Bestelldatum: 2010-23-11 15:27:02

Bestellnummer:SUBITO-2010112302005

Universitäts- und Stadt Bibliothek Köln

Universitäts- und Stadtbibliothek Köln D-50931 Köln

Subito-Bestellung normal

Statsbiblioteket
Overbygningsservices
Victor Albecks Vej 1
8000 Arhus C
DAENEMARK

Tel-Nr. +45 8946 2141
E-mail: illdk@statsbiblioteket.dk
Fax: +45 8946 2130
u 8 Subito Library International

Die Auslieferung sollte spätestens erfolgen bis: **26.11.2010 15:27**

Angaben zum bestellten Dokument: **Lieferart: EMAIL**

Signatur:	**Qaa384**
Titel Zeitschrift / Werk:	**Operational research**
Band / Heft:	**90/?**
Erscheinungsdatum:	
Erscheinungsjahr:	1990
Seiten:	**457-471**
Aufsatztitel:	**Minizing of overstowage in containership operations**
Autor (Artikel):	**Aslidis, A.**
Autor / Hrsg. (Werk):	
Ort / Verlag:	**Amsterdam [u.a.],North Holland Publ. Co.**
ISSN / ISBN:	**0922-517X**
Bemerkung:	1951891/dln
Kontaktperson:	Frau Susie Beck Kristensen

Bitte beachten Sie:

Sie erhalten die Rechnung für diese Lieferung über die Subito-Zentralregulierung.
REKLAMATIONEN bitte unbedingt innerhalb von 10 Tagen an die Lieferbibliothek senden.

Wir weisen den Empfänger darauf hin, dass Sie nach geltendem Urheberrecht die von uns übersandten Vervielfältigungsstücke ausschließlich zu Ihrem privatem oder sonstigen eigenen Gebrauch verwenden und weder entgeltlich noch unentgeltlich in Papierform oder elektronische Kopie verbreiten dürfen.

Universitäts- und Stadtbibliothek Köln
Dokumentenlieferung subito

Kontakt:
USB Köln Fernleihe
Tel.: 02214704510
Fax.: 02214705053
e-mail: dokulieferung@ub.uni-koeln.de

MINIMIZATION OF OVERSTOWAGE IN CONTAINERSHIP OPERATIONS

MINIMISATION DU DEPASSEMENT DE LA CAPACITE UTILISABLE DANS LES OPERATIONS DE CHARGEMENT DE CONTAINERS

ANASTASIOS H. ASLIDIS
Marsoft, Inc.
Cambridge, Massachusetts, U.S.A.

ABSTRACT

Overstowage is a situation arising in stacking operations when the storing and retrieval sequences do not obey the "last-in first-out" rule. In this paper we model overstowage as it appears in containership operations by taking into account placement constraints, specifically, stability requirements of the vessel. The latter translates into a constraint on the vertical center of weight of the containers. A polynomial-time recursive algorithm is developed for the one-stack case. Features of multistack problems are presented.

RESUME

Le dépassement de capacité utilisable est un phénomène qui se produit lorsque les opérations de stockage et d'enlèvement n'obéissent pas à la règle LIFO. Dans cet article, nous modélisons le dépassement de capacité, tel qu'il apparait dans les opérations de chargement de containers, en tenant compte des contraintes de placement et des contraintes de stabilité du navire. Ces dernières contraintes sont représentées par une contrainte liée au centre de gravité des containers. Nous avons élaboré un algorithme récursif polynomial pour résoudre le cas du monostockage, et nous présentons quelques résultats dans le cas de problèmes de multistockage.

KEYWORDS

Containership Operations, Recursive Algorithms, Stack, Storage, Stowage, Warehouse

DEFINITION OF OVERSTOWAGE PROBLEMS

This paper deals with the management of overstowage in containership operations. Overstowage is a condition that arises very often in operations which involve stacking of items. There are innumerable occasions in which a group of items (usually boxes or containers) has to be stored for future use. Storing containers in stacks is the cheapest alternative because no extensive supporting structure (cells) is required, and minimum space is occupied, since the stacks can be placed very close to each other. The low cost of storing items in stacks is counterbalanced by the limited accessibility of the items. To see the latter, suppose that boxes (A, B, C, and D) of equal size are stored in a stack as in Fig. 1. Box A is on top of the stack and it can be retrieved very easily. It is not the same with any of the other boxes, though. Retrieval of box B requires first the removal of box A. Similarly, to retrieve box C requires removal of boxes A and B and to retrieve box D it takes the removal of A, B, and C first.

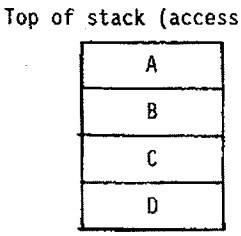


Fig.1 A Stack with Four Items

The apparent reason for the inconvenience in retrieving the items of a stack is the one point of access to the item through the top of the stack only. If item B in Fig.1 needs to be retrieved before item A, then item A has to be removed to allow retrieval of B. After B is removed, A can be placed back on the stack. In a situation like this, item A is said to be overstowed. The temporary removal from and placement back onto the stack of item A is called rehandle or rearrangement of item A. We can describe the problem in terms of storing and retrieving of the items; if, at any moment, these sequences do not obey the "last-in-first-out" rule, then overstowage occurs.

The main theme of this paper is the minimization of overstowage within the different contexts in which it occurs. The objective is to choose when and what items to rearrange in order to minimize the total number of item rearrangements over a certain period.

Definition 1 An item of a stack is overstowed when it blocks the retrieval of another item scheduled to be retrieved while the former is still in the stack.

Definition 2 Rearrangement (or rehandle) of an item is the temporary removal from and replacement back onto the stack of that item (source of cost).

