container to be stored at cell position j in the ship bay. G_{ij} is then represented as follows:

$$G_{ij} = \frac{D_j T_i}{\sum_i T_i} \tag{3}$$

where:

D_j: the distance between the metacenter and cell location j in the ship bay;
T_i: the weight of a container at location in the row.

Formulation and solution method for the loading sequence

Although the desirable GM is in general one meter, other GM values are often used when taking into account other ship condition related factors. Furthermore, loading planners and ship officers in charge of cargo handling may soften the GM restriction in order to reduce the number of required container rehandles that prevent the quick ship turnaround. Such a trade-off analysis requires the set of non-inferior solutions to the two objective problems.

Among a number of techniques for generating a non-inferior solution set, we employ the weighting method (Cohon, 1978). In this method, the problem is defined as a mathematical programming model with a single objective which incorporates the two objectives. Let $x_{ij} = 1$, if a container at location i in the row is the j-th container to be picked up and placed in its corresponding position in the ship bay, and 0 otherwise. The expected number of rehandles and the GM are defined as $\sum_i \sum_j S_{ij} x_{ij}$ and $\sum_i \sum_j G_{ij} x_{ij}$, respectively. The single objective problem is then defined as:

$$[\mathrm{P}] \quad \text{Minimise} \quad Z = \sum_i \sum_j (\alpha S_{ij} + \beta\, G_{ij}) x_{ij} \tag{4}$$

Subject to:

$$\sum_j x_{ij} = 1, \ \forall i \tag{5}$$

$$\sum_i x_{ij} = 1, \ \forall j \tag{6}$$

$$x_{ij} \in \{0, 1\}, \ \forall i, j \tag{7}$$

where α and β are weights on the rehandle and GM objectives. As problem [P] is a classical assignment problem, the LP relaxed formulation (ie constraint set (7) is relaxed) always yields an integer solution because of the totally unimodularity. Non-inferior solutions are generated by solving the problem with varying α and β values.

With the minimum level of ship stability guaranteed, the non-inferior solutions are obtained by:

$$[\mathrm{PG}] \quad \text{Minimise} \quad Z = \sum_i \sum_j (\alpha S_{ij} + \beta\, G_{ij}) x_{ij} \tag{8}$$

Subject to:

$$\sum_j x_{ij} = 1, \ \forall i \tag{9}$$

$$\sum_i x_{ij} = 1, \ \forall j \tag{10}$$

$$\sum_i \sum_j G_{ij} x_{ij} \geq GL, \ \forall i, \ j \tag{11}$$

$$x_{ij} \in \{0, 1\} \ \forall i, \ j \tag{12}$$

where GL is the minimum GM guaranteed. Note that constraint set (12) can be no longer relaxed for easy solution procedure. Problem [PG] is not known to have a polynomially-bounded solution method.

NUMERICAL EXPERIMENTS

Estimating weights for rehandle and GM

As mentioned in the previous section, we employ the weighting method to obtain a set of non-inferior solutions, by varying weights α and β. While later on, we will examine the non-inferior solutions with various patterns of container stack (or row) in the yard, we first try to obtain some insights on these weights from practical operations. For this, we will identify values of α and β through principal component analysis.

We assume that a loading sequence planner, given underlying (observed or independent) variables of rehandle and GM, intuitively adjusts these weights to have a new dependent (or principal component) variable that is defined by a linear function representing well the quality of the loading plan. As two observed variables are given, two principal components are to be defined with respective eigenvalues. Then, the planner is assumed to be concerned with the first principal component, ie the principal component with the largest eigenvalue. Since the principal component is given as a linear function consisting of independent variables, eigenvectors incident to variables correspond to weights α and β.

The data to be used for the analysis is the container loading information observed in Port of Kobe. The information includes container weights and locations in yard stacks and in cargo holds onboard three ships, each with capacity of 600, 2,000, and 2,600 TEUs. Each ship carried some containers that were not to be handled in Kobe; therefore, these were not included in the analysis. In the stacks, there were also containers that were not to be loaded onto the three ships; those containers are excluded from the calculation of the expected number of rehandles $\sum_i \sum_j S_{ij} x_{ij}$. The metacenter positions of the ships are not known; they are all assumed at the upper deck level.

As the data volume is very low, we carry out our analysis not for individual ship data, but for the entire set of the three ship data. Table 1 shows the estimated model (principal component analysis). In this way, we have two models: one for

Table 1: Rehandle-GM analysis

Location onboard	Eigenvector (factor loading)		Proportion
	Rehandle	GM	
On-deck	0.710 (0.821)	−0.705 (−0.816)	67.0%
Hold	0.714 (0.774)	−0.700 (−0.759)	58.8%

containers stored on deck and the other for the ones stored in cargo holds. As the number of rehandles is to be minimised while the GM should be maximised, α and β must be positive and negative respectively in the optimisation problem. The weight estimation by the principal component analysis seems reasonable because, as expected, both models have positive values of eigenvector for the rehandle and negative values for the GM. The absolute values of the two eigenvectors are almost the same both for the 'on-deck' and the 'in-hold' model. This means planners intuitively take into account the rehandle and GM with the same weight in planning loading sequence.

