ISBN: 978-981-18-5852-9
WCSE 2022 Spring Event: 2022 9th International Conference on Industrial Engineering and Applications
doi: 10.18178/wcse.2022.04.198

Solving the Vessel Slot Planning Problem for 40' Containers Using a Multi-stage Rule-based Heuristic

Dalia Rashed [1.2], Amr Eltawil [1] and Mohamed Gheith [2.1]

[1] Department of Industrial and Manufacturing Engineering, Egypt-Japan University of Science and Technology (EJUST), Alexandria, Egypt.

[2] Production Engineering Department, Faculty of Engineering, Alexandria University, Alexandria, Egypt.

Abstract. Container vessel stowage planning is an important and influential decision problem that affects the containers' handling operations and the vessel's berthing time. The stowage planning problem is a complex one that can be solved with a successful decomposition approach, where the problem is decomposed into two stages. The slot planning problem is considered the second stage of that approach. A solution approach with multiple stages was developed, each stage is based on a rule-based heuristic, and additionally, each stage has its objective either to minimize/eliminate a constraint violation or, improve the objective function. The computational results elaborate on the efficiency and effectiveness of the proposed heuristic. The 127 examined industrial instances were all solved to optimality except one case.

Keywords: Stowage planning problem, Slot planning problem, Rule-based heuristics

1. Introduction

The trading of goods has increased globally since the emergence of standard containers six decades ago [1]. A container is a metallic box that allows for high stacks of containers to be formed [2]. Containerized shipping allowed for the expansion in vessel sizes, hence it facilitated the transportation of goods across the countries worldwide [3].

In 2019, container throughput was estimated to be over 802 million TEUs, accordingly, it is accounted for an increase of 2.3 percent from the previous year [4]. The COVID-19 pandemic had a detrimental influence on worldwide trade, according to Container Trade Statistics (CTS), however, container throughput climbed by 6.9% in September 2020 compared to September 2019. To face the high demand for a cost-effective mode of transportation large container vessels that can transport up to 20,000 TEUs are deployed [5].

A container vessel is a ship that transports containers with a small crew along their fixed cyclic route of ports, where loading and unloading movements of the containers to/from the vessel are performed using quay cranes at the container terminals. With larger container vessels, the number of these movements increases and gets complicated, subsequently adding to the time the vessel stays in each port (i.e. long berthing time) and decreasing the turnaround of the container vessel [2].

Since the operating cost of the shipping companies is partially determined by the port's fees, it is of great importance to manage effectively the complex activities in the port to reduce the vessel's turnaround time and the handling charges [7]. The containers' handling operations and the berthing time are affected by the effective arrangement of containers on board the vessel. This decision is called the stowage planning problem. Nevertheless, an effective stowage plan is a key to reaching higher productivity [8].

The container stowage planning problem is an NP-hard problem; it is the problem of assigning a set of containers of different types to a set of dedicated storage spaces within the vessel. This assignment should ensure several structural and operational constraints related to both the vessel and containers, to achieve different objectives one of which is minimizing the loading time [9].

In the past, stowage plans were manually generated by stowage coordinators. Recently, there has been an increasing interest in finding optimization algorithms to help the stowage coordinators to make better

decisions. This could be ascribed to the increase in the complexity of the problem and the importance of getting an optimum solution in a reasonable time.

Producing effective stowage plans in practice is a challenging decision due to the following: the limited time allocated to the stowage coordinators to make the decision, the thousands of loading/unloading moves required by the large container vessels, the complex interactions between the high-level stress limits and stability requirements of the vessel, and the low-level stacking rules. Furthermore, the practical requirement from the shipping companies is that the used algorithms should be fast with a runtime that is less than 10 minutes [6].

Since the stowage planning problem is a complex problem, solution approaches that implement an embedded hierarchical decomposition approach of the problem are commonly used. Such approaches mimic the work process of human stowage coordinators, thus, making the problem scalable and producing satisfactory results solving the real instances in reasonable times [10]–[15]. As shown in Fig. 1, the hierarchical decomposition solution approaches break the problem into two consecutive stages.

Initially, the multi-port master planning stage uses the input data, such as the load list (i.e., a list of containers to be loaded at the current port and a forecast for the later ones), and vessel characteristics to generate a master layout plan. This is accomplished by assigning each group of containers to specific locations in the vessel to satisfy the high-level stress and stability requirements. Furthermore, the next stage, the slot planning stage, assigns an exact position to each container within its location to satisfy the low-level stacking rules [2].

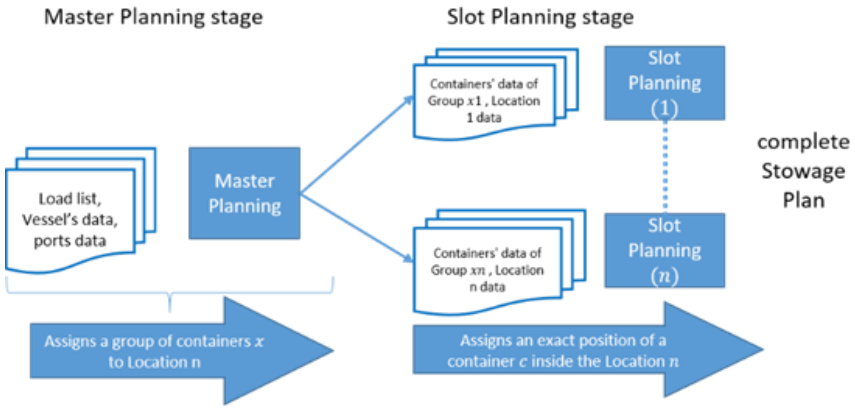

Fig. 1: The stowage planning decomposition into a master planning stage and a slot planning stage.

From the literature; as mentioned earlier, there is a practical requirement from the shipping companies that the whole stowage plan is generated in less than 10 minutes. Given that the first phase is time-consuming and that a large container vessel has about 100 locations. Subsequently, each independent slot planning problem needs to be solved within 1 second, this is translated into a practical constraint for this problem and thus increases the need for finding solutions approaches that are fast and efficient [16]. Additionally, slot planning is an NP-hard problem since it can deal with hundreds of containers which makes solving the problem a hard task [6].

The focus of this study is the slot planning phase while considering the below deck locations only, and the problem is denoted as the Container Stowage Problem for Below Deck Locations (CSPBDL). A heuristic based on a structured rule-based search to solve the problem is introduced. The heuristic has multiple stages, and each stage has the objective either to minimize/eliminate constraint violation or improve the objective function by following guided rule-based structures.

The rest of the paper is organized as follows, section 2 includes the description of the problem under study and section 3 presents the related literature. Subsequently, section 4 includes a detailed description of the proposed heuristic and section 5 elaborates the computational results and analysis. Finally, section 6 presents the conclusions and future work directions.

2. Problem Statement and Description

The objective of the detailed stowage plan is to find a specific cell/slot for each 40'/20' container within each location in the master plan while taking into consideration the stack constraints. It is important to note that there are different types and variations of containers, and they are characterized by some data: size: ISO containers are normally 8' wide, height (8'6" and 9'6"), length (20', 40' and 45'), weight, discharge port (i.e., where it would be unloaded) and whether it is a reefer container or not (i.e., a refrigerated container that needs an electric power supply).