Overstowage in Containership operations

The work of this paper has been motivated by an application in the maritime field. The containers on board a containership are placed in stacks. There are stacks below and above the deck of the vessel. Access to these stacks is possible only from the top of them. The mere existence of stacks on board creates "potential" for overstowage. The nature of containership operations contributes to the latter. Containerships visit ports at which they pick up and deliver containers. If the ports a vessel visits are numbered as 0, 1, 2, .. M, M+1, in general, there are c_{ij} containers going from port i to j.

If the storing and retrieving sequences of the containers are not identical, then occurrence of overstowage is very likely. In fact, overstowage would be

unavoidable, had only one stack to be used on board. The existence of more than one stack makes the situation less severe in terms of the number of overstowed containers. Nevertheless, independently of how many stacks exist on board, minimization of overstowage is our objective.

Particularly in containership stacking operations, stowage of containers is not arbitrary. That is, there exists sets of rules, guidelines, and constraints which must be met. Such are:

i. Stability constraints due to stability requirements of the vessel.
ii. Strength requirements of the vessel's structure. This translates into a requirement for the weight distribution along the vessel.
iii. Cargo placement constraints (cargo in refrigerated containers).
iv. Cargo adjacency constraints due to regulations about hazardous cargo.
v. Container strength constraints.
vi. Container stability constraints etc.

The above list is by no means exhaustive. However, it indicates the complexity of the problem, if all constraints are to be considered explicitly. In fact, the containership case is the most intricate among the occasions in which overstowage occurs. In most other cases, very few or none of the above constraints apply.

Overstowage is present in all circumstances involving stacking operations. Stacking operations in warehouses is one area in which the phenomenon of overstowage is very common. The operation of warehouses involves the arrival of items for storage and their subsequent retrieval later on. If the items are stored in stacks, overstowage is possible and should be minimized. A case similar to a containership visiting a series of ports comes up when a truck visits a series of locations where it loads and/or unloads boxes stored in stacks. Of course, the scale of the problem is smaller (fewer stacks) and most of the previous constraints do not apply. In fact, the concept of stacking does not require the stack to be physically vertical. Any linear arrangement with only one-point access can be modeled as a stack.

Overstowage problems are of a combinatorial nature. Consequently, their solution requires techniques from the field of discrete optimization. Although the general overstowage problem is difficult to solve (Aslidis 1989), special or restricted versions of it can be solved efficiently.

Overstowage problems: Literature Survey

It appears that very little has been published on this problem. The main contributions come from the maritime field. The first researchers who dealt with the problem are W. C. Webster and P. Van Dyke [1970a, 1970b]. Their approach is aimed at the loading/unloading process, but it could be extended to include the allocation and storage of containers. The stability constraints of the vessel are among the primary factors considered, while placement of containers is of secondary importance. The authors report that a small number of trials showed good results ("qualitatively good results") as far as the water ballast and overstowage cost are concerned.

Scott and Chen [1978] attempted another heuristic approach to the same problem. They adopted three heuristic rules which are used to implicitly satisfy the constraints. Containers are aggregated into homogeneous groups based on some container characteristics (such as type, length, height, weight, strength, and destination) and also on placement restrictions. Among the advantages of their approach is the consideration of many constraints, but they also treat minimization of overstowage as a secondary objective.

Shields [1984] followed a different path to solving the container stowage problem. The basic idea in this approach is the random generation and evaluation of many different possible loadings. The criterion through which a specific loading is generated is selected randomly among a set of criteria. The evaluation

of the loading is done by imposing penalties each time an increase in the container handling cost occurs or each time a constraint is violated. The three top (less costly) solutions are approved. The algorithm commences the loading of containers destined to the next port using the three selected loadings as starting points. Again, the three best loadings are chosen. The procedure is repeated at each subsequent port. At the final port, the less costly solution, as well as the resulting intermediate loadings are adopted.

Aslidis [1984] examined a simplified version of containership operations. It was assumed that a vessel visits a series of ports, at which she only picks up containers (one destination problem). Overstowage is not a factor except when some constraint is violated, in which case already stored containers need to be rearranged. A heuristic approach is proposed aiming mainly at satisfying the trim and metacentric height requirements with the minimum rearrangement costs. In another work, Aslidis [1989] provides analytical solution to the one-stack overstowage problem. All the containers are placed onto a single stack and no other constraints apply. This case represents a pure overstowage problem and should be contrasted to situations in which the overstowage cost also depends on the assignment of containers to stacks (multistack problem). The algorithm runs in polynomial time ($O(M^3)$), where M is the number of ports to be visited, and it is used as part of many algorithms for the multi-stack case. This paper solves the one-stack overstowage problem with constraints imposed on the vertical center of weight of the stack. First, we review the unconstrained case algorithm (Aslidis 1990).

OSOP: Definitions and Assumptions

Stacks can be defined as one-dimensional storage systems with one access point. In this paper, we employ uncapacitated stacks, that is, stacks with no limit on the items stowed. We call top of the stack the end of it closer to the access point and bottom of the stack the least accessible end.

The problem is defined as follows: a vessel (containership) is scheduled to visit a series of ports 0,1,2,...,M-1, M, M+1. The vessel can only carry one stack of containers, all of which are assumed to be of equal size. It is assumed that the vessel arrives at port 0 empty. At port 0, and at all subsequent ports up to port M, she picks up containers shipped to other ports of the series. For example, at port 0, she picks up containers going to ports 1, 2,...,M and M+1. If c_{ij} denotes the number of containers going from port i to port j, then the container shipments can be represented by a lower triangular matrix, called the shipment matrix. We deal only with deterministic shipment matrices, that are known at the beginning of the trip.

Definition 3 Containers with the same origin and destination belong to the same group of containers. There are N=(M+1)(M+2)/2 groups. Containers with the same destination belong to the same type of containers. There are (M+1) types (1,2,...M,M+1).