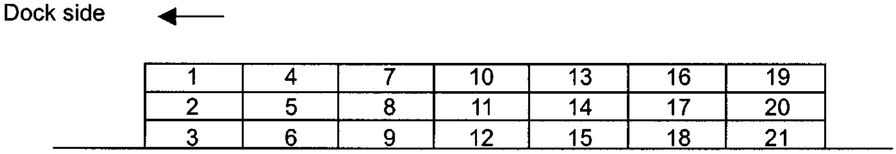
The complexity of efficient loading

In the following section, the numerical experiments are carried out for various patterns of stacked yard containers. Some patterns make the loading process easy in terms of rehandle task but others do not. One way to represent the complexity of the loading task is to calculate the number of containers to be rehandled for the maximum GM. To compute the number of rehandles, we have to trace the container movements based on the loading process for the maximum GM. Adaptation of our analysis to mega containership loading is a time consuming task. Consequently, the *stack complexity* index is introduced, defined by equation (13), representing the complexity of loading:

$$SC = \sum_{i} \Delta(Y_i, Q_i) \tag{13}$$

where:
Y_i: the location of container in the yard stack;
Q_i: the location of container in the ideal stack; and
$\Delta(a,b)$: the distance between locations a and b.

The *ideal stack* is defined as the stack where, as shown in Figure 4, the set of containers is moved and loaded onto the ship, to obtain the maximum GM, without rehandles. $\Delta(a,b)$ is measured along with the loading sequence, ie the number in the row of Figure 4. Note that as GM is a vertical distance, all containers in a particular row (called a set of *convertible containers*) have the same contribution in the formation of the GM. If there are five containers in a bottom row of the ship bay, the marked containers in Figure 5 are convertible. The



1	4	7	10	13	16	19
2	5	8	11	14	17	20
3	6	9	12	15	18	21

Figure 4: Ideal order of loading sequence

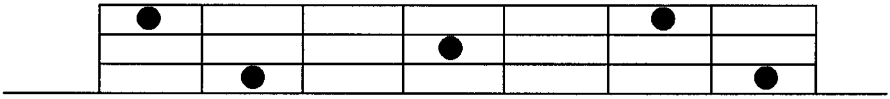


Figure 5: Set of convertible containers

distances of a set of convertible containers (denoted by *CC*) are calculated as follows (where the containers in the ideal stack are numbered in the order of loading sequence):

Step 1: Identify, from the given stack, a set of containers that correspond to a *CC* in the ideal stack for a particular row of the ship hold;

Step 2: Locate the container of the *CC*, in the given stack, that is the closest to the container with the minimum number of the *CC* in the ideal stack. Calculate the distance for it;

Step 3: Delete the container from both *CC*s. If there is no container in the *CC*, then STOP; otherwise go to Step 2.

Given a stack of containers, we are concerned with how the stack complexity value is related or corresponds to the number of rehandles. Consequently, we undertook regression analysis for both values. One hundred different container stack sets with 36 containers were generated with different seed sets for random numbers. The containers are stacked three high in the yard. Figure 6 illustrates the relationship between the two values. The estimated model is given below, where the number of rehandles is denoted by *RH*:

$$RH = 0.17SC + 8.67 \quad (R^2 = 0.23) \tag{14}$$

The coefficient of *SC* is positive, therefore *SC* seems to be a substitute for *RH*.

Analyses

Given the same 100 sets of container stack data that were analysed in the previous section, we first obtain a set of non-inferior solutions for each of them. The objective function coefficients α and β of the weighting method are presented in

