A Placement Heuristic for a Commercial Decision Support System for Container Vessel Stowage

Alberto Delgado
IT University of Copenhagen
Email:alde@itu.dk

Rune Møller Jensen
IT University of Copenhagen
Email:rmj@itu.dk

Nicolas Guilbert
Lund Institute of Technology
Email:nicolas@ange.dk

***Abstract*—Decision support systems have become a viable approach to tackle complex optimization problems. The combination of experts' know-how and efficient optimization algorithms can dramatically improve solution quality and reduce work time. Some of these systems rely on continuous interaction with their users and almost all require fast feedback from the optimization algorithms. We propose a placement heuristic that serves as the optimization component of a decision support system to interactively generate container vessel stowage plans, a complex problem with high economical impact within the shipping industry. Our experimental evaluation shows that the placement heuristic is fast enough for interactive optimization and produces solutions that are competitive with expert users.**

***Index Terms*—Stowage planning, linear programming, heuristic methods.**

I. Introduction

Nowadays a considerable part of trade goods are seaborne, with around 90% of all non-bulk cargo carried by container vessels. These vessels follow specific routes loading and unloading containers at different ports. A *stowage plan* assigns containers to load in a port to vessel slots. Good stowage plans save port fees, optimize use of vessel capacity, and reduce bunker consumption through a better distribution of cargo weight and shorter port stays, i.e., lower sailing speeds. Stowage Planners (SPs) produce these plans manually with the help of graphical tools (e.g., [1], [2]), but high-quality stowage plans are hard to generate with the limited support they provide. Plans for deep-sea vessels with capacities up to 15.000 Twenty-foot Equivalent Units (TEUs) are made just hours before the vessel calls the port, and it is difficult to do so satisfying stacking rules, stress limits, and stability requirements, while optimizing the use of resources (e.g., port cranes and vessel capacity). Additionally, the impact of the stowage plan for the current port in downstream ports needs to be considered as well. Liner shipping companies are in their infancy on using optimization-based decision support tools for this task. Moreover, most of the research on stowage planning during the past twenty years has focused on algorithms to automatically generate stowage plans (e.g., [3], [4], [5], [6]), limiting the role of expert users to result validation. Though some of these approaches generate good quality stowage plans, difficulties arise using them in practice. Stowage planning is full of corner cases disregarded in the models solved by automatic approaches. SPs need to be able to adapt automatic generated plans to specific situations, or be directly involved in the generation of the plans for stowage plans to be successfully used in practice.

Angelstow ([7]) is a decision support tool that assists SPs in the generation of stowage plans. The idea behind Angelstow is to provide SPs with a platform to exploit their know-how on stowage planning. To do so, the process of generating stowage plans is divided into two phases: *master planning*, where high level decisions that define the back-bone of the stowage plan are made (i.e., defining the discharge port of the containers to be stowed in sections of the vessel and setting upper bounds for the weight utilization of bays) and *slot planning*, where more specific decisions following those made in the previous phase take place (i.e., assigning the containers to specific vessel slots). The master and slot planning phases mimic the work process that an SP follows when generating a stowage plan. Angelstow provides an interface that allows SPs to define master and slot plans. To validate a master plan, however, an SP must carry out a considerable number of slot planning decisions, limiting the number of master plans that can be analyzed in the few hours an SP has available to stow a vessel. To address this issue, Angelstow provides SPs with the capability of turning master plans into stowage plans by automatically doing the slot planning. The generation of stowage plans then turns into an interactive process where the SP devises a master plan and Angelstow provides feedback by slot planning it and generating a stowage plan. If the resulting stowage plan is not satisfactory or complete, the SP can change or extend his or her master plan and let Angelstow generates a new stowage plan. To avoid that this interactive process becomes cumbersome, fast slot planning is a key factor. A desired runtime of a slot planning algorithm is around two seconds in average. In this paper, we introduce a 2-phase placement heuristic that generates a stowage plan from a master plan by slot planning a set of containers to be loaded in a single port. In the first phase, a Linear Programming (LP) model distributes the containers to subsections of the vessel. In the second phase, a greedy algorithm stows the containers of each subsection into vessel slots. We evaluate our placement heuristic on three real-life stowage plans. Each stowage plan corresponds to the stowage conditions of a large

978-1-4673-0793-2/12/$31.00

vessel of approximately 15.000 TEUs after leaving the last of six consecutive ports. The vessel is empty in the first port. To carry out our experiments, we use the same master plan decisions that were taken at each port by the SPs to make these stowage plans and let our placement heuristic generate slot plans for all ports.

In this way, we achieve a fair comparison between the slot plans generated by our placement heuristic and those generated by the SPs. Our results show that the runtime of the placement heuristic is reasonable (1.27 seconds in average). Moreover, it drops in average 0.58% and at most 1.89% of the containers to load and utilizes stack heights as efficient as the SPs. Additionally, we introduce two important practical features of stowage planning with strong impact on reduction in vessel capacity that, to the best of our knowledge, have not been considered in previous work. The remainder of the paper is organized as follows. Section II introduces background and describes the problem. Section III introduces Angelstow. Section IV describes related work. Section V presents our placement heuristic. Finally, Section VI and VII present the experiments and draw conclusions.

II. Background and Problem Statement

ISO containers transported on container ships are normally 8' wide, 8'6" high, and either 20', 40', or 45' long. *High-cube* containers are 9'6" high and are at least 40' long. Refrigerated containers (*reefers*) must be placed near power plugs. Containers with dangerous goods (*IMO containers*) must be placed according to a complex set of separation rules.

The capacity of a container ship is given in TEU. Figure 1 depicts the layout of a container vessel. The cargo space is divided into sections called *bays* and each bay is divided into an *on deck* and a *below deck* part by a number of *hatch covers*, which are flat, leak-proof structures. Each sub-section of a bay consists of a row of *container stacks* divided into *slots* that can hold a single 20' ISO container. Figure 2(a) and 2(b) show the container slots of a bay and a stack, respectively. A pair of slots in the same stack and tier are called a *cell*. A cell can hold a single 40' or 45', or two 20' containers. Cells with only one slot are called *odd slots* (e.g., bottom tier, Figure 2(b)). Stacks have height and weight limits. Two weight limits exists for each stack, one regarding the outer container supports and one regarding the inner supports. Limits on the inner supports are often the smallest, as the vessel structure in the middle of a stack is weaker. The inner supports are used only when 20 containers are stowed as depicted in Figure 2(b). When 20 and 40 containers are mixed in the same stack, only half of the 20 weight is considered to be supported by the outer supports, since the other half sits on the inner supports. Below deck, cell guides secure containers transversely. Containers on deck are secured by lashing rods and twist locks with limited strength. Thus, container weights must normally decrease upwards in stacks on deck. Moreover, lashing rods of 20' stacks must be accessible and stack heights must be under the vessel's minimum line of sight. 45' containers can normally only be stowed over the lashing bridge on deck.