Definition 4 A stack is in "in-order" condition if the containers of the stack are placed in ascending order of destination from top to bottom. A stack is in "out-of-order" condition when it is not "in-order".

Assumption 1 It can be assumed without loss of generality that the stack is "in-order" as the vessel arrives at port 1 (or equivalently, she is empty of containers at port 0).

At each port, the vessel delivers some containers and picks up some others. Along with the above, there might be yet some other containers that must be temporarily unloaded because they block the discharge of the containers destined to the port. These containers need to be placed back onto the stack along with the "new" ones; they are overstowed containers.

Our objective is to minimize the number of overstowed containers during the trip of the vessel. To that extent, we might need to rearrange some containers at a port at which they do not block any other container in order to save a larger number of rehandles in later ports. That is, we look for a rearrangement policy (P) which results in the minimum number of additional rehandles of containers. The rearrangement policy should specify what containers of the stack should be rearranged at each port and how they should be placed back onto the stack at every port. Let $R_P(C)$ be the number of rearrangements that result if we apply policy P on the shipment matrix C, and let R(C) correspond to the optimal policy $P^*(R(C) = R_{P^*}(C))$. We will also refer to $R_P(C)$ as the overstowage cost.

A Recursive Algorithm for the OSOP

It is proven (Aslidis 1990) that the optimal policy is expressed as an array with M components satisfying the condition P(i) >= i, i=1,..M. P(i) = k indicates that at port i all containers destined to ports i+1, i+2,..,.k must be rearranged, These containers (and the containers having destination ports k+1 to M+1 which block them) must be removed from the stack, join the new shipments that originate at port i at the quay, and then be placed back onto the stack "in-order". In addition, it can be shown that for all (i,j) such that i < j and P(i) >= j, it should be P(i) >= P(j).

A subproblem PR(i,j) for the same port series, 0 to M+1, and shipment matrix, C, is defined as one in which the vessel visits ports i, i+1, i+2,...,j, and facing a shipment matrix in which all containers originating from port 0 to i are shipped from port i, and, all containers destined to ports j to M+1 assume delivery at port j. Let V(i,j) represent the minimum overstowage cost of problem PR(i,j), and r(i,k,j) be the number of rearrangements (rearrangement cost) at port k of containers of type k+1,..,j-1, for the problem PR(i,j) [$V(i,j) = R^{P(C \text{ of } PR(i,j))}$]. Then, it is shown (Aslidis 1989, 1990) that:

Theorem 1 The minimum overstowage cost and the corresponding optimal rearrangement policy of the one-stack overstowage problem, for an initial empty stack visiting a series of ports 0,1,2,...M,M+1 and facing a shipment matrix C can be found in $O(M^3)$ by solving the following recursive equation,

$$V(i,j) = \min_{i+1 <= k <= j-1} \{ V(i,k) + r(i,k,j) + V(k,j) \}, \quad \begin{matrix} i = 0,1,..M \\ j = i+1,..,M,M+1 \end{matrix} \qquad (1)$$

and $V(i,i+1) = V(i,i+2) = 0, \quad i = 0,1,...M$

Transformation 1 Given a rearrangement policy, P, with the properties discussed above (not necessarily optimal), we can define a shipment matrix C', such that it results in zero overstowage cost. This is based on the observation that when a group of containers, say (i,j), gets rearranged at port k, it is treated no differently from the group (k,j) from that point on. This allows us to replace the containers of group (i,j) with equal number of containers of groups (i,k) and (k,j). By construction the transformed shipment matrix results in no overstowage; however, the number of containers shipped (sum of all c'_{ij}) is greater than that of the original matrix exactly by the overstowage cost of the rearrangement policy, P.

THE ONE-STACK OVERSTOWAGE PROBLEM WITH PLACEMENT CONSTRAINTS

As a first step towards solving the containership problem, we solve the one-stack one with placement constraints. It is useful to see how such constraints can be implemented, particularly in the single stack case for which a complete analytical solution is possible.

The only placement constraint which is relevant to the one stack case is the stability constraint. This constraint refers to the center of weight (hence, c.o.w.) of the containers in the stack. In ship operations, this is the well

known GM requirement, that is the c.o.w. of the stack should be low enough (GM high enough) so the vessel does not capsize.

It is clear that we must now treat each container separately, because each one has its own weight. Let n_i be the number of containers to be picked up at port i, and n the total number of containers handled,

$$n_i = \sum_{k=i+1}^{M+1} c_{ik} \qquad n = \sum_{i=0}^{M} n_i \tag{2}$$

Stability Constraints: The One-Stack One-Destination Case

Since this is an operational constraint it has to be satisfied at each port. Let r_i be the maximum permissible value for the c.o.w. of the stack upon her departure from port i. The value of r_i depends among others on the displacement of the vessel, her volume distribution under the waterline, her vertical distribution of weight and the minimum required metacentric height (GM). In this section we ignore how the r_i's are calculated and we assume them as given. Let also x_i denote the vertical c.o.w. of the stack at the time of departure from port i. Then the stability constraints can be written as

$$x_i \leq r_i, \; i = 1,2,\ldots,M \tag{3}$$

It is quite obvious that since (3) must be observed, it is very rational to place the heavier containers among those picked up at each port, lower positions in the stack. If we follow this policy from the first port, where the stack is assumed empty, then the profile of the vessel in terms of the weight distribution looks as in Fig. 2.

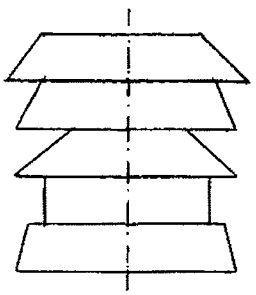


Fig. 2 Vertical Weight Distribution of Stack After Port 4

The above policy does not result in a single container rearrangement as long as constraints (3) are satisfied. If some of (3) are not met, then we must move heavy containers to lower positions to satisfy them.