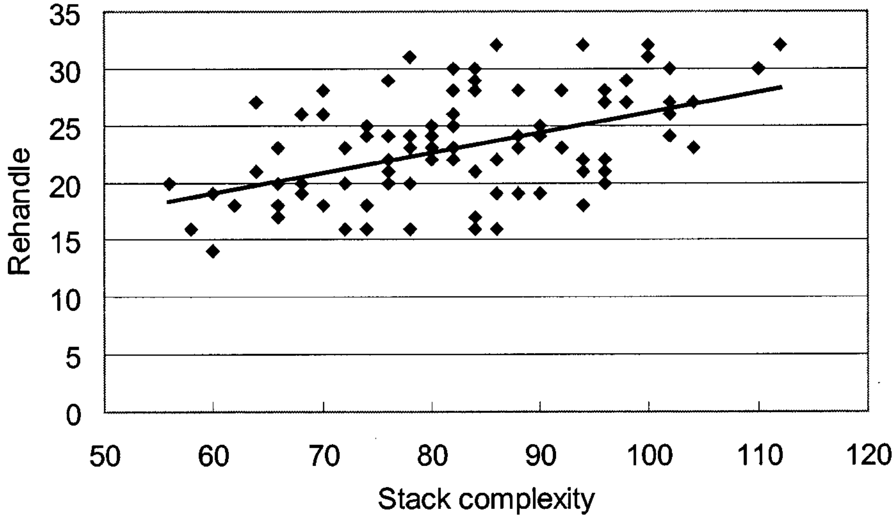


Figure 6: Relationship between rehandle and stack complexity

Table 2. Basically, the weights vary by 0.05, with smaller intervals in the final phase. The weights for the 36 containers range from 5 to 20 tons. We assume that a ship bay accommodates six containers in a horizontal row and six containers high, including the on-deck stowage section. The metacenter is assumed to be located 8.8 m above the hold bottom. As mentioned above, the problem [P], with the integrality constraints relaxed, is a classical assignment problem that is solvable by linear programming, while problem [PG] is solved by integer programming. These are carried out through the linear, non-linear and integer programming solver system LINDO, on a Sun Enterprise-Series workstation.

First, we identify the set of non-inferior solutions by problem [P]. Note that due to its mechanism in the computation process, the weighting method approximates the non-inferior set and solutions are only identified on the convex hull of the non-inferior set. However, this is the case for a solution space of the expected number of rehandles *versus* the GM. Our goal is to identify the approximate non-inferior set in a solution space not with the expected but with the observed number of rehandles, ie rehandles that actually take place during the loading process. By converting the expected number to the observed one, some non-inferior solutions may become inferior. This necessitates the identification of non-inferior solutions from all the converted solutions.

As discussed above, the stack complexity for the 100 container sets roughly varies from 60 to 110. Thus, solution quality depends on stack complexity. To get an insight to this, we analyse non-inferior solution quality by grouping solutions

Table 2: Set of weights

Weight set	α	β
1	−2.00	0
2	−1.95	0.05
3	−1.90	0.10
	⋮	Interval=0.05
38	−0.15	1.85
39	−0.10	1.90
40	−0.07	1.93
41	−0.04	1.96
42	−0.01	1.99
43	−0.0001	1.9999
45	0	2.00

Table 3: The relationship between the solution quality and stack complexity

Stack complexity	η	ζ	R^2
– 70	0.013	4.585	0.76
70 – 80	0.042	3.865	0.76
80 – 90	0.055	3.711	0.72
90 – 100	0.092	3.272	0.64
100 –	0.082	3.566	0.73

$RH = \eta e^{\zeta \mathrm{G}M}$

based on five categories of the stack complexity, ie less than 70, 70 – 80, 80 – 90, 90 – 100, and more than 100. The estimated trade-off curves of the GM and the number of rehandles for these categories, given in Table 3, are illustrated in Figure 7. As expected, the function curve appears near the bottom right corner with smaller value of the stack complexity; this means a better solution is more likely when stack complexity is low.

The use of the expected number of rehandles, instead of the observed ones, enables us to formulate the problem as a mathematical programming problem. Thus, the extent to which the formulation works properly depends on how the observed number is reflected by the expected one. Figure 8 shows their relationship, estimated by the following equation, where the observed and expected numbers are denoted by *ORH* and *ERH*, respectively.

$$ORH = 3.45ERH - 270.42 \quad (R^2 = 0.66) \tag{15}$$

As can be seen in Figure 8, the number of rehandles is over-evaluated in the formulation. However, the expected number works properly in obtaining the solution since the coefficient of determination R^2 is positive and high.

Next, we carry out some experiments for problem [PG] when the minimum GM value is known. Due to the limitations of LINDO with regard to the size of integer programming problems, the experiments were performed with 24 containers of one