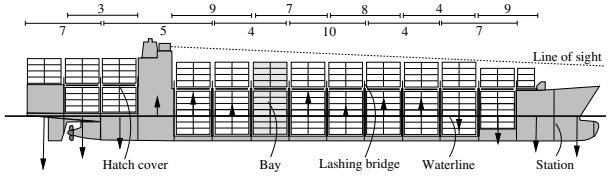

Fig. 1. The arrangement of bays in a small container vessel. The vertical arrows show an example of the resulting forces acting on the ship sections between calculation points (stations). Single crane work hours for adjacent bays are shown at the top.

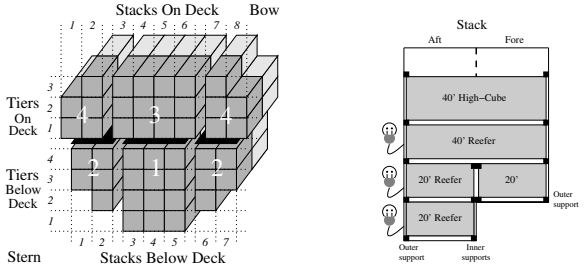

Fig. 2. (a) A vessel bay seen from behind. (b) A side view of a stack of containers. As depicted, power plugs are normally situated at bottom slots.

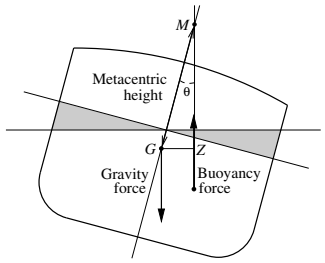

Fig. 3. Transverse stability.

A container ship must sail at even keel and have sufficient *transverse stability*. Figure 3 shows a cross section of a ship. For small inclination angles, the volume of the emerged and immersed water wedges (shaded areas) and thus the distance GZ are approximately proportional with the angle such that the buoyancy force intersects the center line in a fixed position called the *metacenter*, M [8]. For an inclination angle θ, the ship's uprighting force is proportional to $GZ = GM \sin\theta$. GM is called the *metacentric height* and the center of gravity G must be on the center line and result in sufficient GM for the ship to be stable. Maximum and minimum draft restrictions apply due to port depths, working height of cranes, and the propeller. The *trim* is the difference between the fore and aft draft and must be kept within a given span. For a station position p, the *shear force* is the sum of the resulting vertical forces on vessel sections (see Figure 1) acting aft of p, and the *bending moment* is the sum of these forces times the horizontal distance to them from p. Both of these stresses must be within limits. The vessel also has transverse bending moment (*torsion*) limits. Given the displacement and longitudinal center of gravity of a vessel, metacenter, draft, trim, and the buoyancy of each section of the vessel can be derived from hydrostatic tables. Ballast tanks distributed

along the vessel are used to modify displacement and center of gravity by pumping water in or out of the tanks.

Container ships in liner shipping companies transport containers between ports on a fixed cyclic route. It is the liner shippers and not the port terminals that are in charge of producing stowage plans. A *stowage plan* assigns the containers to load in a terminal to slots on the vessel and it is often sent to the terminal shortly before calling it.

In this paper, we focus on the generation of slot plans based on master plans defined by SPs. Master plans embed in their constraints the know-how of SPs on how to deal with complicated combinatorial objectives. In particular they must minimize *overstowage*. Overstowage happens when a container is stowed below a container destined for a later port. In master plans this may happen if the containers stowed on a hatch-cover are destined for later ports than the containers stowed under the hatch-cover. Such *hatch-overtowage* may cause many extra crane moves. A master plan must also distribute the crane moves evenly over the length of the vessel to minimize the makespan of the quay cranes. In addition, it must ensure seaworthiness of the vessel with respect to shear force and bending moments limits, as well as line-of-sight, trim, draft, and GM requirements.

A feasible slot plan must fulfill all constraints specified in its corresponding master plan together with the following rules:

- Stowed containers must form stacks (containers stand on top of each other. They cannot hang in the air).
- 20' containers can not be stowed on top of 40' containers.
- 20' and 40' stack weight capacities are satisfied.
- Stack height capacities are satisfied.
- No overstowage is accepted.
- Reefer containers must be stowed in reefer slots.
- 20' and 40' slot capacity constraints are satisfied.
- The full capacity of a cell must be utilized. It is not possible to stow a single 20' container in cells with capacity for two.

In addition, we follow the set of rules of thumb listed below to generate high quality slot plans:

- Containers are distributed evenly among stacks on deck.
- The number of used stacks is minimized in storage areas below deck.
- Wasted volume capacity is minimized.
- It is preferred to stow containers below deck, if possible.
- Heavy containers cannot be stowed on top of light containers on deck.

III. Angelstow

Angelstow provides an interface that allows SPs to define master and slot plans (although the latter can be generated automatically as well). In a master plan, an SP defines the discharge port of the containers to stow in a sub-section of a vessel bay called a *compartment*. Only a single discharge port is allowed for each compartment. This is seldom a limitation in practice since most vessels have sufficient compartments to allow only stowing containers of one discharge port. Moreover, due to the robustness of this approach with respect to avoiding overstowage in downstream ports, it is considered a good stowage practice to do so.

Automatic slot planning can be activated at any time, but only the compartments that the SP has defined a discharge port for will be slot planned. The SP can also set upper bounds on the total weight of containers stowed in the compartments. In this way, a partial master plan can be slot planned without the SP losing control of the overall weight distribution over the vessel. This is essential for achieving the draft, trim, GM, and stress force objectives of the master plan.