The capacity of a container vessel is given in TEU, and the cargo space of a container vessel is divided into a number of subsequent bays as shown in the abstract side view of the vessel shown in Fig. 2a), each bay is divided by a hatch cover (represented by a red dotted line) into an on-deck and below-deck areas. The hatch cover is a flat, leak-proof structure that prevents the vessel from taking in water.

Each bay is a collection of vertical container stacks that allows for vertical stacking of containers and horizontal tiers, and it can be further considered as several locations which is a set of stacks that could be over or below the deck. As shown in Fig. 2b), the bay is divided into 4 Locations (L1, L2, L3, L4). This example also demonstrates the existence of blocked cells at the bottom (i.e., no container can be located in them) because of the shape of the vessel.

The intersection between the horizontal tier and the vertical stack forms a cell that can either hold a single 40' container or two 20' containers as shown in Fig. 2c). A cell consists of a fore slot and an aft slot. Some of these cells/slots have access to power plugs to provide electricity to the reefer containers, they are referred to as reefer cells/slots as shown in Fig. 2b). Each container stack has weight and height limit restrictions.

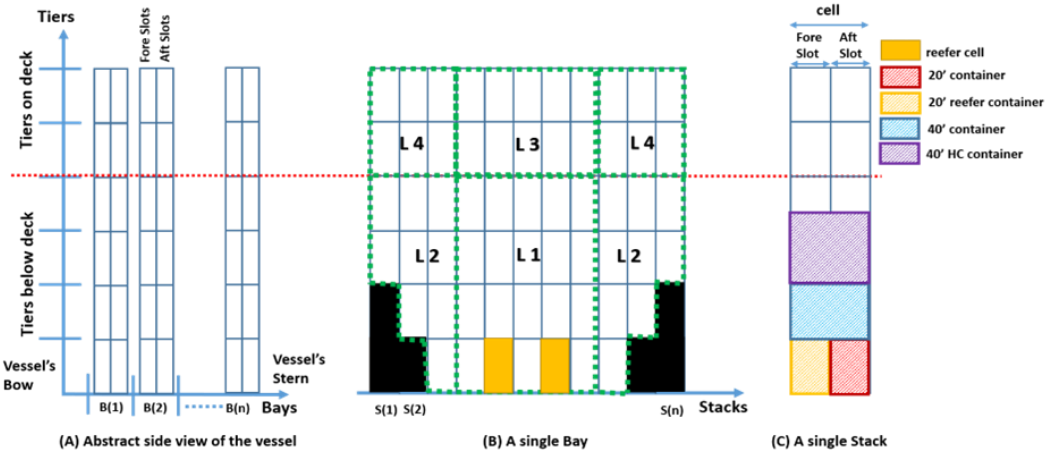

Fig. 2: The elements of the slot planning problem.

The problem under consideration is the problem of assigning a set of 40' containers to a set of below deck cells. These containers could be either standard (8'6" high), high cube containers (9'6" high), reefer, or not. It is important to note that the loading and unloading of the containers are performed by quay cranes, which can only access the top container of a stack.

The Container Stowage Planning Problem for a Below Deck Location (CSPBDL) is defined by the following constraints and objectives, furthermore, the IP model is found in [6],. A feasible stowage plan for a below-deck location must satisfy the following constraints:

- The containers assigned to a certain stack must be stacked on top of each other (i.e., no container can physically hang in the air).
- A 40' reefer container must be placed in a cell that has access to a power plug.
- The sum of the weights and heights of the containers that are allocated to a certain stack must conform to the stack weight/height limits.
- Already existing containers cannot be relocated.

The necessary sets and variables used in the objective function in equation (1) are defined in Table 1. Furthermore, the objective function is a scaler summation of a set of objectives (i.e., with different weights)

that are defined to evaluate the quality of the generated valid solutions that satisfy the above-mentioned constraints. The first term aims at decreasing the overstowage (i.e., a container is an overstowing container if it has a discharge port further than any of the containers which are located beneath it). The second term decreases the number of discharge ports in each stack while the third term decreases the number of used stacks and finally the last term decreases the number of non-reefer containers occupying reefer cells.

Table 1: The sets, parameters and decision variables used to calculate the objective function.

Sets and parameters	
C	Set of containers to be loaded, where $c \in \{1,2,\dots,C\}$
S	Set of stacks, where $s \in \{1,2,\dots,S\}$
D	Set of discharge ports, where $d \in \{1,2,\dots,D\}$
H_s	Set of cells in stack s
R_{hs}	Number of reefer plugs in cell h of stack s
R_c	Indicated whether Container c is reefer or not, where $R_c \in \{0,1\}$
Decision variables	
$O_c \in \{0,1\}$	Indicates whether container c is an overstowing container or not
$p_{sd} \in \{0,1\}$	Indicates whether stack s has a container with a discharge port d or not
$a_s \in \{0,1\}$	Indicates whether a stack s is empty or not
$E_{chs} \in \{0,1\}$	Indicates whether a container c is assigned to cell h of stack s or not

$$100 \sum_{c \in C} O_c + 20 \sum_{s \in S} \sum_{d \in D} p_{sd} + 10 \sum_{s \in S} a_s + 5 \sum_{s \in S} \sum_{h \in H_s} \sum_{c \in C} R_{hs} * E_{chs} (1 - R_c) \quad (1)$$

The economic aspects of a stowage plan are targeted mainly by the first objective while the other objectives are considered as the rules of thumb of the shipping industry; they allow for better stowage plans in the downstream ports of the vessel's voyage.

3. Related Work

The stowage planning problem is a variation from the container stacking problem and the container premarshalling problem [17], [18]. It has been tackled in the literature through two different approaches: the single-phase approach and the decomposition approach. The former approach deals with the whole vessel at once considering the container vessel having a set of available container storage areas and assign a specific container to a specific area as in [9], [19], [20], while the latter decomposes the problem into two successive phases, the multi-port master planning phase and the slot planning phase.

The single approach is appropriate for smaller vessels as it can produce fast solutions for a complete stowage plan for multiple ports, but for a very unconstrained problem (relaxed problem) where a lot of the real practical constraints are being discarded. On the other hand, the decomposition approach can be applied to vast variations of the problem while incorporating the practical constraints that face the real stowage planning problem. The second phase of the decomposition approach is the main concern of this work, and that is why the literature is limited to this phase only.

The Slot Planning Problem (SPP) for the below deck The Slot Planning Problem (SPP) for the below deck locations has been approached in [6], [16], [21]–[23]. The objective which has been tackled in these papers was to minimize a scalar objective function including four components; overstowage, number of stacks used, multiple destinations in a stack (i.e., avoid having a stack that has containers of different destination ports), and finally having a non-reefer container placed in a reefer slot to satisfy the stacking rules. The problems considered 20', 40', high cube, and reefer containers.

Delgado et al. [21], used a constraint programming approach to solve the problem, they tested a total of 17 instances and got optimal solutions for all of them. Furthermore, Pacino and Jensen [22] used a local search extended heuristic to solve the same problem, and then the authors extended their work to include an

integer programming model. They increased the number of instances to 133 to test the solution approach. The results show that about 86% of the total instances were solved to optimality [23], [16].

The local search extended heuristic got the solution faster than the constraint programming in larger instances. Delgado et al, provided an integer programming model and extensive experiments on 236 instances [6]. A constraint programming model was able to solve 90% of the instances in less than one second, and it was faster than the integer programming model.