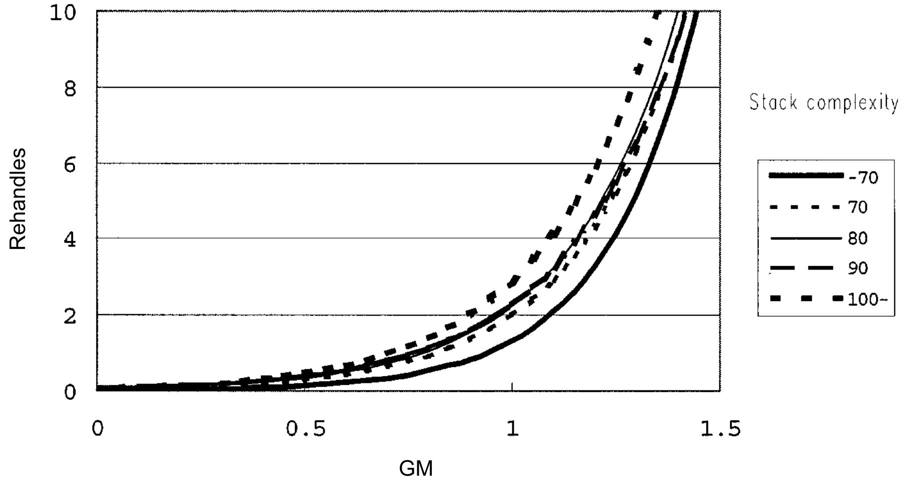


Figure 7: Trade-off curves of the GM and the number of rehandles

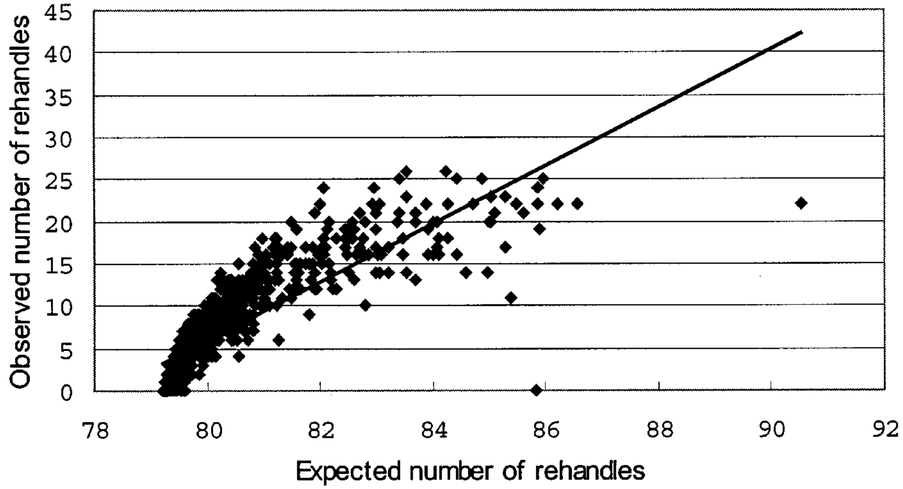


Figure 8: Relationship between the expected and observed numbers of rehandles

particular stack. Thus, the containers stacked three high in the yard are loaded into a ship bay with four containers wide and six high. One hundred different stacks were first generated. Their stack complexity ranged from 30 to 72. Nine typical stack patterns were subsequently chosen for the experiments.

For a particular stack pattern, problem [P] was solved with constraints (7) relaxed, [PG] without constraint (11), and [PG] with a GM of 0.8, 1.0 and 1.2.

Hereafter these are referred to as [PLP], [PGR], [PG8], [PG10], and [PG12]. LINDO applies the LP-simplex method for PLP and the branch-and-bound method for the integer programming problems.

Table 4 shows the GM, the number of rehandles, and CPU time (in seconds) of solutions obtained by the five models mentioned above. Solutions are obtained with the weight sets ranging from 1 to 45, while solutions with nine out of the 45 sets are illustrated in the Table. Note that the GM and the number of rehandles of each PLP solution are obviously the same as those of PGR. Computational time for PGR was supposed to be larger than PLP. However, contrary to this expectation, both computational times were nearly the same. PGR utilises the branch-and-bound method where the lower bound, to fathom unnecessary branches, is supposed to be calculated by the LP relaxation to the original problem. It is expected that in the early stage of the branch-and-bound procedure, an optimal solution is detected in solving the relaxation problem. Thus, the computational time of PGR is almost the same as that of PLP.

As we expected, the GM and the number of rehandles generally increase together with the weight set number, ie α and β. PG, under the GM constraint, yields a feasible solution with a particular weight set, for which PLP and PGR detected a solution with the GM not suitable for PG with the GM constraint. Its computational time, however, is considerably higher (up to 80 times as much) than the corresponding PLP and PGR. Note that for some problems (eg problem 8 with weight set 35), different formulations yield the same GM value but different number of rehandles. As their objective function values are the same, their GMs and expected number of rehandles are identical. Different loading sequences may, however, result in a particular expected number. For problem 8 in PG8, containers in locations 12 and 16 of the yard stack are loaded as the 14th and 8th container respectively, while in PG10 they are loaded as the 8th and 14th one. The value associated with the container in location 12 in PG8 is different from that in PG10 and this is also the case for the container in location 16. Of course, one has the same total value of the containers in both locations in PG8 and PG10. The difference in decision variable value for the expected number of rehandles results in the different observed number of rehandles.

Finally we looked into the total computational time for identifying the non-inferior solution set. As it was not possible to employ a computer program on the workstation to do it for all the solutions generated by LINDO, Table 5 reveals the total computational time being taken by the LINDO computation. The time increases considerably as the GM constraint becomes harder. Therefore, even if one knows the minimum GM being guaranteed, it is better to employ PLP (or PGR) to yield all the solutions (including non-feasible ones in terms of the GM) and then identify the non-inferior set only from the feasible ones.