Recall that a slot plan is an assignment of the containers to load to specific vessel slots. A considerable number of slot plans must be carried out by an SP in order to determine whether it is possible to generate a stowage plan out of a master plan. SPs generate plans under high time pressure and uncertainty, since they receive the load list of containers a few hours before the vessel calls port, and last minute changes to the load list are allowed. Due to these conditions, the number of master plans that can be analyzed is very limited. To address this issue, Angelstow provides SPs with the capability of turning master plans into stowage plans by automatically doing the slot planning. The generation of a stowage plan then becomes the interactive process described next. An SP devises a master plan and, if needed, some slot planning decisions are made to complement it. Next, Angelstow performs the slot planning based on the master plan and slot planning decisions specified by the SP. Once Angelstow finishes, the SP can modify the stowage plan at will by performing slot planning decisions on it. In case it is not possible to adapt the stowage plan to what the SP considers a reasonable solution, the master plan can be changed and the process starts over again. To avoid this interactive process to become cumbersome, fast slot planning is a key factor, which is an issue addressed by the placement heuristic presented in Section V. We estimate that a time limit of two seconds for generating slot plans is necessary to efficiently support interactive optimization in Anglestow.

A multi-port view of stowage planning is important to reduce the negative impact of a stowage plan in downstream ports. A bad stowage plan can considerably reduce the vessel capacity for containers to load in upcoming ports. Angelstow provides a multi-port view (see Figure 5) where SPs can see the effects in downstream ports of the master plan they devise for the current one. Additionally, even though SPs focus their work on the current port, information available in load lists for downstream ports (i.e., forecasts) can be used to devise master plans for the downstream ports as well.

When the SP wants to check the feasibility of the master plans formulated for each of the ports a vessel calls, Angelstow generates slot plans for each port independently using the placement heuristic introduced in section V. Figure 4 depicts the process that Angelstow follows to generate slot plans for all the ports the vessel calls. At each port, the slot planning component (placement heuristic) uses as input 1) the master plan defined by the SP in Anglestow , 2) any slot

planning decisions made by the SP (e.g., requiring that certain containers are stowed in certain slots), and 3) a load list of containers for the port. The containers stowed by Anglestow in the previous ports have fixed positions and are represented as an extra set of slot planning decisions extending the ones made by the SP. Thus, the slot plan made for the current port becomes the release containers (onboard containers) of the next downstream port. A more sophisticated approach to multi-port slot planning would be to represent the problem in a single optimization model. This has not been done since 1) the impact is limited because most of the important decisions with respect to multi-port planning are at the master planning level, and 2) this may deteriorate the runtime performance.

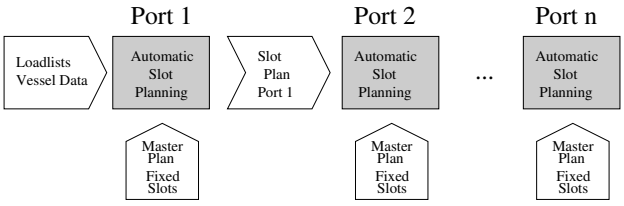

Fig. 4. Multi-port slot planning.

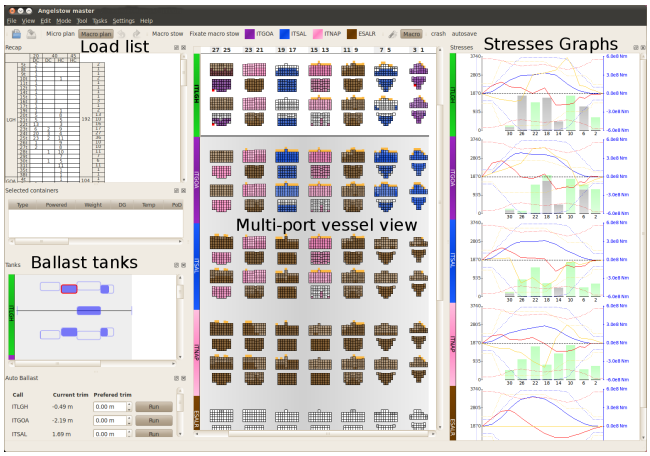

Fig. 5. Front view of Angelstow. To the left, a summary of the load list of containers and a view of the current state of the ballast tanks. In the center, the multi-port view of Angelstow. Each row represents a port that the vessel calls. To the right, the stresses graphs and weight distribution along the vessel bays.

IV. Literature Review

With the stowing of container vessels being a key process in the liner shipping industry, the problem of how to optimize this task has caught the attention of several researchers since the 1970s. Academic work can be divided into two main categories: single-phase and multi-phase approaches. Single-phase approaches have a plain representation of the stowage planning problem (e.g., stow containers into vessel slots). A common denominator among these approaches is to sacrifice model accuracy to achieve scalability. Early work in this category (e.g., [9], [10]) introduced heuristic approaches that solved simplifications of the stowage planning problem. These approaches sequentially refine solutions by applying placement heuristics, local search, and solving IP models. Later on, [11] introduced an IP model of the first accurate but intractable formulation of the stowage planning problem. Constraint programming [12], metaheuristics such as genetic algorithms (e.g.,[13], [14]) and simulated annealing [15], and more recently IP (e.g, [16], [17], [18]) have been used to solve still simplified versions of the stowage planning problem.

Multi-phase approaches decompose the problem hierarchically into two or more layers of abstraction (e.g., first distribute containers over vessel sub-sections, then stow them into vessel slots). These approaches are currently the most successful in terms of model accuracy and scalability. The first work of this type appeared in the early 1970s, where [19] introduced a 3-phase heuristic. Twenty years later, a model that includes several major aspects of the problem in a 2-phase approach was introduced in [20]. This approach solves the stowage planning problem for multiple ports. It uses a branch-and-bound algorithm for solving the first phase, where containers are distributed over sub-sections of the vessel, and applies a tabu search for stowing containers into vessel slots in the second phase. Other approaches that use similar decompositions include solving multi-port stowage planing with an iterative improvement approach based on the transportation simplex method [21], a bin-packing heuristic [22], and solving a mixed-integer program for the first phase and a constraint programming and constraint-based local search models for the second one [6]. 3-phase approaches include combinations of constructive heuristics, 0/1 IP, and metaheuristics (e.g., [4]) and heuristics combined with LS (e.g., [5]).

Multi-phase approaches developed by the industry include a multi-stage placement heuristic using a number of lower-bounds [3] and a combination of sequential LPs for distributing containers over vessel sections, and a hierarchy of IPs for stowing them into vessel slots [23].