Parreño et al., extended the problem to include the IMO containers and introduced an additional objective which is the minimization of the rolled out containers (i.e. containers that were removed from the list of the containers to load) [2]. The authors developed a Greedy Randomized Adaptive Search Procedure (GRASP) algorithm and presented a novel integer programming formulation. Furthermore, they provided a comparison between different strategies for the GRASP algorithm and compared their work against the constraint programming and constraint-based local search approaches.

In conclusion, the number of articles discussing the slot planning phase is limited and the proposed solution approaches to solve this problem are rather small. The below deck locations have predefined objectives and constraints, decided by the shipping line. The problem is NP-hard, and it is a crucial sub-problem of the stowage planning problem. The time saved in reaching the optimal or near-optimal solutions of the slot planning problem aid the stowage coordinators in developing an effective complete stowage plan. Consequently, an effective stowage plan reduces the time spent at the port, increases the efficiency and profit for both the shipping line and the port.

In this work, a solution approach with multiple stages is presented and tested. In this approach, each stage uses a rule-based heuristic structure to eliminate/reduce a constraint violation and/or improve the objective function. This approach mimics the planning process of the stowage coordinator and gives the flexibility of adding new considerations by adding another stage in the solution approach.

4. Solution Approach

To tackle the problem of stowing 40’ containers within a location, an approach with multiple stages was proposed that imitates the planning process of the stowage coordinator, each stage has the objective either to minimize/eliminate constraint violation or improve the objective function by following guided rule-based structures. The sets, parameters, and variables used in our approach are introduced in Table 2. An overview of the proposed approach is presented in Fig. 3, then the details of the approach are presented below.

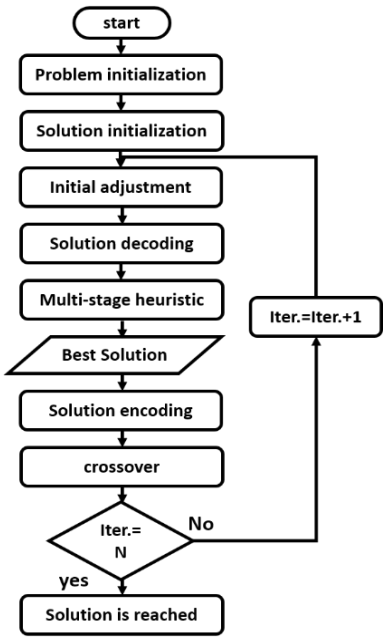

Fig. 3: An overview of the proposed solution approach.

4.1. Problem Initialization

In this part all the inputs of the problem are declared, some of which are: the characteristics of the stacks (i.e., height and weight limits), tiers and cells characteristics (i.e., reefer/not, feasible to stow a 40' container /not). Furthermore, the characteristics of the containers to be loaded and the already loaded if exists (i.e., reefer/not, height, weight), and the number of the discharge ports in the problem.

Table 2: The sets and variables used in the proposed solution approach.

C	Set of containers to be loaded, where $c \in \{1,2,\dots,C\}$
S	Set of stacks, where $s \in \{1,2,\dots,S\}$
T	Set of tires, where $t \in \{1,2,\dots,T\}$
X	A complete solution
C_s	Set of containers assigned to stack s
R_s	Set of reefer containers assigned to stack s
EC_s	Set of feasible empty cells in stack s
RC_s	Set of the remaining number of reefer cells in stack s
$devC(s)$	Difference between the number of assigned containers to stack (s) and the feasible empty cells in the same stack, where $s = 1\dots S$ & $devC(s) = C_s - EC_s$
$devH(s)$	Difference between the sum of the total heights of the containers in a stack (s) and its height limit, $devH(s) = \mathrm{H_s} - \mathrm{HL_s}$
$devW(s)$	Difference between the sum of the total weights of the containers in a stack (s) and its weight limit, $devW(s) = W_s - \mathrm{WL_s}$
$devR(s)$	Difference between the remaining number of reefer cells and the assigned reefer containers in stack s, $devR(s) = RC_s - R_s$

4.2. Solution Initialization

In this part, the initial set of solutions is randomly generated. The set size is 20 solutions and each solution X as shown in Fig. 4 has a number of elements that is equal to the total number of containers to be loaded (C). Whereas the location of the element in solution X indicates the container's index $(c), c = 1,2,..,C$. Each element in the solution is a random positive real number (x), where $\{x \in R^{+} | 1 \leq x < S + 1\}$ and (S) is the number of stacks in the problem.

This number is a float number within which the integer part specifies the stack number that the container is assigned to, while the fractional part controls the order of the container within the stack (i.e., all the containers assigned to a specific stack are ordered from the bottom up in descending order of their fractional number).

X= | x^1 | x^2 | ….. | x^c | …... | x^C |

Fig. 4: A solution representation.

An example will be used to illustrate the explanation of all parts of our algorithm and the related information is as follows: the problem has seven containers to be loaded, and the location has 3 stacks and 4 tiers. The details of the characteristics of both the container to be loaded and the location are provided in Fig. 5 showing the blocked cells and the reefer ones and the height and weight limits of each stack.

C	H	W	P	R
1	2.59	1	2	1
2	2.9	1.5	3	1
3	2.9	1.5	3	1
4	2.9	2	3	1
5	2.9	1.5	2	1
6	2.59	1	3	0
7	2.59	1	2	1

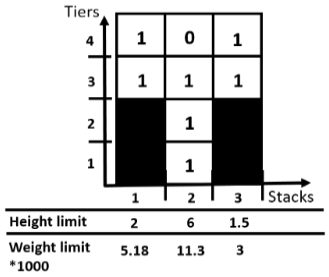

Fig. 5: The information of the example of the containers to be loaded at the left including (container's number, height, the wight, discharge port and reefer characteristics (1 (reefer), 0 (non-reefer)), the location characteristics on the right showing the blocked cells and the reefer ones denoted by (ones) and the height and weight limits of each stack.

The starting solution X can be as shown in Fig. 6. As noted, the containers {1, 3, 6} are assigned to the 1st stack, containers {2, 5} are assigned to the 2nd stack, and finally, containers {4, 7} are assigned to the 3rd stack. The fractional part of each number specifies the tier that the container will occupy.

X_{before}=

1.32	2.45	1.76	3.76	2.67	1.22	3.56

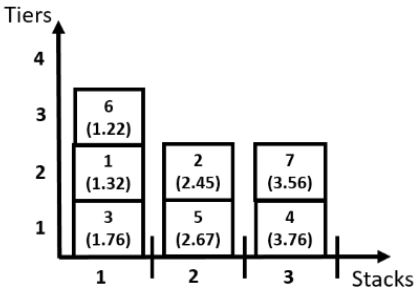

Fig. 6: A randomly generated solution X for seven containers and its interpretation.

4.3. Initial Adjustment

In this part, the starting solution X is adjusted for feasibility in terms of the capacity of available cells for each stack s i.e., $C_s \leq EC_s$. The mechanism of this subroutine is presented in Fig. 7. Initially $(devC)$ is calculated for all stacks, the positive values belong to a violating stack. Subsequently, assign the stack that has the Maximum violation and the stack with the minimum deviation from the available cells. $Max(devC) = Max(devC(s))$ and $Min(devC) = Min(devC(s))$, where $s = \{1, \ldots, S\}$. To remove the violation, a number of containers are moved from the most violating stack to the stack with the least deviation value by changing the integer part of the random number assigned to the containers.