Table 4: (Part 1) Solution comparison between LP and IP

P#*	WS**	PLP & PGR				PG8			PG10			PG12		
(SC***)		GM	RH	$CPU_{(PLP)}$	$CPU_{(PGR)}$	GM	RH	CPU	GM	RH	CPU	GM	RH	CPU
1	1	0.99	0	0.14	0.15	0.99	0	0.23	1.19	0	0.88	1.26	1	6.61
(34)	5	0.99	0	0.14	0.15	0.99	0	0.19	1.19	0	0.50	1.26	1	2.49
	10	1.19	0	0.13	0.16	1.19	0	0.17	1.19	0	0.21	1.26	1	0.33
	15	1.19	0	0.14	0.16	1.19	0	0.20	1.19	0	0.21	1.26	1	0.23
	20	1.26	1	0.13	0.15	1.26	1	0.22	1.26	1	0.19	1.26	1	0.18
	25	1.32	1	0.14	0.14	1.32	1	0.20	1.32	1	0.19	1.32	1	0.21
	30	1.34	1	0.13	0.14	1.34	1	0.20	1.34	1	0.17	1.34	1	0.19
	35	1.40	2	0.14	0.15	1.40	2	0.15	1.40	2	0.18	1.40	2	0.21
	40	1.43	3	0.13	0.15	1.43	3	0.18	1.43	3	0.15	1.43	3	0.15
	45	1.43	19	0.11	0.14	1.43	43	0.15	1.43	28	0.16	1.43	41	0.16
2	1	0.77	0	0.13	0.15	0.90	0	0.93	1.01	1	3.97	1.21	4	2.95
(42)	5	0.77	0	0.14	0.15	0.90	0	0.78	1.01	1	1.93	1.21	4	2.79
	10	0.77	0	0.13	0.15	0.90	0	0.49	1.01	1	0.39	1.21	5	2.11
	15	1.01	1	0.14	0.16	1.01	1	0.19	1.01	1	0.20	1.21	4	2.10
	20	1.07	2	0.13	0.16	1.07	2	0.20	1.07	2	0.21	1.21	5	0.91
	25	1.10	3	0.14	0.16	1.10	3	0.17	1.10	3	0.19	1.21	5	0.87
	30	1.20	4	0.13	0.15	1.20	4	0.18	1.20	4	0.19	1.21	4	0.81
	35	1.37	7	0.13	0.14	1.37	7	0.17	1.37	7	0.18	1.37	7	0.16
	40	1.43	9	0.14	0.16	1.43	9	0.16	1.43	9	0.15	1.43	8	0.17
	45	1.43	16	0.13	0.15	1.43	41	0.16	1.43	31	0.15	1.43	36	0.15
3	1	1.05	0	0.13	0.15	1.05	0	0.21	1.05	0	0.21	1.22	1	1.04
(44)	5	1.05	0	0.14	0.16	1.05	0	0.21	1.05	0	0.20	1.22	1	0.69
	10	1.05	0	0.12	0.17	1.05	0	0.19	1.05	0	0.20	1.22	1	0.83
	15	1.05	0	0.14	0.14	1.05	0	0.19	1.05	0	0.19	1.22	1	0.35
	20	1.18	1	0.13	0.16	1.18	1	0.20	1.18	1	0.23	1.22	1	0.35
	25	1.34	4	0.14	0.15	1.34	4	0.20	1.34	4	0.18	1.34	4	0.20
	30	1.36	5	0.13	0.15	1.36	5	0.18	1.36	5	0.19	1.36	5	0.18
	35	1.37	5	0.13	0.15	1.37	5	0.17	1.37	5	0.19	1.37	5	0.19
	40	1.42	8	0.13	0.14	1.42	7	0.15	1.42	8	0.16	1.42	7	0.17
	45	1.43	37	0.12	0.14	1.43	33	0.16	1.43	35	0.15	1.43	42	0.16