Several software tools are available in the market for container stowage planning decision support (e.g., [1], [2], [24], [25]). All of them assist SPs with the generation of stowage plans by providing feedback on their slot planning decisions. None of them, however, stow containers automatically. To the best of our knowledge, our placement heuristic is the first attempt to use optimization techniques for interactive generation of stowage plans. Moreover, our approach addresses two new practical features of stowage planning with strong impact on the reduction of the vessel capacity that have not been considered in previous work: high-cube containers and odd slots.

V. Heuristic Approach to Slot Planning

In this section we introduce our 2-phase heuristic decomposition to generate slot plans. In the first phase, containers are grouped together by their features and distributed over the compartments by a Linear Programming (LP) model. A compartment consists of a subset of stacks not necessarily adjacent to each other, but within the same bay. Compartments are either above or under a hatch-cover. Figure 2(a) shows four

compartments within a bay. In the second phase of the heuristic, a greedy algorithm finds specific slots for all containers following the distribution dictated by the first phase. Currently, we limit our scope to consider containers that are *20'* and *40'* long and model reefer and high-cube characteristics.

A. LP Model

Here we present the LP model implemented in the first phase of our slot planning heuristic. The set of compartments of the vessel is defined as $\mathcal{L}$. Containers from the load list I are grouped together according to their features into a set of types $\mathcal{T}$. A type $\tau \in \mathcal{T}$ is a 5-tuple (l, h, r, w, p), where $l \in L = \{20, 40\}$ is the length of containers in feet, $h \in H = \{HC, DC\}$ is the height of the container (high-cube or dry cargo), $r \in R = \{RF, NR\}$ is the reefer properties of the container (reefer or non-reefer cargo), $w \in W$ is the weight class of the container defined by the weight of the container rounded to nearest integer ton, and $p \in P$ is the discharge port of the container, with P being the set of possible discharge ports from the load list of containers. Let $\mathcal{T}^{\alpha}$ be the subset of types with a particular attribute value (e.g., $\mathcal{T}^{NR}$ is the subset of all non-reefer container types), and $\mathcal{T}^{\neg\alpha}$ the subset of types without a particular attribute (e.g., $\mathcal{T}^{\neg d}$ is the subset of types with discharge port different than d).

The decision variables $x_l^{\tau} \in \mathbb{R}^+$ represent the number of containers of type $\tau \in \mathcal{T}$ stowed in compartment $l \in \mathcal{L}$. In this formulation we have dropped the integrality constraint over the decision variables. Given the number of containers that can be stowed in a compartment (up to a couple of hundred), and the small number of valid types per compartment seen in practice, the loss of precision in our solution by doing this is low and can be easily dealt with in the second phase of our placement heuristic. As seen in [6], the gain in computation speed by doing this is substantial.

Auxiliary variables are introduced for each compartment $l \in \mathcal{L}$. $y_l^{HC} \in \mathbb{R}^+$ represents the number of high-cube containers stowed in l above the high-cube killing limit, and $y^O \in \mathbb{R}^+$ the number of misused odd slots. We propose the LP model below for solving the problem of distributing types of containers to compartments in the first phase of our placement heuristic:

maximize

$$\sum_{\tau \in \mathcal{T}} \sum_{l \in \mathcal{L}} C_l^{\tau} x_l^{\tau} - C^{HC} \sum_{l \in \mathcal{L}} y_l^{HC} - C^O y^O \tag{1}$$

subject to

$$\sum_{\tau \in \mathcal{T}^{\neg d_l}} x_t^{\tau} = 0 \quad l \in \mathcal{L} \tag{2}$$

$$\sum_{l \in \mathcal{L}} x_l^{\tau} \leq I^{\tau} \quad \tau \in \mathcal{T} \tag{3}$$

$$\sum_{l \in \mathcal{L}^b} \sum_{\tau \in \mathcal{T}} w^{\tau} x_l^{\tau} \leq W_b^u \quad \forall b \in \mathcal{B} \tag{4}$$

$$\sum_{\tau \in \mathcal{T}} V^{\tau} x_l^{\tau} \leq S_l^V \quad \forall l \in \mathcal{L} \tag{5}$$

$$\sum_{\tau \in \mathcal{T}^{20}} x_l^{\tau} + \sum_{\tau \in \mathcal{T}^{40}} 2x_l^{\tau} \leq S_l^S \quad \forall l \in \mathcal{L} \tag{6}$$

$$\sum_{\tau \in \mathcal{T}^{20}} x_l^{\tau} \leq S_l^{20} \quad \forall l \in \mathcal{L} \tag{7}$$

$$\sum_{\tau \in \mathcal{T}^{40}} V^{\tau} x_l^{\tau} \leq S_l^{40} \quad \forall l \in \mathcal{L}^{NO} \tag{8}$$

$$\sum_{\tau \in \mathcal{T}^{20} \cap \mathcal{T}^{NR}} A_l^{NR} x_l^{\tau} \geq c_l^{ONR} \quad \forall l \in \mathcal{L}^O \tag{9}$$

$$c_l^{ONR} \leq A_l^{NR} S_l^{ONR} \quad \forall l \in \mathcal{L}^O \tag{10}$$

$$\sum_{\tau \in \mathcal{T}^{20}} A_l^R x_l^{\tau} \geq c_l^{OR} \quad \forall l \in \mathcal{L}^O \tag{11}$$

$$c_l^{OR} \leq A_l^R S_l^{OR} \quad \forall l \in \mathcal{L}^O \tag{12}$$

$$\sum_{\tau \in \mathcal{T}^{20}} x_l^{\tau} \geq \frac{c_l^{ONR}}{A_l^{NR}} + \frac{c_l^{OR}}{A_l^R} \quad \forall l \in \mathcal{L}^O \tag{13}$$

$$\sum_{\tau \in \mathcal{T}^{40}} V^{\tau} x_l^{\tau} \leq S_l^{40NO} + c_l^{ONR} + c_l^{OR} \quad \forall l \in \mathcal{L}^O \tag{14}$$

$$\sum_{\tau \in \mathcal{T}^{20}} w^{\tau} x_l^{\tau} \leq W_l^{20} \quad \forall l \in \mathcal{L} \tag{15}$$

$$\sum_{\tau \in \mathcal{T}^{20}} \frac{1}{2} w^{\tau} x_l^{\tau} + \sum_{\tau \in \mathcal{T}^{40}} w^{\tau} x_l^{\tau} \leq W_l^{40} \quad \forall l \in \mathcal{L}^{40} \tag{16}$$