Since all containers of the stack have the same destination, the effect of rearrangements made at early ports of the sequence (that is the decrease in x_i) is observed at later ports, too. For example, it may be optimal to perform rearrangements at port 2 in order to satisfy the stability constraint at port 5. In other words, if x_k is greater than r_k and must be reduced by at least g_k ($=x_k-r_k$), it suffices to reduce any combination of the x_i's ($i=1,\ldots k$) by a total of g_k. We would like to achieve that by doing the minimum number of container rearrangements.

Let us formally define g_k as the amount by which (3) is violated at port k, if no rearrangement is done at any port.

$$g_k = \max(0, \; x_k - r_k) \;, \; k=1,2,\ldots M \tag{4}$$

We define $b_k(m)$ as the maximum possible improvement (decrease) of x_k, with at most m container rearrangements performed at ports 0,1,2,...,k-1,k. Also let $\underline{S}_{k,m}$ denote the stack profile at port k, corresponding to $b_k(m)$. $\underline{S}_{k,m}$ is a vector with as many components as the total number of containers onboard (=n). Each component holds the weight of the container placed in the corresponding positions of the stack (the n^{th} component corresponds to the bottom position).

Since the effects of a rearrangement are permanent then $b_k(m)$ can be calculated as

$$b_k(m) = \max_{l} \{b_{k-1}(m-l) + f_{k,m}(l)\} \ , \ m=1,2,\ldots B_k \qquad (5)$$

where, l may assume values from 0 to $L_k = \min(m, \sum_{i=0}^{k-1} n_i)$

B_k is equal to $\sum_{i=0}^{k} (k-i)*n_i$

$f_{k,m}(l)$ is the improvement in x_k at port k, with l additional rearrangements given that m-l rearrangements have been performed at ports 1 to (k-1).

and, $b_0(m) = 0$, for all m, $b_k(0) = 0$, k=1,...M. (6)

Let us now show how $f_{k,m}(l)$ can be calculated. Given that we know that m-l rearrangements have been optimally performed at ports 1 to (k-1) (resulting in a total improvement in the c.o.w. of $b_{k-1}(m-l)$, we also know $\underline{S}_{k-1,\ m-l}$. Then, performing l additional rearrangements simply means that we take off from the stack the top l containers, which we place back along with the group of new containers (that is a total of n_k+l) in decreasing order of weight. The difference between the c.o.w. of the stack as calculated above from the c.o.w. that would result, did we not perform the l rearrangements is the value of $f_{k,m}(l)$.

Lemma 1 Functions $b_k(m)$, k=1,...m; m=0,1,...B_k are correctly calculated by using the recursive equation (5) and the boundary condition (6). The time it takes to compute $b_k(m)$ is $O(M^2n^3)$.

Let us now examine how we can take advantage of the method to compute $b^k(m)$'s to satisfy those of (4) which are violated. The values of g_k, k=1,...M dictate by how much the c.o.w. of the stack should be decreased up to port k. Then, by solving

$$g_k = b_k(m), \ k=1,\ldots,M \qquad (7)$$

we find the minimum number of required rearrangements up to port k to achieve the desired reduction of x_k. Let m_k denote this value of m. From (7) we find that,

$$m_k = b^{-1}{}_k(g_k), \ k=1,\ldots,M \qquad (8)$$

As long as m_k is known, the only thing that remains to be found is how m_k is distributed over the ports 1 to (k-1). This information can be found by retrieving the value of l that achieves the maximum in the corresponding recursive equation (5), and then working backwards in a recursive manner. The information about the above mentioned value of l can be stored at the same time (5) is evaluated and consequently retrieved in O(1) time. If t^*_k stands for the optimal number of rearrangement at each port, k=1,...,m, that satisfy (after performed) constraints (3), it must be

$$\sum_{i=1}^{k} t^*_i \ >= m_k \ , \quad k = 1, \ \ldots, M \qquad (9)$$

Knowing m_k helps reducing the number of states in the next stages of the recursive algorithm (i.e. calculation of $b_{k+1}(m')$). Specifically if we know that m cannot be less than m_k, and consequently l cannot be larger than m-m* we can reduce both the number of values of $b_{k+1}(m)$ to be computed and the number of terms to be compared. Algorithm GM-1 implements the above analysis to solve the one-destination one-stack problem. Theorem 2 below summarizes the result.

Algorithm GM-1

Input: - (M+1) groups of n_i containers, i=1, M with destination port M+1
- The weight of each container
- r_i, i=1,...M, the upper limit for the c.o.w. of the stack after port i

Output: - A rearrangement plan expressed in the number of top containers of the stack to be rearranged at each port, so that r_i's are observed. Containers are always pushed onto the stack in decreasing order of weight.

Step 1 - Initialization-boundary conditions
Set $b_0(m) = 0$, m=1,...n
$m_0 = 0$
for i=1 to M do
calculate x_i, $g_i = \max(0, x_i - r_i)$;

Step 2 - Basic recursion
For k=1 to M do
for $m=m_0$ to B_k
for $l=0$ to $B_k - m_0$
compute $f_{km}(l)$;
compute $b_k(m) = \max_l [b_{k-1}(m-l) + f_{km}(l)]$
$l_k(m) = l^*$ such that $b_k(m) = b_{k-1}(m-l^*) + f_{km}(l^*)$
$m_k = b^{-1}_k(g_k)$;

Step 3 - Constraint checking
For k=1 to M do
if $g_k \leq b_k(m_{k-1})$ then $m_k = m_{k-1}$;

Step 4 - Distribution of rearrangements
$t_m = l_m(m_M)$
for k = (M-1) down to 1 do $t_k = l_k(m_{m+1} - t_{k+1})$;
END.