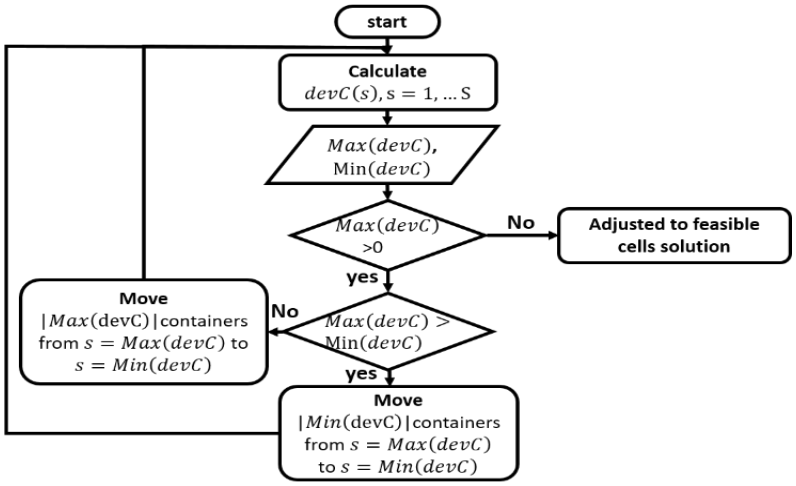

Fig. 7: The overview of initial adjustment.

For the same example presented before, stack one and stack three have two cells available and stack two has four cells available. As shown in Fig. 8 the first stack was the most violating and the second stack has the least deviation value. This resulted in moving container one (i.e., the first container assigned to stack one) to stack two and changing its random number. After the initial adjustment, the solution is decoded to start the multi-stage heuristic, the solution representation of After is transformed from a series of random numbers to locations as shown in the interpretation in Fig. 8.

4.4. The Multi-Stage Heuristic

In this subroutine, a sequence of procedures is performed to ensure a set of objectives: The feasibility of the solution regarding each stack's limits of both the height and the weight, the solution doesn't have any reefer container assigned to a non-reefer cell, and finally to minimize the value of the objective function. Consequently, this is achieved using rule-based search structures on the solution X_{after}, followed by calculating the fitness function (i.e., total penalty) of the solution. An overview of the multi-stage heuristic is presented in the flowchart presented in Fig. 9.

A brief description of each procedure is given and illustrated using the same example as follows:

X_{after}=

2.32	2.45	1.76	3.76	2.67	1.22	3.56

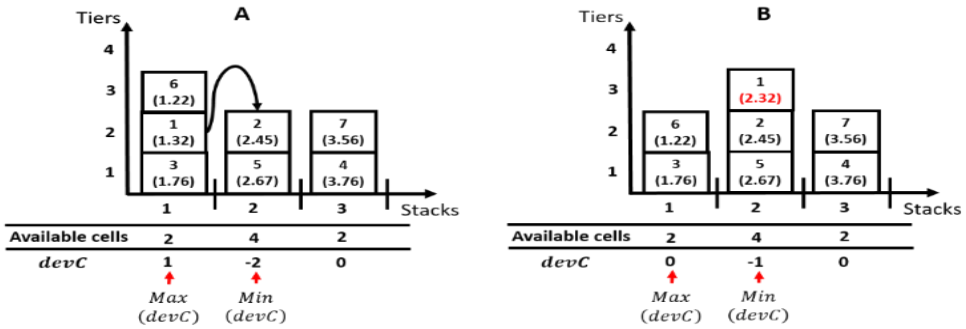

Fig. 8: (a) The interpretation of X_{before} and (b) the interpretation of X_{after} after the initial adjustment.

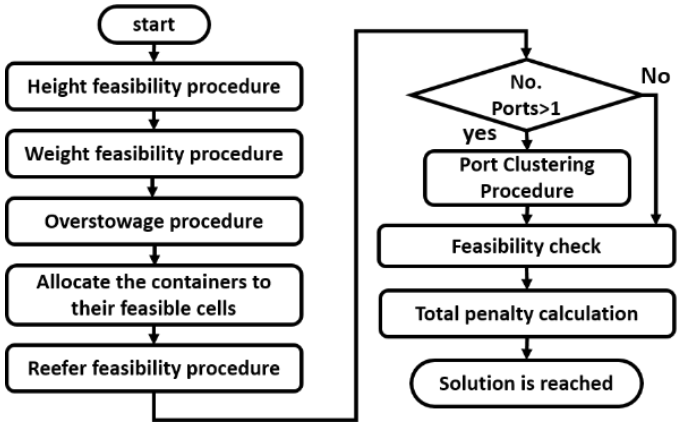

Fig. 9: The flow chart of the multi-stage heuristic.

a) Height feasibility procedure

This is performed using a guided rule-based search structure with the objective of minimizing the violation of the height feasibility constraint of the stacks and to cluster the containers within fewer number of stacks. The main idea of this procedure is shown in Fig. 10 and the Containers clustering sub-routine is illustrated in Fig. 11. Additionally, Fig. 12 follows the previous example for the illustration of this procedure. (HL_s) is the height limit of stack s and (H_s) is the sum of the heights of the assigned containers to stack s.

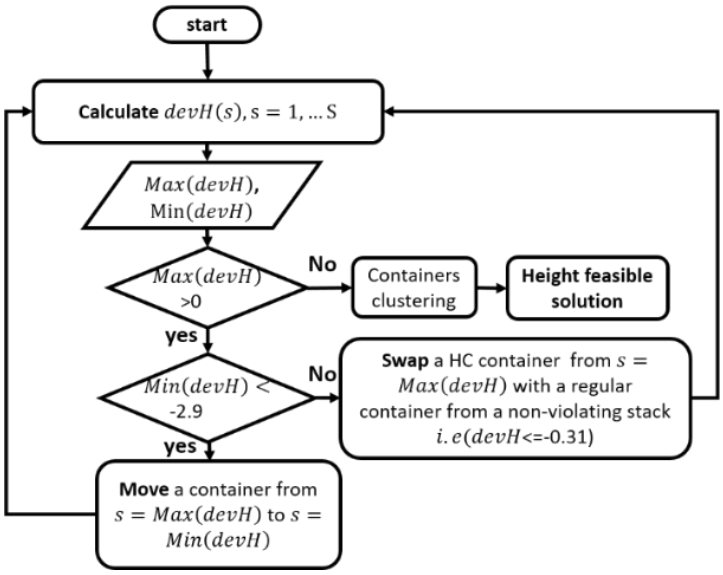

Fig. 10: The main idea of the height feasibility procedure.

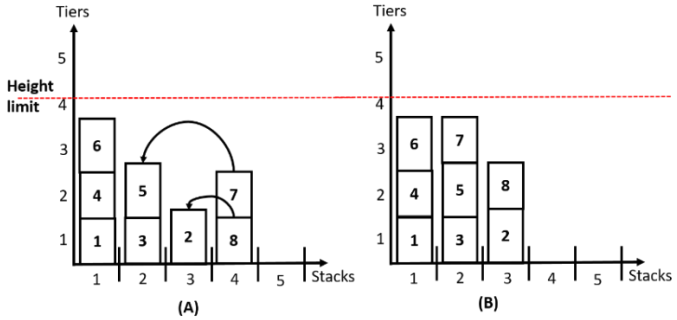

Fig. 11: Containers clustering sub-routine in the height feasibility procedure.