$$\sum_{\tau \in \mathcal{T}^R} x_l^{\tau} \leq S_l^{RS} \quad \forall l \in \mathcal{L}^R \tag{17}$$

$$\sum_{\tau \in \mathcal{T}^R \cap \mathcal{T}^{20}} \frac{1}{2} x_l^{\tau} + \sum_{\tau \in \mathcal{T}^R \cap \mathcal{T}^{40}} x_l^{\tau} \leq S_l^{RC} \quad \forall l \in \mathcal{L}^R \tag{18}$$

$$\sum_{\tau' \in \mathcal{T}^{(w^{\tau'} \leq w^{\tau} \wedge teu^{\tau'} = teu^{\tau})}} x_l^{\tau'} \leq S_l^{\tau} \quad \forall l \in \mathcal{L}, \tau \in T \tag{19}$$

$$\sum_{\tau \in \mathcal{T}^{20}} x_l^{\tau} + y_l^O \geq S_l^O \quad \forall l \in \mathcal{L}^O \tag{20}$$

$$\sum_{d \in \mathcal{D}} \sum_{l \in \mathcal{L}^d} y_l^O - O^d \leq y^O \tag{21}$$

$$\sum_{\tau \in \mathcal{T}^{HC}} x_l^{\tau} - y_l^{HC} \leq S_l^{HC} \quad \forall l \in \mathcal{L} \tag{22}$$

The objective (1) maximizes the quality of the stowage plan. There is a weight C_l^{τ} defined for each decision variable x_l^{τ} that reflects preferences with respect to types of containers and compartments. Reefer containers are more profitable, therefore, we prefer to load reefer over non-reefer containers. Compartments below deck are preferred over compartments on deck due to accessibility, and heavy containers are preferred to be stowed below deck to increase stability of the vessel. Recall that high-cube containers utilize more volume capacity than dry cargo containers. Given the slot and volume capacity of compartment l, we define the high-cube killing limit of l, S_l^{HC}, as the number of high-cube containers that can be stowed in l such that its slot capacity can still be completely utilized. When the number of high-cube containers stowed in compartment l goes beyond S_l^{HC}, the slot capacity of l is reduced (*kill cells*) in order to provide the volume capacity necessary to stow the extra high-cube containers. We penalize the number of high-cube containers stowed in compartment l that are above S_l^{HC} (auxiliary variable y_l^{HC}) with cost C^{HC}. The number of misused odd slots in the vessel, represented by variable y^O, is penalized by cost C^O. Odd slots are at the bottom cells of a stack, and it is necessary to fill them up with *20'* containers in order to access the full capacity of the stack. Our model penalizes odd slots that are not being used when there are *20'* containers available in the load list to do so.

Constraint (2) restricts the containers to stow in compart-

ment l by only allowing containers of types with discharge port d_l, the discharge port for l defined by the SP during master planning. Constraint (3) ensures that we do not stow more containers of type τ in the vessel than those available from the load list I^τ. Let $\mathcal{B}$ be the set of bays in the vessel, $\mathcal{L}^b$ the set of compartments in bay b, w^τ the weight of the weight class of a container of type τ, and W_b^U the upper bound of the weight capacity for bay b that may have been defined by the SP during master planning. Constraint (4) restricts the weight of a bay according to the user preferences.

Containers consume volume capacity from compartments depending on their type. For instance, *20'* containers need a single slot, while *40'* containers need two. Since high-cube containers are all *40'* they consume slightly more than two slots. Constant V^τ represents the volume consumed by type τ. Constraint (5) restricts the volume of containers stowed in $l \in \mathcal{L}$ to be within its volume capacity limit S_l^V. The volume capacity of compartment l is calculated by adding up the volume capacity of all stacks in l. Since stack volume capacity is continuous, the volume capacity of l could be erroneously increased by the fractional capacity of the stacks. This allows extra containers in l that will not be possible to stow since containers can not be split into pieces. To overcome this issue, we constrain the containers assigned to compartment l to be within the slot capacity of the compartment, S_l^S, with constraint (6).

Due to physical limitations in the vessel, there are slots that can hold only containers of a specific length. We limit the number of *20'* containers stowed in compartment l to be within the *20'* capacity of the compartment, S_l^{20}, with constraint (7). Similarly, we constrain the *40'* volume capacity of compartments without odd slots, $\mathcal{L}^{NO}$, with constraint (8).

When present, odd slots are mainly found at the bottom cells of a stack. For the *40'* volume capacity of a stack with odd slots to become available, the odd slots must be filled in with *20'* containers. Thus, in compartments with odd slots, *40'* volume capacity is increased proportionally to the utilized odd slots.

Constraint (9) sets the auxiliary variable c_l^{ONR} to be the extra *40'* volume capacity gained by stowing *20'* non-reefer containers in the odd slots of compartment l. Notice that we must distinguish between reefer and non-reefer containers since a reefer container only can be placed in reefer odd slots. The set of compartments with odd slots is defined by $\mathcal{L}^O$. The average volume gained by stowing a single *20'* non-reefer container in an odd slot is defined by A_l^{NR}, and it is equal to the average volume of the stacks above the odd slots. This gain is limited by the number of non-reefer odd slots in l, S_l^{ONR}, by constraint (10). In a similar fashion, constraints (11) and (12) set the auxiliary variable c_l^{OR} to be the extra *40'* volume capacity gained by stowing *20'* containers in l (reefer and non-reefer) in an odd reefer slot. To avoid counting volume gains produced by the *20'* non-reefer containers twice, constraint (13) forces the number of *20'* containers to be at least as many as the containers used to calculate the volume gains when stowing *20'* in odd slots. Finally, constraint (14) limits the number of *40'* containers in odd compartments to be within the *40'* volume capacity. The *40'* volume capacity of compartment $l \in \mathcal{L}^O$ is defined as the volume capacity of the compartment stacks with no odd slots, $S_l^{40\,NO}$, plus the increment in *40'* volume capacity given by stowing *20'* non-reefer containers, c_l^{ONR}, and the increment by stowing *20'* reefer containers, c_l^{OR}, in the odd slots of l.