Theorem 2 Algorithm GM-1 correctly solves single-stack single-destination overstowage problems with stability constraints in $O(M^2n^3)$ time. The space requirements of algorithm GM-1 are $O(Mn^2)$.

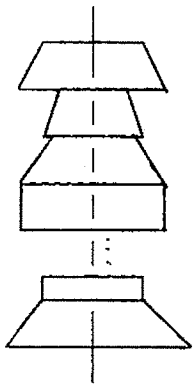
The One-Stack Overstowage Problem with Stability Constraints

Our approach in this section is going to be similar to the one in the previous section. In fact, the formulation of the problem in terms of the satisfaction of the stability constraint is going to resemble the one-destination case. This is achieved through the following steps.

First, we solve the OSOP as if no stability constraint exists. Let $\underline{P}^*$ be the optimal policy and $R^*(C)$ the corresponding minimum rearrangement cost. At this point we perform the transformation of the shipment matrix described above as Transformation 1 in reference to policy $\underline{P}^*$. Recall that this transformation is always done in conjunction with a specific rearrangement policy. The result is a shipment matrix C′ of which the minimum overstowage cost is zero and the optimal policy is the one with no rearrangements at all ($P^{*\prime}(i)=i$). However, the number of containers shipped under matrix C′ is greater than under C by exactly the overstowage cost of the policy P*. According to Transformation 1, if n and n′ are the number of containers handled under shipment matrices C and C′ correspondingly it holds that $n' = n + R_{P^*}(C)$. By definition P*, and consequently, C′, correspond to the minimum overstowage cost. Any deviation from

them is going to increase the latter. Thus, we must satisfy the stability constraints with the minimum deviation from the above.

Since the transformed shipment matrix results in zero overstowage, the vertical weight distribution of the stack looks as shown in Fig. 3. Again we assume that containers within each group are placed in decreasing order of weight.



Destination	Origin
K	5 (5 < K)
K+1	5
K+2	4
K+2	3
M	1
M+1	0

Fig. 3 Weight Distribution of Stack After Port 5

The definition of r_k, x_k, and g_k is the same as in the one-destination case, that is x_k, k=1,...,M, is the vertical c.o.w. after port k, if no rearrangements are performed under shipment matrix C'. (Remember that from now on we always refer to C'.) The stability constraints can be written again as

$$x_i \leq r_i \ , \ i=1,\ldots,M \qquad (3)$$

One of the properties of the one-destination case is that the effect of rearrangements at earlier ports last for the entire sequence. In the multidestination case though, when a group with some intermediate destination is delivered, any improvement in the c.o.w. due to rearrangements involving this group in the previous ports is lost. This is unfortunate because $b_k(m)$ cannot be defined in the same manner as before, since its value depends on what containers of those rearranged are still on board at port k.

As we mentioned a few paragraphs above, there is zero overstowage cost under the shipment matrix C', if the stability constraints are ignored. It can be proven that C' can have a total of (M+1) groups at most. This is so because groups with the same origin have (obviously) different destinations and groups with later origins can have no later destinations than groups from earlier ports. So, the difference "earlier destination-origin" decreases at least by one, as the vessel moves to the next port. The latter is a consequence of the zero overstowage condition (in the absence of the stability constraints).

Fig. 4 shows how the origin and destinations of the groups may look. What is missing from the figure is groups of the form (i, i+1). We are going to take care of these groups soon; for the time being we ignore them because they are too "temporary", that is they have a very short presence on board (just for one port).

Let us take the example of Fig. 4 and determine which groups are on board along every leg of the sequence (assume that port (M-1) comes directly after 6).

Leg	Groups
0-1	(0,M+1), (0,M)
1-2	(0,M+1), (0,M), (1,M-1), (1,6)
2-3	(0,M+1), (0,M), (1,M-1), (1,6), (2,6), (2,5)
3-4	(0,M+1), (0,M), (1,M-1), (1,6), (2,6), (2,5), (3,5)
4-5	(0,M+1), (0,M), (1,M-1), (1,6), (2,6), (2,5), (3,5)
5-6	(0,M+1), (0,M), (1,M-1), (1,6), (2,6)
6-(M-1)	(0,M+1), (0,M), (1,M-1)
(M-1)-M	(0,M+1), (0,M)
M-(M+1)	(0,M+1)

As it is evident from the above different groups of containers are present at different legs of the trip, however, if we reorder the legs then it appears as if the vessel only picks up containers having the same destination. For the above example the reordering has as follows.

Leg	Groups On Board
M-(M+1)	(0,M+1)
0-1,(M-1)	(0,M+1), (0,M)
6-(M-1)	(0,M+1), (0,M), (1,M-1)
1-2	(0,M+1), (0,M), (1,M-1), (1,6)
5-6	(0,M+1), (0,M), (1,M-1), (1,6), (2,6)
2-3	(0,M+1), (0,M), (1,M-1), (1,6), (2,6), (2,5)
3-4, 4-5	(0,M+1), (0,M), (1,M-1), (1,6), (2,6), (2,5), (3,5)

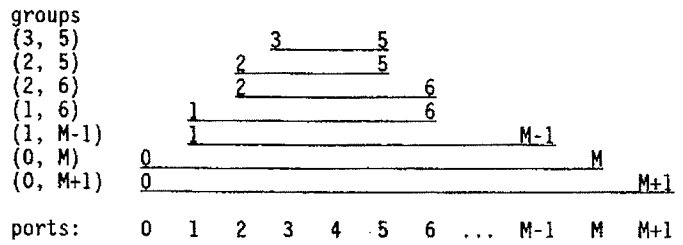


Fig. 4 Present Groups in a Zero Overstowage Shipment Matrix

A more careful examination of the above ordering reveals that it is done in an increasing origin-decreasing destination manner of the top group of containers of the stack in that particular leg. The similarities to the one-destination case become now clear. In principle an equation like (5) can be written for this case, too, with the functions $b_k(m)$ similarly defined. There are still some differences though that are not as simple. These are:

(i) The existence of groups of the form (i, i+1), i=0,1,...m, that is groups that stay on board only for one leg.