Continued

Table 4: (Part 1) continued

P#*	WS**	PLP & PGR				PG8			PG10			PG12		
(SC***)		GM	RH	$\text{CPU}_{(\text{PLP})}$	$\text{CPU}_{(\text{PGR})}$	GM	RH	CPU	GM	RH	CPU	GM	RH	CPU
4	1	0.50	0	0.14	0.16	0.82	0	0.47	1.01	2	1.07	1.22	5	3.91
(48)	5	0.50	0	0.14	0.15	0.82	0	0.36	1.01	2	0.74	1.22	5	2.63
	10	0.70	0	0.15	0.15	0.82	0	0.25	1.01	2	1.73	1.22	5	1.98
	15	1.01	2	0.13	0.15	1.01	2	0.25	1.01	2	0.19	1.22	5	2.23
	20	1.06	3	0.13	0.16	1.06	3	0.21	1.06	3	0.18	1.22	5	0.69
	25	1.30	5	0.14	0.16	1.30	5	0.22	1.30	5	0.18	1.30	5	0.20
	30	1.32	6	0.13	0.15	1.32	6	0.17	1.32	6	0.17	1.32	6	0.18
	35	1.35	6	0.12	0.14	1.35	6	0.16	1.35	6	0.18	1.35	6	0.16
	40	1.41	9	0.13	0.13	1.41	9	0.16	1.41	9	0.14	1.41	9	0.15
	45	1.43	34	0.12	0.15	1.43	46	0.15	1.43	33	0.15	1.43	33	0.15
5	1	−0.71	0	0.13	0.15	0.83	3	2.29	1.03	2	2.14	1.21	4	1.88
(52)	5	−0.05	0	0.14	0.15	0.83	3	1.39	1.03	2	1.99	1.21	4	1.70
	10	0.35	1	0.13	0.16	0.83	3	1.00	1.03	2	0.98	1.21	4	0.43
	15	0.78	2	0.15	0.15	0.91	2	0.56	1.03	2	0.40	1.21	4	0.23
	20	1.08	3	0.14	0.15	1.08	3	0.26	1.08	3	0.25	1.21	4	0.24
	25	1.24	5	0.13	0.14	1.24	5	0.18	1.24	5	0.19	1.24	5	0.20
	30	1.30	5	0.14	0.14	1.30	5	0.19	1.30	5	0.20	1.30	5	0.20
	35	1.35	5	0.13	0.14	1.35	5	0.15	1.35	5	0.20	1.35	5	0.18
	40	1.42	8	0.12	0.13	1.42	8	0.17	1.42	8	0.15	1.42	8	0.16
	45	1.43	34	0.13	0.12	1.43	31	0.16	1.43	32	0.15	1.43	32	0.15

* Problem#
** Weight set
*** Stack complexity

Table 4: (Part 2) Solution comparison between LP and IP

P#*	WS**	PLP & PGR				PG8			PG10			PG12		
(SC***)		GM	RH	CPU$_{(PLP)}$	CPU$_{(PGR)}$	GM	RH	CPU	GM	RH	CPU	GM	RH	CPU
6	1	0.22	0	0.14	0.14	0.81	1	5.14	1.00	2	2.38	1.23	4	3.35
(54)	5	0.22	0	0.14	0.16	0.81	1	8.56	1.00	2	1.63	1.23	4	2.59
	10	0.61	0	0.14	0.14	0.81	1	0.69	1.00	2	1.35	1.23	4	4.94
	15	0.81	1	0.14	0.16	0.81	1	0.19	1.00	2	0.59	1.23	4	2.30
	20	0.96	1	0.14	0.16	0.96	1	0.18	1.00	2	0.22	1.23	5	1.89
	25	1.12	5	0.14	0.15	1.12	4	0.21	1.12	4	0.18	1.23	5	0.21
	30	1.31	5	0.13	0.15	1.31	4	0.19	1.31	5	0.18	1.31	5	0.18
	35	1.38	8	0.13	0.13	1.38	8	0.21	1.38	9	0.17	1.38	9	0.17
	40	1.42	10	0.13	0.15	1.42	8	0.15	1.42	9	0.17	1.42	9	0.17
	45	1.43	39	0.12	0.15	1.43	35	0.16	1.43	43	0.16	1.43	22	0.16
7	1	0.05	0	0.14	0.15	0.84	2	2.04	1.01	5	3.03	1.23	71	2.25
(58)	5	0.05	0	0.14	0.16	0.84	2	2.17	1.01	5	2.26	1.23	7	3.32
	10	0.05	0	0.14	0.16	0.84	2	1.96	1.01	5	1.79	1.23	7	2.15
	15	0.42	0	0.14	0.15	0.84	2	0.76	1.01	5	2.46	1.23	7	1.53
	20	0.84	2	0.14	0.17	0.84	2	0.17	1.01	5	0.22	1.23	7	2.30
	25	1.29	8	0.14	0.15	1.29	8	0.18	1.29	8	0.18	1.29	9	0.19
	30	1.35	10	0.14	0.15	1.35	9	0.16	1.35	9	0.17	1.35	10	0.17
	35	1.37	9	0.13	0.15	1.37	8	0.16	1.37	9	0.17	1.37	9	0.19
	40	1.40	11	0.13	0.14	1.40	13	0.16	1.40	13	0.17	1.40	13	0.20
	45	1.43	23	0.13	0.16	1.43	32	0.16	1.43	37	0.16	1.43	26	0.16
8	1	0.22	0	0.13	0.15	0.82	2	3.69	1.00	5	5.15	1.21	10	3.68
(64)	5	0.22	0	0.14	0.16	0.82	2	1.15	1.00	5	5.58	1.21	10	3.82
	10	0.50	1	0.13	0.16	0.82	2	3.13	1.00	5	3.61	1.21	10	1.31
	15	0.62	2	0.14	0.15	0.82	2	1.05	1.00	5	1.47	1.21	10	2.65
	20	0.92	4	0.14	0.16	0.92	4	0.19	1.00	5	0.25	1.21	10	1.75
	25	0.95	4	0.14	0.17	0.95	4	0.20	1.00	5	0.23	1.21	10	4.38
	30	1.04	6	0.13	0.16	1.04	6	0.23	1.04	6	0.19	1.21	13	2.31
	35	1.34	16	0.13	0.14	1.34	14	0.18	1.34	16	0.30	1.34	14	0.20
	40	1.36	14	0.13	0.13	1.36	14	0.16	1.36	14	0.16	1.36	14	0.18
	45	1.43	42	0.13	0.14	1.43	41	0.16	1.43	42	0.17	1.43	23	0.16