Recall that the outer support points of a stack hold all the weight of *40'* containers and half of the weight of *20'* containers, while the inner support points hold the other half of the weight of *20'* containers. We can then define the weight limit of *20'* containers in a compartment as W_l^{20}, which is equal to twice the weight capacity of inner support points. Similarly, we can define W_l^{40} to be the weight limit of *40'* containers taking into account that this capacity is also reduced with half the weight of *20'* containers. The corresponding constraints are constraints (15) and (16), respectively. For the reefer capacity, constraints (17) and (18) limit the number of reefer containers that can be stowed in a compartment. Constraint (17) limits the number of reefer containers with respect to the number of reefer slots, S_l^{RS}, in compartment l. There are cases, however, where both slots in a cell are reefer slots. In this situation, if a *40'* container is placed in this kind of cell, two reefer slots are used instead of only one. Constraint (18) addresses this issue by constraining the number of reefer containers with respect to the number of reefer cells in the compartment, S_l^{RC}.

In a pre-processing stage, we compute an upper bound, S_l^τ, on the number of containers of type τ that can be stowed in compartment l. This upper bound is defined as the maximum number of containers of type τ that can be stowed in l when relaxing all constraints but those enforcing weight and TEU capacity. Thus, the number of containers of types with the same length as τ (i.e., demanding the same TEU capacity) and with greater or equal weight must be smaller or equal to S_l^τ. Constraint (19) enforces this.

Having S_l^O defined as the number of odd slots in compartment l, constraint (20) bounds the auxiliary variable y_l^O, which is the number of odd slots not used in l. Let $\mathcal{L}^d$ be the set of compartments where the SP decided to stow only containers of types with discharge port d, and O^d the number of odd slots in $\mathcal{L}^d$ that we know in advance can not be used due to lack of *20'* containers. Constraint (21) bounds the auxiliary variable y^O which is the number of odd slots not used in the vessel due to misplacement of *20'* containers.

As mentioned before, when the number of high-cube containers in a compartment l surpasses S_l^{HC}, the high-cube killing limit, it is necessary to reduce the slot capacity of the compartment in order to stow the extra high-cube containers. The relation between the number of killed cells and the extra number of high-cube containers that can be allocated in l is not linear, i.e., several high-cube containers can be stowed in l by killing a single cell. Any linearization attempt on measuring the slot capacity reduction due to killed cells in l can be reduced to measure the number of high-cube containers above S_l^{HC} stowed in l. Constraint (22) bounds auxiliary variable

y_l^{HC} to the number of high-cube containers in compartment l above S_l^{HC}.

B. Greedy Algorithm

We introduce a greedy algorithm that stows containers in specific slots in such a way that the set of stacking constraints and rules of thumb described in Section II are satisfied and optimized, respectively. Our greedy algorithm (Algorithm 1) works as follows: Given a set of containers C and a set of stacks S, for each container $c \in C$, the algorithm finds the best stack $s \in S$ where c can be stowed. Containers are stowed bottom-up in stacks, therefore, when looking for a feasible stack for c, the greedy algorithm only checks the lowest empty cell of each stack. Once the best stack has been found, c is stowed in the lowest empty cell of the stack. In case it is not possible to find a feasible stack to stow c, the container is added to the list of unstowed containers, uc. Once all $c \in C$ have been stowed, or attempted to be stowed, the list of unstowed containers is returned.

Algorithm 1 GreedySelection(C, S)

```
uc ← ∅ // Set of unstowed containers from C
repeat
    Select container c from C
    bs ← NULL // Best stack for container c
    for s_i ∈ S do
        if it is possible to stow c in s_i then
            update bs to s_i if s_i is better than bs
        end if
    end for
    if there is no feasible stack for c then
        uc ← uc ∪ {c}
    else
        Stow container c in stack bs
    end if
    C ← C − {c}
until C ≠ ∅
return uc
```

It is important to notice that if the stowing of container c in the lowest empty cell of stack s fulfills all constraints described in Section II , such stow will not make infeasible any of the previous stows of containers in cells of stack s. A stack s_i is preferred over another stack s_j for stowing container c, if by stowing container c at the lowest empty cell of stack s_i, more of the rules of thumb described in Section II are satisfied than by stowing c at the lowest empty cell of stack s_j. Containers are selected from C following a specific order to avoid breaking stacking constraints. To avoid having *20*' containers on top of *40*', *20*' containers are selected first. Among containers with the same length, reefer containers are selected before non-reefers since reefer plugs are usually placed at the bottom of the stacks. At last, when containers have the same length and reefer capabilities, heavier containers are selected first. Stowing containers bottom-up in stacks eliminates the possibility of having containers hanging in the air.

We use the greedy algorithm introduced above to slot plan the vessel, following the distribution dictated by the first phase solution. First, we distribute the containers from load list I among the compartments following the solution from the first phase. Let q_l^τ be the number of containers of type τ assigned to compartment l during the first phase of the placement heuristic. For each compartment $l \in \mathcal{L}$ and type $\tau \in \mathcal{T}$, we select from the load list q_l^τ containers of type τ to stow in compartment l. Since q_l^τ could be a continuous value, unit containers are stowed in the compartments with the largest fractional part. Once containers are distributed to compartments, we stow independently each $l \in \mathcal{L}$ with the greedy algorithm.

Finally, we use a second run of the greedy algorithm to stow some of the dropped containers. Let uc_l be the set of containers assigned to compartment l that the greedy algorithm could not stow, $U = \{c_i | l \in \mathcal{L}, c_i \in u_l\}$ be the set of all containers the greedy algorithm was unable to stow, and S^V be the set of stacks of the vessel. We change to a vessel perspective and use the greedy algorithm to stow the set of containers U in stacks from S^V. The containers that are not stowed at this point are reported as unstowed containers to the SP.

VI. Experiments

In order to evaluate the placement heuristic introduced in this paper, we use three real-life stowage plans made by SPs on a vessel with approximately 15.000 TEU capacity. Table I specifies the features of the three stowage plans. ID represents the identifier of the instance, while SWP shows the identifier of the stowage plan, and P the port number in the schedule. O*20*' and O*40*' are the number of *20*' and *40*' containers already onboard the vessel when it calls the port, respectively. L*20*' and L*40*' represent the number of *20*' and *40*' containers to load when the vessel calls the port, respectively, and U the TEU utilization of the vessel when leaving the port.

Each stowage plan is a stowage condition of the vessel after visiting the last of six consecutive ports. The vessel is assumed to arrive empty to the first port. We extract the master planning decisions made at each port in order to achieve these stowage conditions. This will not include all master planning decisions, as we do not have information about containers sent internally between the ports. However, SPs have slot planned the master planning decisions that we can extract from the stowage plans, making them suitable for comparing the performance of our placement heuristic against the SPs' work. In the few compartments where containers with different discharge ports are mixed, something our placement heuristic does not allow, we assign the discharge port of the majority of containers to the compartment.