(ii) The complication in computing functions $f_{k,m}(1)$. This complication is due to the different destinations that some groups of containers may have. For example, in terms of the previous example, if at port 2, we "mix" one - the lightest - container of group (2,6) with group (2,5), it will cost us one rearrangement. If we "mix" one of (1,6)'s with group (2,5), it will cost us two rearrangements - one at port 2 and one at port 5.

From now on we always refer to the reordered trip. In doing so we restore the very important property of the one-destination case that the effects of rearrangements at the early ports last until the end of the trip. The latter is true for rearrangements involving all groups except from groups of (i,i+1) type, the effects of which are short-lasting.

Let us now see how we can account for the two differences from the one-destination case discussed above. We deal first with the second one. In fact, it is straightforward to implement the requirement of different costs in the

calculation of functions $f_{km}(l)$. We must simply keep track of the origin and destination of each container (along with its weight), and, when a container already on board is "intermixed" with the group of new containers (and there is always one such group in the reordered trip), the resulting number of rearrangements is one, if the container and the group have common origin or destination (in terms of the original trip), and, two, otherwise. This becomes obvious by looking at Fig. 5 below.

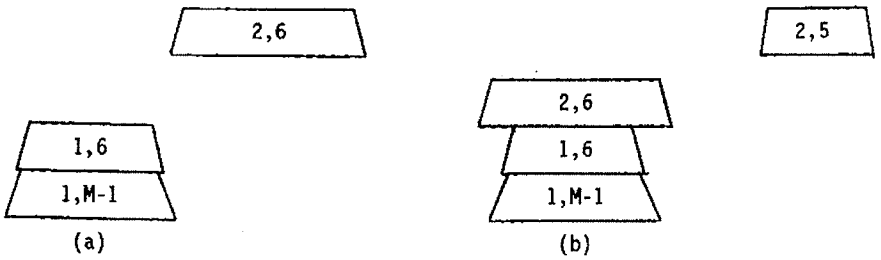


Fig. 5 Calculation of $f_{km}(l)$ in the Multi-destination Case

In Fig. 5(a) group (2,6) is the newcoming group. If we "mix" one container of group (1,6) with (2,6)'s, we "pay" only for the rearrangement of the (1,6)-container at port 2. If we "mix" an (1,M-1)-container with those of group (2,6) we "pay" for one rearrangement at port 2 and one at port 6 of the (1,M-1)-container. Even if the (1,M-1)-container is already "mixed" within the (1,6)'s we "pay" for two rearrangements (at port 2 and 6) in addition to the one "paid" at port 1 and which has been accounted for. Similar observations can be made in Fig. 5(b).

Definition 5 The reordered port sequence of an initial one with C' (transformed C under the optimal rearrangement policy) is one in which the vessel picks up the groups of containers in increasing order of origin and, among those with the same origin, decreasing order of destination. It may have up to (M+2) ports.

Lemma 2 In the reordered port sequence gains from rearrangements in early ports are maintained throughout the (reordered) port sequence.

Lemma 3 In calculating the $f_{km}(l)$ functions for the reordering port sequence, we must count twice the rearrangements that do not result in "mixing" (i.e. overstowing) containers of groups with the same origin or destination.

Based on the above results we can define (always in reference to the reordered port series), $b_k(m)$ as the maximum improvement in x_k if m rearrangements are performed, or more accurately charged, along ports 1 to k. Then we can write

$$b_k(m) = \max \{ b_{k-1}(m-l) + f_{km}(l) \} , \quad m = 1,2, ..,B_k \tag{10}$$

where now m and l stand for the rearrangement cost (at most twice as much as the number of rearrangements), and

$$\begin{aligned} l &\quad \text{may assume values from 0 to } L_k = \min(m, 2* \textstyle\sum_{i=0}^{k-1} n_i) \\ B_k &\quad \text{is equal to } 2* \textstyle\sum_{i=0}^{k} (k-i)*n_i \\ f_{k,m}(l) &\quad \text{is the improvement in } x_k \text{ at port k, with l additional rearrangement cost given that m-l rearrangements have been performed at ports 1 to (k-1).} \end{aligned} \tag{11}$$

Conditions (6) hold as well.

So if the groups (i,i+1) are ignored, we can solve the recursive equation (10)

along with (11), and (6) and get an optimal rearrangement policy which satisfies the stability requirement. The latter process has three possible outcomes.

(a) All stability requirements are satisfied and the solution is optimal even if groups (i,i+1) are considered.
(b) All stability requirements are satisfied but a better (cheaper) solution is possible if groups (i,i+1) are considered.
(c) Some stability constraints are not satisfied.

In any event, it appears that we should check how much the consideration of groups (i,i+1) affects the solution. This is mandatory in case (c); also, it is worth examining whether and to what extent the optimal solution changes even when the stability constraints are satisfied.

Definition 6 Groups (i,i+1), i = 0,1,..,m are called one-leg groups.

A preliminary analysis indicates that no significant savings should be expected through rearrangements of the containers of groups (i,i+1), simply because the effect of their rearrangement lasts only for one leg of the trip, while rearrangements of the other groups last for the remaining part of the trip (again, referring to the reordered port sequence). Figure 6 shows the g_i's as well as the resulting change in x_i, $d(x_i)$, after rearrangements of containers except one-leg ones are performed (it refers to the "original" port series).