Stack one and stack three were in violation, initially, container number seven is moved to the second stack with its updated random number as shown in Fig. 12 (a & b). Additionally, a swap between container number three and container number one is performed in Fig. 12 (b & c). This resulted in a height feasible container assignment.

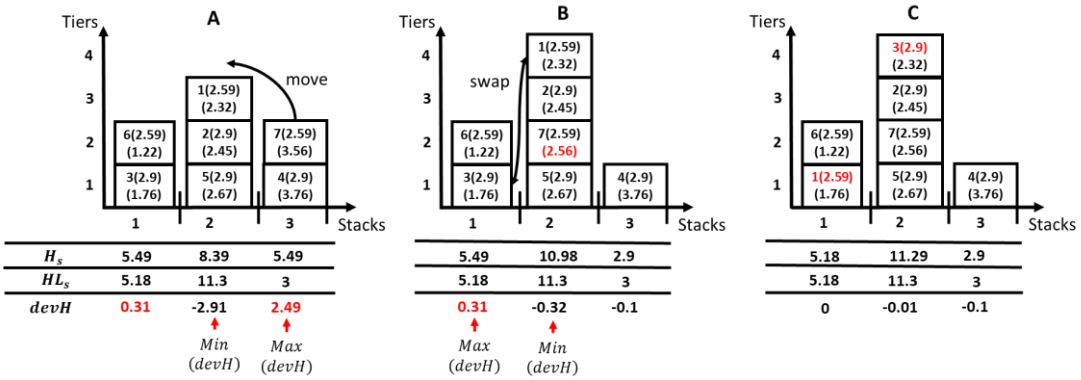

Fig. 12: (a) The decoded X_{after} with containers' heights and (c) is the height feasible solution showing the steps of getting it.

b) Weight feasibility procedure

A rule-based search structure is used with the objective of eliminating the violation of the weight feasibility constraint of the stacks. Moving or swapping the containers to remove the weight constraint violation should be performed in a manner that will not violate the previous procedure (i.e., height feasibility). The main idea of this procedure is shown in Fig. 13.

Fig. 14 follows the example for the illustration of this procedure, (WL_s) is the weight limit of stack s and (W_s) is the sum of the weights of the assigned containers to stack s. Stack three was in violation, the swap between container number three and container number four is performed (i.e., they are of the same height) this resulted in a weight and height feasible container assignment in Fig. 14 (b).

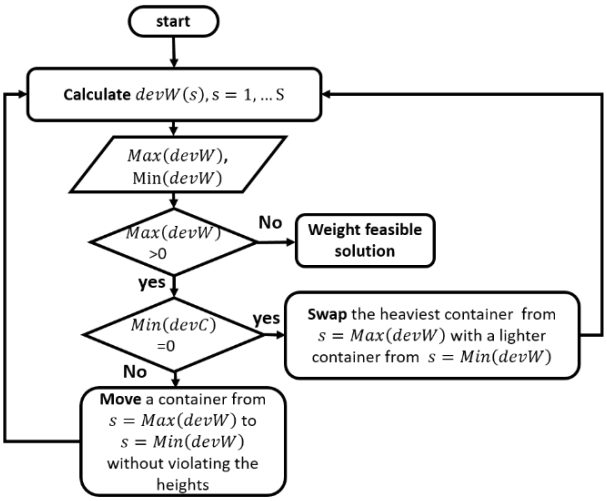

Fig. 13: The main idea of the weight feasibility procedure.

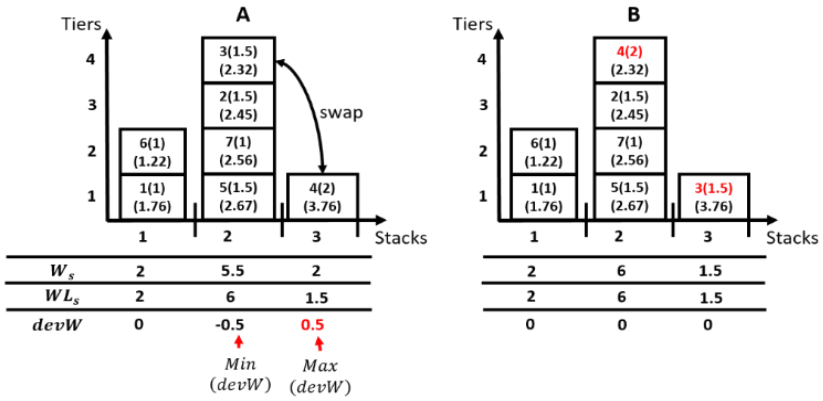

Fig. 14: (a) The height feasible solution with containers' weights and (b) is the weight feasible solution showing the steps of getting it.

c) Overstowage procedure

In this part, the heuristic seeks to minimize the overstowage objective. The containers are arranged in descending order of their destination ports from the bottom of the stack as illustrated with the example shown in Fig. 15 (b). however, from the problem description, the location can have blocked cells and to this point of the heuristic the exact position of the containers inside the location was not needed and the containers were assigned to their stacks from the bottom up and they were not positioned in their actual cells.

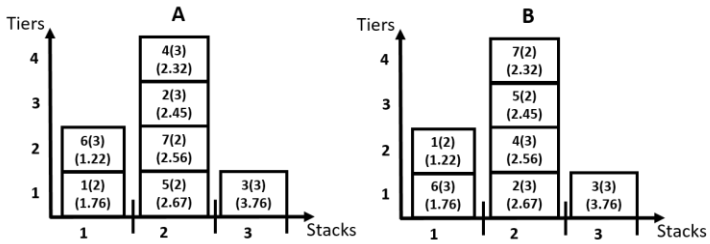

Fig. 15: (a) The solution with containers' destinations, and (b) the solution after rearranging the overstowage procedure.

To proceed with the heuristic and do a comparison between the reefer characteristic of the container and the cell it occupies; containers must move to their feasible cells. If there are any containers already existing, they will be added to the location as well. Fig. 16 illustrates that in the given example.

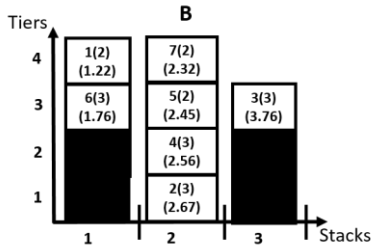

Fig. 16: The assignment of the containers to be loaded to their feasible cells.

d) Reefer feasibility procedure

Then an attempt to reduce/eliminate the reefer infeasibility where there is a reefer container occupying a non-reefer cell (i.e., violating container). A rule-based search structure was adopted to swap/move the violating containers to reefer cells. It seeks a better solution by swapping containers to reduce and/or eliminate the height infeasibility as done in the height feasibility procedure but only swapping containers with the same reefer characteristics.