Continued

Table 4: (Part 2) continued

P#*	WS**	PLP & PGR				PG8			PG10			PG12		
(SC***)		GM	RH	$CPU_{(PLP)}$	$CPU_{(PGR)}$	GM	RH	CPU	GM	RH	CPU	GM	RH	CPU
9	1	0.33	0	0.14	0.16	0.81	2	7.69	1.00	4	2.10	1.22	8	1.63
(72)	5	0.33	0	0.15	0.15	0.81	2	1.81	1.00	4	0.88	1.22	8	4.21
	10	0.62	0	0.14	0.16	0.81	2	0.37	1.00	4	1.23	1.22	7	1.49
	15	0.82	2	0.14	0.16	0.81	2	0.18	1.00	5	1.26	1.22	8	0.90
	20	1.00	5	0.13	0.16	1.00	5	0.20	1.00	5	0.18	1.22	8	5.43
	25	1.13	7	0.14	0.16	1.13	7	0.25	1.13	7	0.20	1.22	8	1.00
	30	1.26	7	0.13	0.15	1.26	7	0.18	1.26	7	0.17	1.26	7	0.21
	35	1.28	9	0.14	0.15	1.28	9	0.19	1.28	9	0.18	1.28	9	0.18
	40	1.40	14	0.13	0.15	1.40	14	0.16	1.40	14	0.16	1.40	14	0.17
	45	1.43	44	0.13	0.14	1.43	28	0.17	1.43	32	0.15	1.43	41	0.16

Table 5: Total computational time (in seconds)

Problem #	PLP	PGR	PG8	PG10	PG12
1	6.07	6.75	8.73	11.49	28.58
2	6.09	6.84	14.22	20.80	57.03
3	6.08	6.82	8.45	8.67	19.78
4	6.01	6.78	9.90	17.41	52.73
5	6.02	6.60	26.18	23.23	23.99
6	6.02	6.72	37.67	35.21	68.80
7	6.03	6.91	37.59	41.76	45.18
8	5.99	6.87	33.13	54.54	101.90
9	6.02	6.90	30.77	25.45	44.91
Average	6.04	6.80	22.96	26.51	49.21

Solution methods for the multi-objective problem include the constraint method (Cohon, 1978) that solves the problem by optimising one objective while all others are constrained to some value. When this method is applied to the loading problem, the rehandle-objective problem is defined with GM constraints. This problem formulation must be made as (8)-(12) while β is set to zero. Thus, the constraint method is restrictive for our problem in terms of computational time.

CONCLUDING REMARKS

This paper addressed the problem of obtaining the non-inferior solution set for containership loading. In ship loading tasks, a major concern is ship stability, typically represented by the GM (the distance between the centre of gravity and the metacenter). Another concern is container rehandling which occurs when specific containers are picked up from the container stacks in the yard. The number of rehandles is supposed to be a function of the loading order of containers. However, the difficulty lies in the fact that the number of rehandles is only obtained after the loading sequence is determined; in other words, after the solution of the loading problem is identified. To overcome this problem, the notion of the expected number of rehandles was introduced in this paper. From extensive numerical experiments, it was shown that the expected value has a close positive relationship with the observed number of rehandles.

Two types of mathematical programming formulations were investigated in this study: linear and integer programming. The former is formulated as a classical assignment problem that guarantees an integer solution due to the totally unimodularity. However, this does not always guarantee a feasible solution in terms of the GM, when a minimum GM value is provided. When the minimum guaranteed GM is given to a loading planner, it may be practical to identify the non-inferior solution set satisfying the GM constraint. For this, the integer programming with the GM constraint seems more attractive. However, in accordance with the experiments we conducted, it became clear that the integer programming solution procedure requires considerable computational time and, from this point of view, the linear programming formulation is more practical. After all, from a practical viewpoint, a load planner should identify a non-inferior solution set of the loading problem by employing the classical assignment problem and then find solutions from the solution set that satisfies the GM criterion.