As depicted in Figure 4, slot plans are generated independently for each port in a schedule. Given that our goal is to evaluate the performance of the placement heuristic introduced in Section V, we generate a data set of 18 instances, one for each of the six ports in the three real-life stowage plans, and analyse them independently. All the experiments were run on a

TABLE I
Instance Overview

ID	SWP	P	O*20*'	O*40*'	L*20*'	L*40*'	U
1		1	0	0	182	939	2026
2		2	182	939	219	750	3779
3		3	401	1689	563	1439	7220
4	1	4	964	3128	175	412	8219
5		5	1192	3540	228	685	9817
6		6	1420	4225	625	747	11936
7		1	0	0	101	531	1163
8		2	101	531	141	731	2766
9		3	242	1262	403	1416	6001
10	2	4	645	2678	407	614	7636
11		5	1052	3292	199	827	9489
12		6	1251	4119	827	1141	12598
13		1	0	0	102	864	1830
14		2	102	864	242	684	3440
15		3	344	1548	473	1276	6465
16	3	4	817	2824	116	304	7189
17		5	933	3128	144	613	8529
18		6	1077	3741	652	894	10969

TABLE II
Experimental Results.

ID	SWP	P	D *20*'	D *40*'	LP(s)	H(s)	KS	KH
1		1	2	0	0.30	0.06	8	8
2		2	3	1	0.74	0.05	9	10
3		3	1	0	1.86	0.09	6	5
4	1	4	1	0	1.01	0.03	8	5
5		5	4	12	1.34	0.04	12	14
6		6	26	0	2.21	0.06	7	15
7		1	1	0	0.14	0.03	9	9
8		2	3	2	0.57	0.04	18	14
9		3	3	0	1.47	0.08	9	16
10	2	4	3	0	1.23	0.05	1	6
11		5	3	1	1.23	0.04	9	14
12		6	29	0	2.97	0.08	7	9
13		1	0	0	0.23	0.05	19	23
14		2	4	0	0.75	0.06	11	17
15		3	3	0	1.66	0.10	6	16
16	3	4	2	0	0.58	0.03	8	3
17		5	5	0	1.08	0.04	13	13
18		6	18	0	2.57	0.07	40	20

Ubuntu 10.4 system, with Intel Core 2 Duo, 2.7 GB of RAM, and CPLEX 12.1.

Table II presents the results of our experiment. ID represents the identifier of the instances, SWP the identifier of the stowage plan and P the port number on the schedule. D*20*' and D*40*' present the number of *20*' and *40*' containers dropped by the placement heuristic, while LP(s) and H(s) are the time in seconds used by the LP model and the greedy heuristic. At last, KS and KH are the number of cells killed in the slot plans generated by the SPs and the placement heuristic, respectively.

We measure three key factors: First, the response time of our placement heuristic, since one of our goals is to keep the generation of slot plans within two seconds in average. Second, the number of containers that could not be stowed. Third, the number of cells killed in a slot plan, which is one of the most common reasons for sub-utilizing the volume capacity of a container vessel.

Our placement heuristic managed to generate slot plans for each instance in 1.27 seconds in average, a number well within our desired time for the interactive generation of slot plans (two seconds, in average). The longest time was 3.05 seconds (instance 12), while the fastest one was 0.17 seconds (instance 7). There are two important things to notice about the run times presented in Table II. First, it is easy to see that the placement heuristic spends most of the time solving the LP model. Second, we can observe that there is a direct relation between the number of containers to stow and the time used by the heuristic to generate the slot plans. For each stowage plan, the ports with the greatest number of containers to stow, ports 3 and 6, are also the ports where the slot planning takes the longest. It is important to notice as well that the number of onboard containers do not seem to have a strong influence on runtime, since, otherwise, we would see runtimes increasing from port 1 to 6 in all stowage plans.

The maximum amount of dropped containers is 1.89% of the containers in the load list (instance 6), and 0.58% in average, a considerable low fraction. A discrepancy between the number of *20*' and *40*' containers being dropped can be easily spotted. In 4 out of 18 instances *40*' containers were dropped, and with respect to the total dropped containers for all instances, only 16 out of 129 are *40*' long. There are two main reasons for this discrepancy. The first reason is that since there are no odd slots in our test vessel, the number of *20*' containers distributed to a compartment by the LP must be even. Given that cell capacity must be completely utilized, no cell with an odd number of *20*' containers assigned to it is allowed. Thus, our greedy algorithm will drop at least one *20*' container when an odd number of *20*' containers has been assigned to a compartment by the LP. When we use our greedy heuristic to attempt stowing all dropped containers, most of the stacks are already full or partially utilized with *40*' containers on top, making it impossible to stow *20*' containers. The second reason relates to the fact that reefer plugs are usually available only at bottom tiers of the compartments of the vessel. In our set of stowage plans, a considerable number of reefer containers, most of them *20*' long, are loaded in port number six. Since more that 50% of the capacity of the vessel is already utilized at this point, we have already used most of the reefer slots at the bottom tiers of the vessel, even by stowing non reefer containers. Additionally, most of the few remaining free reefer slots are in stacks partially filled with *40*' containers.

The number of killed cells in our set of instances average 11.1 per instance for the slot plans made by SPs, and 12.1 for those made by our placement heuristic. In 10 out of 18 instances SPs managed to kill less cells than our heuristic, while our heuristic killed less cells in 5 instances. Both, SPs and our heuristic, killed the same number of cells in the remaining three instances. SPs killed 20 cells more than our placement heuristic in the worst case (instance 18), while the placement heuristic killed 10 more cells than the SPs (instance 16). Even though the results of this experiment favours the SPs, it is important to notice that SPs have an accumulated expert knowledge of years generating stowage plans, and we are getting close to match that knowledge (difference on the averaged killed cells is only one) with a heuristic that stows 1968 containers in a vessel partially full (63%) in three seconds.