The details are illustrated in Fig. 17 it is important to note that if $devR(s)$ has a negative value this indicates that the reefer containers assigned to this stack (s) exceed the reefer capacity of this stack (i.e., a reefer violating stack). Moreover, if $devR \geq 0$ then this indicates that the reefer containers assigned to this stack (s) are equal/less than the reefer capacity of this stack (i.e., a reefer non-violating stack) Even though the stack may be a non-reefer violating stack this doesn't ensure that it doesn't have violating-containers (i.e., a reefer container in a reefer cell), therefore swaps must be performed within this stack.

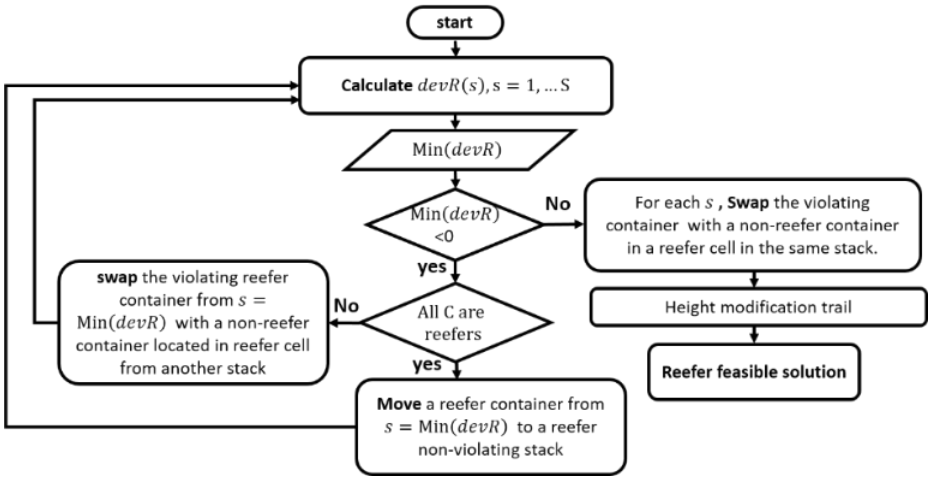

Fig. 17: The main idea of the "reefer feasibility procedure".

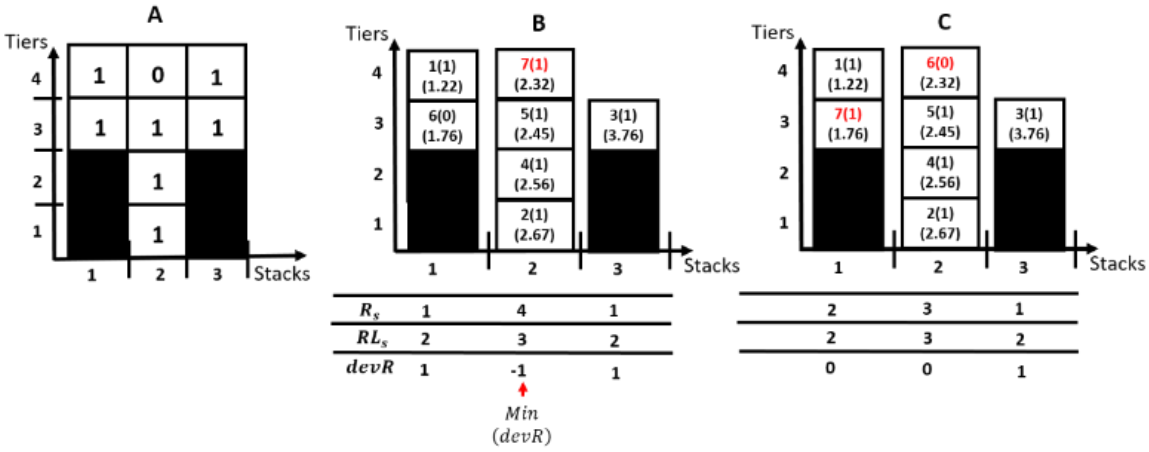

Fig. 18: (a) Location reefer characteristics, (b) and (c) are the assignment of the containers before and after applying the reefer feasibility algorithm respectively.

In the same example, only container number six is non-reefer which will be used to illustrate the procedure. Fig. 18 (a) is the reefer characteristics of the location, the number "1" means that the cell is reefer. Fig. 18 (c) is the solution after applying this procedure where container number seven in stack two was swapped with container number six in stack two.

e) Port clustering procedure

If there is more than one port of destination in the problem, this procedure is used to minimize the port clustering objective. It moves sequentially over the stacks swapping containers between different stacks. This is to cluster the containers with the same port of destination within the same stack, each swapping container must be of the same height and reefer characteristics. In the given example no port clustering can be performed. Fig. 19 can be an illustrative example of the simple overall objective of this procedure, and its effect on decreasing the fitness function by decreasing the port clustering penalty.

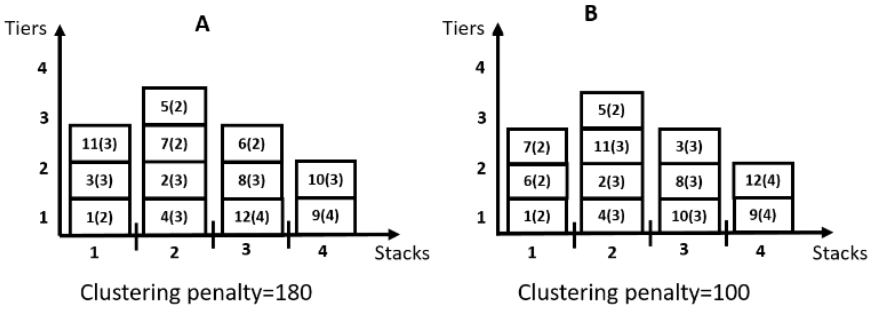

Fig. 19: The effect of the port clustering procedure on decreasing the port clustering penalty of a feasible solution.

f) Total fitness function calculation

In this example, the fitness function (i.e., objective function) =100*(1) +20*(1+2+1) +10*(1+1+1) +5*(0) =210. It means 100*(overstowing containers) +20*(number of discharge ports in each stack) +10*(stacks used) +5*(number of non-reefer containers in a reefer cell). In this example container number six overstows container number 5.

g) Height-weight-reefer feasibility check

The algorithm aspires to eliminate the infeasibility at each step, but it can't be ensured. So, a feasibility check must be performed. Consequently, if the solution is feasible then it is encoded back to its initial form. In the example, the solution X is feasible, then its encoded form is shown in Fig. 20.

X=	1.22	2.67	3.76	2.56	2.45	2.32	1.76

Fig. 20: The solution X after applying the multi-stage heuristic.

4.5. Iterations

The first set of 20 different solutions have different objective functions (i.e., penalty cost), these solutions are sorted based on their objective function value. The minimum value (i.e., best solution) is recorded as the initial solution X. Subsequently, the idea of using a single point crossover is adopted from the famous genetic algorithm heuristic to introduce the intensification process to our solution heuristic. Consequently, a single point crossover between the solution (i) and solution $(i+1)$, where $i = 1{:}\,19$, in which the cross-over point is randomly generated. After 10 iterations the final solution is reached.

5. Computational results and analysis

The proposed approach was coded using MATLAB R2017a ® and the instances were run on an Intel (R) Core (TM) i7 CPU @ 3.6 GHz, and 8.00 GB RAM computer. There are 127 slot planning benchmark instances of 40' containers that are found in the literature [2]. The instances are partitioned into five different groups where each one is a combination of instances that have similar features as shown in Table 3.