A methodology to use the one-leg groups to satisfy the stability constraints is described below. The main idea is to reduce one or some of the g_i's that are positive; let $d(g_i)$ be such a decrease in one g_i. Then we solve the problem without groups (i,i+1) but with reduced g'_i ($=g_i-d(g_i)$). Any remaining required decrease in x_i's, if any, is going to be provided by local rearrangements of one-leg groups. The following definition is going to make notation simpler.

Let $C_k(m)$ be the cost of satisfying the stability constraints in ports 1,2,...k with m rearrangements performed on containers of groups other than one-leg groups. That is $C_k(m)-m$ is the minimum number of rearrangements involving one-leg groups that are required to satisfy the stability constraints. Then we can write:

$$c_k(m) = \min_{l=0,..,\sum_{i=0}^{k-1} n_i} \{ c_{k-1}(m-l) + l + h_k(m-l,l) \} \qquad (12)$$

where

$h_k(m-l,l)$ is the cost of rearrangements involving group (k,k+1) of port k in order to satisfy the stability constraint, given that (m-l) rearrangements not involving one-leg groups have been performed at ports 1 to (k-1) and l at port k,

and, $h^*_u(m)$ corresponds to the value of l that achieves the minimum.

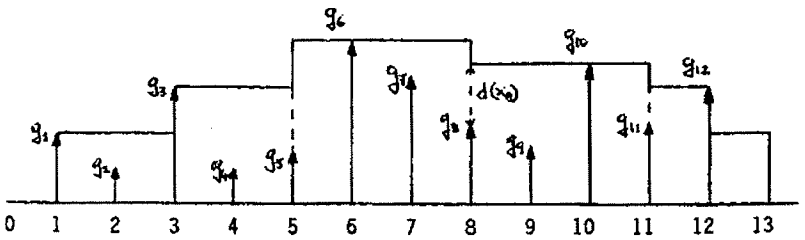


Figure 6 r_i's and Resulting dx_i's, After Solution Without Groups (i,i+1) - Original Port Series

The stack profile as the vessel arrives at port k is known, since we have assumed that (12) has been solved up to port (k-1). Then, knowing the number of rearrangements of containers of groups other than (k,k+1), we can construct the

profile of the stack as leaving port k. Yet we have not considered any rearrangements of containers of the (k,k+1) group. In fact, we are going to perform as many of those as required to satisfy the stability constraint at port k. If the constraint is already satisfied, then $h_k(m\text{-}1,1) = 0$. If the constraint cannot be satisfied then $h_k(m\text{-}1,1) = \infty$. In all other cases, $h_k(m\text{-}1,1)$ is the rearrangement cost corresponding to the rearrangements required to achieve the desired remaining decrease in x_k.

At port 0, where there is only one group on board, there is no way to change x_i by rearrangement; so the boundary condition for $c_k(m)$ is

$$C_1(m) = m+h_1(0,m) \ , \ m=0,1,\ldots,n \tag{13}$$

Equations (12) and (13) along with the way of calculating $h_k(m\text{-}1,1)$ constitute a recursive algorithm, that evaluates the minimum rearrangement cost to satisfy the stability constraints of the full one-stack overstowage problem. It is interesting to notice that this recursion is the "dual" of the one presented in algorithm GM-1. The former minimizes rearrangement costs, while the latter maximizes rearrangement benefits.

We must deal now in a more systematic way with the transformation of the original to the "reordered" port sequence. The recursive algorithms developed above refer to the "reordered" port sequence which, as we mentioned earlier, can contain up to (M+2) ports. However, if the shipment matrix contains less than (M+1) groups other than one-leg groups the "reordered" port sequence will consist of less than (M+2) ports. The example of Fig. 4 represents such a case. Two legs of the original series, 0-1 and (M-1)-M correspond to the same leg of the "reordered" port sequence. If, for some reason, r_0 and r_{M-1} of the original series are different, then we use the maximum of them in running (12), (13), and (6) - that is when the one-leg groups are ignored. This is sufficient, since the profile of the stack is the same in the two legs and we satisfy the stronger requirement. In the case that more than two legs share the same leg of the "reordered" port sequence, the maximum r_i is again used.

The above requirement translates slightly differently in the "dual" recursion, (12) and (13). Since $h_k(m\text{-}1,1)$ measures the cost to satisfy the stability constraint, we must account for its satisfaction along each leg of the trip. That is, again in terms of the above example, we must account for the cost of satisfying the stability constraints along leg 0-1 and (M-1)-M. These costs may well be different, because (i) r_0 and r_{M-1} may be different, and/or (ii) one-leg groups (0,1) and (M-1, M) may contain different numbers and weights of containers. The above observations are taken into account by replacing (12) by

$$c_k(m) = \min_{0<=l<=L_k} \{ c_{k-1}(m\text{-}l) + l + \sum_{j=1}^{J_k} h^j{}_k(m\text{-}l,1) \}, \ k = 2,3,..,M' \tag{14}$$

where J_k is the number of legs of the original trip that correspond to the k^{th} leg of the "reordered" port sequence, and (13) by

$$c_1(m) = m + \sum_{j=1}^{J_1} h^j{}_1(0,m), \quad m = 1,2,\ldots,n \tag{15}$$

If M' is the last port of the "reordered" port sequence, then the optimal solution is

$$c_{M'}(m^*) = \min_m \{ c_{M'}(m) \} \tag{16}$$

Recursive equations (14), (15), and (16) obviously solve the one-destination problem. This can be seen easily if we consider only groups of type (i, M+1), i=0,1,...M. Then we observe that no reordering of the port sequence is required, and, that all rearrangements contribute one unit to the rearrangement cost (because all the containers have the same destination) as it happens with the one-destination problem. Of course, in the latter, there are no one-leg groups. This does not mean that functions $h_k(m\text{-}1,1)$ are of no use. In fact,