To simplify the loading problem, mainly in order to facilitate the solution procedure, the loading problem with only a single ship hold and yard stack was investigated in this paper. This solution procedure could be useful in practical load planning with minor modifications. As regards ship stability, other measures, such as the heel and the trim, ought to be taken into account as well in future research.

ENDNOTES

[1] Note that 'rehandle' in the present paper is not the same as 'rehandle' in Martin Jr. *et al.*, which deals with rehandling that occurs while unloading target containers in subsequent ports.

REFERENCES

Ballis, A and Abacoumkin, C. 1996: A container terminal simulation model with animation capabilities. *Journal of Advanced Transportation* 30: 37-57.

Ballis, A, Golias, J and Abacoumkin, C. 1997: A comparison between conventional and advanced handling systems for low volume container maritime terminals. *Maritime Policy & Management* 24: 73-92.

Castilho, BD and Daganzo, CF. 1993: Handling strategies for import containers at marine terminals. *Transportation Research B* 27: 151-166.

Chen, T, Lin, K and Juang, YC. 2000: Empirical studies on yard operations part 2: Quantifying unproductive moves undertaken in quay transfer operations. *Maritime Policy and management* 27: 191-207.

Chung, YG, Randhawa, SU and McDowell, ED. 1988: A simulation analysis for a transtainer-based container handling facility. *Computers & Operations Research* 71: 113-125.

Cohon, JL. 1978: *Multiobjective Programming and Planning.* Academic Press: New York.

Daganzo, CF. 1989: The crane scheduling problem. *Transportation Research B* 23: 159-175.

Evers, JJM and Koppers, SAJ. 1996: Automated guided vehicle traffic control at a container terminal. *Transportation Research A* 30: 21-34.

Gambardella, LM, Rizzoli, AE and Zaffalon, M. 1998: Simulation and planning of an intermodal container terminal. *Simulation* 71: 107-116.

Haghani, A and Kaisar, EI. 2001: *A model for designing container loading plans for containerships.* Presented in the Annual Conference of the Transportation Research Board.

Hee, KMV, Huitink, B and Leegwater, DK. 1988: PORTPLAN, decision support system for port terminals. *European Journal of Operational Research* 34: 249-261.

Hee, KMV and Wijbrands, R. 1988: Decision support system for container terminal planning. *European Journal of Operational Research* 34: 262-272.

Kim, KH. 1994: Analysis of rehandles of transfer crane in a container yard. *APOR-Conference* 3: 357-365.

Kim, KH. 1997: Evaluation of the number of rehandles in container yard. *Computers & Industrial Engineering* 32: 701-711.

Kim, KH and Bae, JW. 1998: Re-marshalling export containers in port container terminals. *Computers & Industrial Engineering* 35: 655-658.

Kim, KH and Kim, DY. 1994: Group storage methods at container port terminals. *The materials Handling Engineering Division of the 75th Anniversary Commemorative.* Volume ASME, 15-20.

Kim, KY and Kim, KH. 1999: A routing algorithm for a single straddle carrier to load export containers onto a containership. *International Journal of Production Economics* 59: 425-433.

Kim, KH and Kim, DY. 1999: An optimal routing algorithm for a transfer crane in port container terminals. *Transportation Science* 33: 17-33.

Lai, KK and Lam K. 1994: A study of container yard equipment allocation strategy in Hong Kong. *International Journal of Modelling & Simulation* 14: 134-138.

Martin Jr, GL, Randhawa, SU and McDowell, ED. 1988: Computerized container-ship load planning: a methodology and evaluation. *Computers & Industrial Engineering* 14: 429-440.

Martin-Vega, LA. 1985: Aircraft load planning and the computer: description and review. *Computers & Industrial Engineering* 4: 357-369.

Merkuryev, Y, Tolujew, J, Blumel, E, Novitsky, L, Ginters, E, Viktorova, E, Merkuryeva, G and Pronins, J. 1998: A modelling and simulation methodology for managing the Riga harbour container terminal. *Simulation* 71: 84-95.

Peterkofsky, RI and Daganzo, CF. 1990: A branch and bound solution method for the crane scheduling problem. *Transportation Research* 24B: 159-172.

Steenken, S, Henning, A, Freigang, S and Voss, S. 1993: Routing of straddle carriers at a container terminal with the special aspect of internal moves. *OR Spektrum* 15: 167-172.

Taleb-Ibrahimi, M, De Catilho, B and Daganzo, CF. 1993: Storage space vs handling work in container terminals. *Transportation Research* 27B: 13-32.

Zhang, J, Amiouny, S, Batholdi, J and Vande Vate, J. 1992: Balanced Loading. *Operations Research* 40: 238-246.