The first column represents the group ID, the second column indicates the number of instances in each group, then the (maximum-minimum-average) number of containers in (TEUs) in each group is presented and followed by the (maximum-minimum-average) available capacity of TEUs in each group. The following columns indicate if the group has a specific feature; whether the group has containers that are High Cube

(HC) and/or Reefer (R). The last three columns indicate the number of instances that have containers belonging to one, two, or more destination ports in each group. Furthermore, almost 79% of the solved instances have a container utilization (i.e., an indication of how full the location is with containers) of more than 70% utilization, which is the case in real-life instances.

The model was first run to reach the initial solution without any iterations (i.e., crossover) each instance is run five times and the fitness function of each of the five runs and the average time is reported, and the proposed solution is then allowed to be stopped after one, five and ten iterations each is run for five times to test the effect of the iterations on the solution improvement.

Table 3: Instances groups and characteristics.

Grp	#inst.	Cont. (TEU)			Cap. (TEU)			HC	R	#Ports		
		Min.	Max.	Avg.	Min.	Max.	Avg.			1	2	≥3
1	13	8	116	54	16	116	63			13		
2	15	16	74	46	42	172	73	√	√	15		
3	78	2	202	63	6	208	79	√		78		
4	13	40	136	87	40	148	96	√			13	
5	8	10	156	93	72	176	122	√	√		6	2

The results for the different number of iterations are reported in Table 4, in the table, (IS) stands for the number of instances solved in each group, (opt) indicates the number of instances that are solved to optimality. The bolded red number indicates that there are non-optimal solutions reached within this group. (%R) the ratio of the five runs that were able to obtain the optimum solution amongst all the runs performed in that group. (T(s)) is the average time in second for all the runs of all the instances in the group.

The results indicate that our algorithm is effective and efficient in reaching the optimal solutions of the benchmark instances. The quality of the initial solution is of a high standard as it was able to reach the optimum solution for 125 instances of the benchmark instances in 95% of the total runs, the two non-optimal solutions were deviated by 4.2% and 14.3% from the optimum solution. Moreover, the average time to obtain the solution was 0.32 seconds which is satisfactory for the one-second limitation.

Table 4: The instances results.

Grp	Initial solution				Solution after 1 iteration				Solution after 5 iterations				Solution after 10 iterations			
	I S	opt	%R	T(s)	I S	opt	%R	T(s)	IS	opt	%R	T(s)	I S	opt	%R	T(s)
1	13	13	1	.28	13	13	1	.33	13	13	1	.42	13	13	1	.52
2	15	15	1	.31	15	15	1	.37	15	15	1	.47	15	15	1	.58
3	78	**76**	.95	.29	78	**77**	.96	.34	78	**77**	.97	.44	78	**77**	.98	.54
4	13	13	.92	.44	13	13	.95	.51	13	13	1	.71	13	13	1	.92
5	8	8	.85	.46	8	8	.88	.55	8	8	.98	.75	8	8	1	.91

After ten iterations the solutions and the consistency of reaching the optimum solutions are improved, all instances but one was solved to optimality in 99% of the total runs, the non-optimal solution was deviated by 2.8% from the optimal solution, additionally, the average time of getting the solutions was 0.61 second which is less than the time constraint.

The performance of our algorithm is compared against the recently proposed solution approaches to solve the problem of slot planning of below deck locations; namely the GRASP proposed by Parreño et al. [2], Constraint programming proposed by Delgado et al. [6], and Constraint-based local search proposed by Pacino and Jensen [16]. The CP in [6] is run within a time limit of 1 second and 10 seconds, furthermore, only the results of the 10 seconds are reported as they have better performance in terms of the percentage of the obtained optimal solutions. Additionally, the number of instances that couldn't obtain a feasible solution within the time limit is reduced. The comparison is held on the instances that have only the 40' containers (Instances 1,6,4,8 and10 respectively).

The comparison is shown in Fig. 21 and Table 5. The first column indicates the group ID and. consecutively, for each solution method, column %Sol indicates the percentage of problems solved, column %Opt presents the percentage of problems solved optimally, and column T(s) shows the total running time needed to solve all the instances in the group.

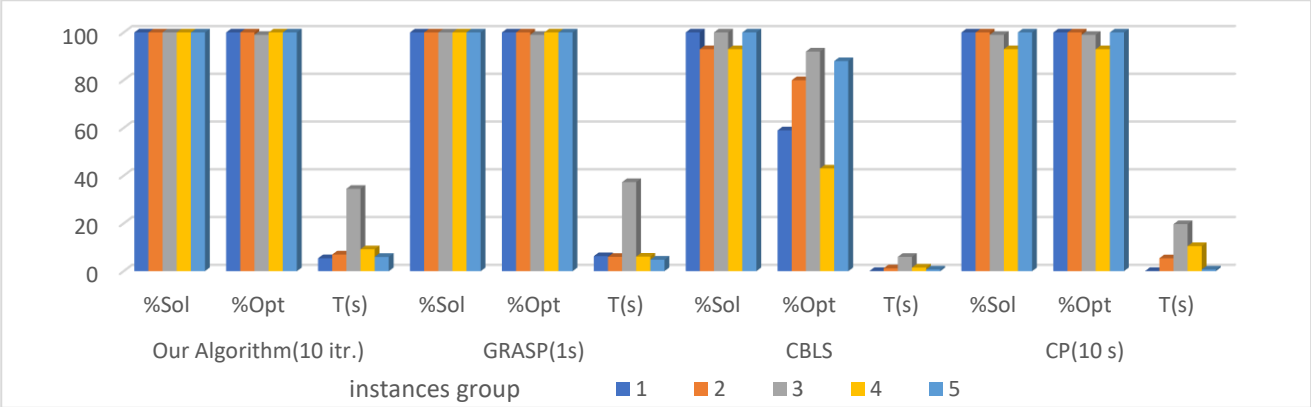

Fig. 21: Comparing the results against GRSAP, CBLS and constraint programming algorithms.

Table 5: Comparing with the GRSAP and constraint programming algorithm and the CBLS heuristic algorithm.

Grp	Our algorithm (10 itr.)			GRASP (1s)			CBLS			CP (10 s)		
	%Sol	% Opt	T(s)	%Sol	% Opt	T(s)	%Sol	% Opt	T(s)	%Sol	% Opt	T(s)
1	100	100	6.19	100	100	6.3	100	59	0.1	100	100	0.1
2	100	100	8.07	100	100	6	93	80	1.2	100	100	5.4
3	100	99	39.4	100	99	37.3	100	92	6	99	99	19.7
4	100	100	7.06	100	100	6.1	93	43	1.5	93	93	10.5
5	100	100	4.21	100	100	4.8	100	88	0.7	100	100	0.7

From the results, the performance of our algorithm in terms of the quality of the solution in terms of reaching the optimal solutions is equivalent to the GRASP, additionally, the run times are almost the same. Additionally, the quality of the solutions of both the CBLS and the CP (10s) are somewhat worse as they fail to find a feasible solution in 2.8% and 1.6% respectively. Furthermore, both the CBLS and the CP (10s) fail to find optimal solutions in 27.6% and 1.6% respectively. However, both the CBLS and the CP are much faster, but this can be attributed to the fact that these approaches used parallel processing. Consequently, the CBLS is considered the fastest approach but fails to reach the optimum solutions in more cases, this can be attributed to getting stuck in a local optimum point.