they are used to indicate whether the stability constraints are satisfied. As it was mentioned when $h_k(m\text{-}1,1)$ were introduced, $h_k(m\text{-}1,1) = 0$ indicates that no rearrangements involving containers of group (k,k+1) are required to satisfy the constraints. Also, $h_k(m\text{-}1,1) = \infty$ indicates that the constraints cannot be satisfied. So in the special case of the one-destination problem function, $h_k(m\text{-}1,1)$ takes on two values, 0 or ∞ , depending on whether the stability constraint is satisfied or not. Then it turns out that $c_k(m)$ can be equal either to m or infinity. Nevertheless $c_k(m)$ still depends on 1 through its dependence on $h_k(m\text{-}1,1)$, which depends on 1.

After the above remarks that link our two approaches developed in this chapter we formalize our analysis by presenting an algorithm for solving the OSOP with stability constraints. This algorithm is called GM-OSOP. The correctness of the algorithm as proven above is summarized in Theorem 3.

Theorem 3 Algorithm GM-OSOP correctly solves the one stack overstowage problem with stability constraints in $O(M^2n^3)$ time.

Algorithm GM-OSOP

Input - Shipment matrix C for a given port sequence 0,1,...M,M+1.
- Weights of the individual containers.
- A vessel that can carry containers in one stack only.
- A requirement (r_i) for the c.o.w. of the containers of the stack at each port.

Output - A rearrangement plan that minimizes the total number of container rearrangements due to overstowage and to the constraint for the c.o.w. of the stack.

Step 1 - For the port sequence 0,1,...M,M+1 and shipment matrix C, compute the optimal policy P*(C), in absence of the c.o.w. constraints. Rearrangement cost R*(C).

Step 2 - Perform Transformation 1 on shipment matrix C for the optimal policy P*(C). Output a shipment matrix C resulting in zero overstowage under the no-rearrangement policy.

Step 3 - Apply the procedure described in Definition 6 and reorder the port sequence based on matrix C', (0,1,...M, M+1)

Step 4 - Run recursion (14), (15), and (16) for the "reordered" port sequence to get rearrangement cost $C_{M'}(m^*)$ and the distribution of rearrangements m^*_k, k=1,...M', of groups other than (k,k+1) and $h^*_k(m^*_k)$ of group (k,k+1), k=1,...M'.

Step 5 - Optimal rearrangement policy: P*(C), superimposed by m^*_k rearrangements of the top containers of groups other than (k,k+1) and by $h^*_k(m^*_k)$ ones involving group (k,k+1). Overall rearrangement cost: $R^*(C) + C_{M'}(M^*)$
END.

FURTHER RESEARCH ON OVERSTOWAGE PROBLEMS

In this paper we have dealt with the one-stack overstowage problem and extended the algorithm developed by Aslidis [1989, 1990] to take into account placement constraints on the containers in the stack. In particular, we have examined the

restrictions about the vertical center of weight. A new level of complexity is introduced when multistack problems are considered. There, each container or group of containers must be assigned to a stack. Also, when a container is rearranged, it does not have to return to the same stack. Furthermore, stacks may have finite capacity; or, the shipment matrix may be stochastic (the latter is very often the case).

As it appears that the multistack problems are significantly more difficult to solve with exact polynomial-time algorithms [Aslidis, 1989] the importance of developing "effective" heuristic methods increases. The solution of the one-stack problem is essential for the development of such heuristics. In fact, further modeling efforts of the one-stack case might be helpful in solving the multistack one.

Containership operations may result in overstowage not only on board a vessel, but also at the storage area of the port terminal. Consequently, one might attempt to minimize overstowage in the combined ship-port system.

Finally, the concept of stack can be generalized to include tree or multi-level stacks. Stacking containers on board a vessel requires indeed the multi-level stack concept, because we can stack containers above and below the deck. Other stacking models can be also defined (i.e. two-way access stacks) with application in various disciplines.

REFERENCES

Aslidis, A. [1984], "Optimal Container Loading", unpublished Master's Thesis, Massachusetts Institute of Technology.

Aslidis, A. [1989], "Combinatorial Algorithms for Stacking Problems", unpublished PhD Thesis, Massachusetts Institute of Technology.

Aslidis, A. [1990], "A polynomial-time algorithm for the one-stack overstowage problem", submitted to Operations Research.

Christofides, N. and Colloff, I. [1972], "The Rearrangement of Items in a Warehouse", Operations Research, 20.

Christofides, N., Mingozzi, A., and Toth, P. [1980], "Dynamic Loading and Unloading of Liquids into Tanks", Operations Research, 28, No. 3, pp. 633-649.

Gillmer, T.C. and Johnson, B. [1987], "Introduction to Naval Architecture", Naval Institute Press, 3rd edition.

Ladany, S.P. and Mehrez, A. [1984], "Optimal Routing of a Single Vehicle with Loading and Unloading Constraints", Transportation Planning and Technology, 8, pp. 301-306.

Shields, J.J. [1984], "Container Stowage: A Computer Aided Pre-planning System", Marine Technology, 21, No. 4.

Scott, D.K., and Chen, Der-San [1978], "A Loading Model for a Container Ship",November 15, Los Angeles, California (publication or presentation unknown).

Webster, W.C. and Van Dyke, P. [1970a], "Container Loading: A Container Allocation Model: I - Introduction, Background", presented at Computer-Aided Design Engineering Summer Conference, University of Michigan.

Webster, W.C. and Van Dyke, P. [1970b], "Container Loading: A Container Allocation Model: II - Strategy, Conclusions", presented at Computer-Aided Design Engineering Summer Conference, University of Michigan.