It is important to notice when comparing the number of killed cells by the SPs and our placement heuristic that we have two simplifications in the generation of slot plans compared to SPs: we treat 45' containers as 40' and we do not consider lashing constraints in on deck compartments. We still believe, however, that it is relevant to show how our heuristic performs compared to SPs in terms of number of cells killed, since the impact of considering 45' containers and lashing constraints will be minimal. Cells are only killed in compartments below deck due to the physical limitations imposed by hatch covers. Let us consider what will happen if we allow our placement heuristic to handle 45' containers. First, all 45' containers will be stowed on deck since on deck compartments are the only ones with 45' capacity. This could affect the number of killed cells compared to the ones from the current slot plans if a 45' dry cargo container moved on deck is replaced by a 40' high-cube container. This is, however, very unlikely since most, if not all, 45' containers are high-cube containers. Any container our heuristic uses to replace the 45' containers going on deck will be of the same height or smaller and will not increase the number of killed cells.

Roughly speaking, lashing constraints can be seen as constraints that restrict the weight distribution of containers stowed in on deck compartments. When the GM increases, these constraints force the number of light containers stowed on deck to be increased. Thus, in cases where all heavy containers are high-cube, we are forced to stow them below deck and an increment in the number of killed cells would be expected. This increment, however, will not be dramatic. As part of the cost values in the objective function of our LP model, we reward stowing heavy containers in compartments below deck, thus, we are generating stowage plans that already stow as many light containers on deck as possible.

VII. Conclusion

This paper introduced a placement heuristic that serves as the optimization component of a decision support system to interactively generate container stowage plans. Our heuristic uses the know-how of SPs represented in a master plan to stow containers from a load list into vessel slots. Our experiments showed that with a 1.27s of runtime average, our placement heuristic can be used for interactive optimization. Additionally, the slot plans produced are competitive with expert users. For a container vessel of approximately 15000 TEUs, our placement heuristic managed to only drop at most 1.89% of the load list and 0.58% in average, and only killed 1 cell more in average than the expert user.

As future work we plan to make our slot plans more accurate by handling 45' containers and including lashing constraints. We are also interested in studying the impact of reducing the granularity of the weight classes in the LP model's runtime and in the percentage of containers dropped by the greedy heuristic.

Acknowledgments: We would like to thank the Angelstow team at Ange Optimization for their extensive support of this work. This research is sponsored in part by the Danish Agency for Science, Technology, and Innovation.

References

[1] Navis, "PowerStow," www.navis.com, 2011.
[2] Interschalt, "Seacos," www.interschalt.de, 2011.
[3] M. Gumus, P. Kaminsky, E. Tiemroth, and M. Ayik, "A multi-stage decomposition heuristic for the container stowage problem," in *Proceedings of the 2008 MSOM Conference*, 2008.
[4] D. Ambrosino, D. Anghinolfi, M. Paolucci, and A. Sciomachen, "An experimental comparison of different heuristics for the master bay plan problem," in *Proceedings of the 9th Int. Symposium on Experimental Algorithms*, 2010, pp. 314–325.
[5] M. Yoke, H. Low, X. Xiao, F. Liu, S. Y. Huang, W. J. Hsu, and Z. Li, "An automated stowage planning system for large containerships," in *In Proceedings of the 4th Virtual Int. Conference on Intelligent Production Machines and Systems*, 2009.
[6] D. Pacino, A. Delgado, R. M. Jensen, and T. Bebbington, "Fast generation of near-optimal plans for eco-efficient stowage of large container vessels," in *Proceedings of the Second International Conference on Computational Logistics (ICCL'11)*. Springer, 2011, pp. 286–301.
[7] Ange Optimization, "Angelstow," http://ange.dk/main/angelstow, 2012.
[8] E. C. Tupper, *Introdution to Naval Architecture*. Elsevier, 2009.
[9] D. Scott and D. Chen, "A loading model for a container ship," Matson Navigation Company, Los Angeles, Tech. Rep., 1978.
[10] A. H. Aslidis, "Optimal container loading," Master's thesis, Massachusetts Institute of Technology, 1984.
[11] R. Botter and M. A. Brinati, "Stowage container planning: A model for getting an optimal solution," in *Proceedings of the 7th Int. Conf. on Computer Applications in the Automation of Shipyard Operation and Ship Design*, 1992, pp. 217–229.
[12] D. Ambrosino and A. Sciomachen, "A constraint satisfaction approach for master bay plans," *Maritime Engineering and Ports*, vol. 36, pp. 175–184, 1998.
[13] Y. Davidor and M. Avihail, "A method for determining a vessel stowage plan, Patent Publication WO9735266," 1996.
[14] O. Dubrovsky and G. L. M. Penn, "A genetic algorithm with a compact solution encoding for the container ship stowage problem," *J. of Heuristics*, vol. 8, pp. 585–599, 2002.
[15] M. Flor, "Heuristic algorithms for solving the container ship stowage problem," Master's thesis, Technion, Haifa, Isreal, 1998.
[16] D. Ambrosino and A. Sciomachen, "Impact of yard organization on the master bay planning problem," *Maritime Economics and Logistics*, no. 5, pp. 285–300, 2003.
[17] P. Giemesch and A. Jellinghaus, "Optimization models for the containership stowage problem," in *Proceedings of the Int. Conference of the German Operations Research Society*, 2003.
[18] F. Li, C. Tian, R. Cao, and W. Ding, "An integer programming for container stowage problem," in *Proceedings of the Int. Conference on Computational Science, Part I*. Springer, 2008, pp. 853–862, LNCS 5101.
[19] W. C. Webster and P. Van Dyke, "Container loading. a container allocation model: I - introduction background, II - strategy, conclusions," in *Proceedings of Computer-Aided Ship Design Engineering Summer Conference*. University of Michigan, 1970.
[20] I. D. Wilson and R. P., "Principles of combinatorial optimization applied to container-ship stowage planning," *Journal of Heuristics*, no. 5, pp. 403–418, 1999.
[21] J. Kang and Y. Kim, "Stowage planning in maritime container transportation," *Journal of the Operations Research Society*, vol. 53, no. 4, pp. 415–426, 2002.
[22] W.-Y. Zhang, Y. Lin, and Z.-S. Ji, "Model and algorithm for container ship stowage planning based on bin-packing problem," *Journal of Marine Science and Application*, vol. 4, no. 3, 2005.
[23] N. Guilbert and B. Paquin, "Container vessel stowage planning, Patent Publication US2010/0145501," 2010.
[24] Mller+Blanck, "Capstan3," www.capstan3.com, 2011.
[25] Autoship Systems Corporation, "Autoship," www.autoship.com, 2011.