6. Conclusion

In this paper, the slot planning problem of the 40’ containers for the below-deck locations was solved. A multiple-stage-based approach was developed where each stage is based on a guided rule-based structure. Moreover, it has an objective either to minimize/eliminate constraint violation or improve the objective function. The computational results show that our approach is fast and can produce optimal solutions for all but one instance of the 127 industrial instances that were solved in less than 1 second, which is well within the time constraint for practical usage. The performance of our algorithm showed its efficiency when compared with the state-of-the-art algorithms.

After being used to solve 40’ feet container problems, the procedure can be further generalized to include the 20’ as well as the IMOs containers. Further, future research includes also extending the CSPBDL to include on deck locations and special containers such as out-of-gauge and pallet-wide.

7. References

[1] D. Steenken, S. Voß, and R. Stahlbock, “Container terminal operation and operations research - A classification and literature review,” in *Container Terminals and Automated Transport Systems: Logistics Control Issues and Quantitative Decision Support*, Springer Berlin Heidelberg, 2005, pp. 3–49.

[2] F. Parreño, D. Pacino, and R. Alvarez-Valdes, “A GRASP algorithm for the container stowage slot planning problem,” *Transp. Res. Part E Logist. Transp. Rev.*, vol. 94, pp. 141–157, 2016, doi: 10.1016/j.tre.2016.07.011.

[3] Y. H. Venus Lun, K. Lai, and T. C. E. Cheng, “Investigation of the influences of ‘transport complex economy’ and political risk on freight transport growth,” *Int. J. Logist. Res. Appl.*, vol. 14, no. 5, pp. 285–296, Oct. 2011, doi: 10.1080/13675567.2011.635641.

[4] “Maritime shipping and international trade (web page),” 2021.

https://porteconomicsmanagement.org/pemp/contents/part1/maritime-shipping-and-international-trade/world-container-throughput/ (accessed Jan. 20, 2021).

[5] M. S. Bilican, R. Evren, and M. Karatas, "A Mathematical Model and Two-Stage Heuristic for the Container Stowage Planning Problem with Stability Parameters," *IEEE Access*, vol. 8, pp. 113392–113413, 2020, doi: 10.1109/ACCESS.2020.3003557.

[6] A. Delgado, R. M. Jensen, K. Janstrup, T. H. Rose, and K. H. Andersen, "A Constraint Programming model for fast optimal stowage of container vessel bays," *Eur. J. Oper. Res.*, vol. 220, no. 1, pp. 251–261, 2012, doi: 10.1016/j.ejor.2012.01.028.

[7] W. Shi and K. X. Li, "Themes and tools of maritime transport research during 2000-2014," *Marit. Policy Manag.*, vol. 44, no. 2, pp. 151–169, Feb. 2017, doi: 10.1080/03088839.2016.1274833.

[8] C. S. Parthibaraj, P. L. K. Palaniappan, and A. Gunasekaran, "Multi-agent system with iterative auction mechanism for master bay plan problem in marine logistics," *Marit. Policy Manag.*, vol. 44, no. 6, pp. 705–726, 2017, doi: 10.1080/03088839.2017.1324219.

[9] K. I. Cohen M.W., Coelho V.N., Dahan A., "Container Vessel Stowage Planning System Using Genetic Algorithm," in *Lecture Notes in Computer Science*, vol. 10199, Springer, Cham, 2017, pp. 557–572.

[10] 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 *In International Conference on Computational Logistics*, 2011, vol. 6971 LNCS, pp. 286–301, doi: 10.1007/978-3-642-24264-9_22.

[11] I. D. Wilson and P. A. Roach, "Container stowage planning: A methodology for generating computerised solutions," *J. Oper. Res. Soc.*, vol. 51, no. 11, pp. 1248–1255, 2000, doi: 10.1057/palgrave.jors.2601022.

[12] D. Ambrosino and A. Sciomachen, "Using a Bin Packing Approach for Stowing Hazardous Containers into Containerships," in *Optimized Packings with Applications*, Fasano G.,., vol. 105, G. Fasano and J. D. Pintér, Eds. Cham: Springer International Publishing, 2015, pp. 1–18.

[13] D. Ambrosino, D. Anghinolfi, M. Paolucci, and A. Sciomachen, "A new three-step heuristic for the Master Bay Plan Problem," *Marit. Econ. Logist.*, vol. 11, no. 1, pp. 98–120, 2009, doi: 10.1057/mel.2008.19.

[14] 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 International Symposium on Experimental Algorithms*, 2010, pp. 314–325, doi: 10.1007/978-3-642-13193-6_27.

[15] J. G. Kang and Y. D. Kim, "Stowage planning in maritime container transportation," *J. Oper. Res. Soc.*, vol. 53, no. 4, pp. 415–426, 2002, doi: 10.1057/palgrave.jors.2601322.

[16] D. Pacino and R. M. Jensen, "Fast Slot Planning Using Constraint-Based Local Search," in *IAENG Transactions on Engineering Technologies: Special Issue of the International MultiConference of Engineers and Computer Scientists 2012*, vol. 186, G.-C. Yang, S.-I. Ao, X. Huang, and O. Castillo, Eds. Dordrecht: Springer Netherlands, 2013, pp. 49–63.

[17] M. Gheith, A. B. Eltawil, and N. A. Harraz, "Solving the container pre-marshalling problem using variable length genetic algorithms," *Eng. Optim.*, vol. 48, no. 4, pp. 687–705, Apr. 2016, doi: 10.1080/0305215X.2015.1031661.

[18] M. S. Gheith, A. B. El-Tawil, and N. A. Harraz, "A proposed heuristic for solving the container pre-marshalling problem," *19th Int. Conf. Ind. Eng. Eng. Manag.*, no. January, pp. 955–964, 2013, doi: 10.1007/978-3-642-37270-4_91.

[19] D. Ding and M. C. Chou, "Stowage planning for container ships: A heuristic algorithm to reduce the number of shifts," *Eur. J. Oper. Res.*, vol. 246, no. 1, pp. 242–249, 2015, doi: 10.1016/j.ejor.2015.03.044.

[20] D. M. Rahsed, M. S. Gheith, A. B. Eltawil, D. M. Rashed, M. S. Gheith, and A. B. Eltawil, "A Rule-based Greedy Algorithm to Solve Stowage Planning Problem," in *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Dec. 2018, no. December, pp. 437–441, doi: 10.1109/IEEM.2018.8607517.

[21] A. Delgado, R. M. Jensen, and C. Schulte, "Generating optimal stowage plans for container vessel bays," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, vol. 5732 LNCS, pp. 6–20, doi: 10.1007/978-3-642-04244-7_4.

[22] D. Pacino and R. Jensen, "A Local Search Extended Heuristic for Stowing Under Deck Locations of Container Vessels," *Odysseus 2009 Fourth Int. Work. Freight Transp. Logist. Çeme, Turkey*, pp. 1–3, 2009.

[23] D. Pacino and R. M. Jensen, "Constraint-Based Local Search for Container Stowage Slot Planning," *Lect. Notes Eng. Comput. Sci.*, vol. 2, pp. 1467–1472, 2012.