Scientia Iranica E (2017) 24(1), 309–318

Sharif University of Technology
Scientia Iranica
Transactions E: Industrial Engineering
www.scientiairanica.com

Decreasing the crane working time in retrieving the containers from a bay

E. Azari, H. Eskandari* and A. Nourmohammadi

Faculty of Industrial Engineering, Tarbiat Modares University, Tehran, P.O. Box 14117, Iran.

Received 1 August 2014; received in revised form 2 September 2015; accepted 20 February 2016

KEYWORDS
Container terminal;
Container retrieval problem;
Heuristic;
Branch and bound.

Abstract. Given the initial layout of a container terminal with a bay, the container retrieval problem aims at obtaining a movement sequence for the crane to retrieve all the containers arranged in a pre-defined order. In this study, we develop a computationally efficient heuristic, called constant summation (CSUM), to minimize the total working time of crane, required to retrieve all the containers from a given bay. Numerical results show that CSUM is very promising in dealing with the container retrieval problems and outperforms the best known approaches recently presented in the literature in terms of the crane's working time.

1. Introduction

In an attempt to utilize the available spaces, many yards/bays stack up containers and retrieve them using either one [1] or more [2] cranes. In any container terminal, such as a yard/bay, items have to be retrieved in a sequence, which is influenced by various constraints, e.g. space, movement, or safety limitations. Thus, the Container Retrieval Problem (CRP) arises in yards where the containers are usually stacked up and some of them are buried under the others, which makes their retrieval time-consuming and costly due to the need for transferring of the blocking containers before being able to access them.

Usually, the yard/bay includes several blocks. Those blocks adjacent to the sea are usually assigned to the containers that are to be exported in the near future, which are called the export containers. Regarding the high costs associated with loading up the ships with containers as well as the constraints resulted from the limited berthing locations, it is reasonable to optimize the retrieval of the export containers by loading them onto the ship as short as possible. To do so, a common approach is to pre-marshal the export containers within the blocks nearby the sea according to the stowage plan so that no additional relocation of containers will be required while loading them onto the ships. However, there are many uncertain phenomena, which may lead to undesirable changes in the stowage plan. In addition, there might be limited time available to perform the pre-marshaling operations. Therefore, it is necessary to develop a fast heuristic to optimize the loading of a block of prioritized containers onto a ship in terms of time. A similar problem occurs at the landside where the corresponding import containers have to be unloaded onto the trucks.

To decrease the time required to load the containers onto a ship, most of the studies in the literature have attempted to decrease the number of necessary relocations [3-5]. Experimental results indicate that there is a high correlation between working time of the crane, denoted by T_{CW} , and the number of relocations, denoted by N_{MOV} [6]. T_{CW} is more beneficial than N_{MOV}, because T_{CW} is indeed the main objective function of the CRP; thus, we attempt to minimize ob-

*. *Corresponding author.*
E-mail addresses: Es_azari@yahoo.com (E. Azari); heskandari2008@yahoo.com (H. Eskandari); nourmohammadi.amir@gmail.com (A. Nourmohammadi)

jective N_{MOV} with the hope of obtaining the minimum possible T_{CW}. Therefore, in this study, we attempt to develop a heuristic approach that minimizes T_{CW}.

When N_{MOV} is considered to be minimized, the problem is called the Containers/Blocks Relocation Problem (CRP or BRP) [4,5,7]. Kim and Hong [3] developed a Branch and Bound (B&B) algorithm that made use of the number of blocking containers in the initial layout as a lower bound for the number of required relocations. Since their proposed B&B algorithm was only applicable to small-sized problem instances, they also developed a greedy heuristic based on the expected values of the future relocations. Caserta et al. [8] developed an alternative stacking area representation based on a binary matrix. Since in their approach immediate access to the information about stacked items was allowed, it provided a superior computational time.

Later Caserta et al. [9] applied the corridor method, which was proposed in [10], to solve the BRP by shortening the search tree dimensions. Their computational results on medium and large-sized problem instances verified the effectiveness of their proposed method. Recently, Forster and Bortfeldt [5] defined six categories of movements based on the natural classification of possible moves and then developed a tree search heuristic, accordingly. Their computational results proved that this heuristic was highly competitive in comparison with the best known approaches in the literature. Subsequently, Zhu et al. [4] employed the iterative deepening A* algorithm, which was based on novel lower-bound measures as well as new heuristics. Their IDA* heuristic was able to find the optimal or near-optimal solutions for the considered problems in a time-efficient computational effort.

In a recent paper, Jovanovic and Voß [11] proposed a new heuristic approach that considered the properties of the upcoming movement while relocating the destination of a block. Their approach outperformed several approaches available in the literature. Caserta et al. [7] proposed two different integer programming mathematical models for the BRP. The first model, called BRP-I, mapped the complete feasible region of the BRP, but generated a large search space. By adding some realistic assumptions, they limited the search space and developed a more usable model, called BRP-II. However, their model was only capable of solving small to relatively medium-scaled instances. In a recent paper, Eskandari and Azari [12] showed that BRP-II model was not correct and it over-satisfied the aforementioned simplifying assumptions. They also indicated that some results reported in [7] were incorrect. In addition, they proposed the corrected BRP-II model, called BRP2c. Finally, in order to speed up solving the BRP2c model, they proposed BRP2ci model by incorporating some new cutting constraints into BRP2c so that the resultant model was able to decrease the computational time by 25 times on average. Recently, Petering and Hussein [13] also proposed a new mixed integer formulation of BRP, called BRP-III. This new formulation, in comparison to the BRP-I [7], had the advantages of fewer decision variables and lower runtime. In a more recent paper, Zehendner et al. [14] also found the incorrectness of BRP-II model proposed by [7]. As in [12], they first corrected the BRP-II model and then improved it by removing superfluous variables, tightening some constraints, introducing a new upper bound, and applying a pre-processing step to fix several variables, which led to better computational efficiency.

Lee and Lee [6] applied a neighborhood search approach to resolve the CRP by considering blocks with more than one bay. Their proposed approach generated some tours in each iteration by using a mathematical model and the shortest one was chosen, subsequently. To the best of our knowledge, their approach entails a high computational time and is the first paper that chose T_{CW} as the objective function of the CRP.

In addition to [6], very few related studies have considered T_{CW} as the objective function [15,16]. Forster and Bortfeldt in [15] used a tree search procedure, which was previously proposed by them in [5], to solve the problems with more than one bay. They showed that their approach was more promising than the one proposed by Lee and Lee [6]. Also, Ünlüyurt and Aydın [16] considered T_{CW} as the objective function, which will be discussed in more detail in Section 5.

Although the retrieval operations and their required time are dependent on other operations such as routing and transferring of the containers by trucks, to keep the problem simple, we assume that there is always a truck available to transfer the containers from the bay towards the berthing location. By this assumption, the problem could be considered independent from other operations that may influence it.

In this paper, we aim to minimize the total working time of crane required to retrieve all the containers from a single bay, which is less considered in the literature. To this purpose, a computationally efficient heuristic algorithm, called constant summation (CSUM), is developed. The main contribution of this paper is the branching strategy proposed in CSUM, which aims at reducing the overall distance that the crane needs to move. The computational results indicate that CSUM is timely efficient and outperforms the best known solution approaches.

The rest of this study is organized as follows. Section 2 describes the CRP and some related definitions and notations are introduced in Section 3. The proposed heuristic algorithms are presented in Section 4 and the computational results are discussed

in Section 5. Finally, the last section includes the concluding remarks.

2. Container retrieval problem

In general, we can divide a container terminal into three major areas: the quayside, the landside, and the container yard [17]. A container yard consists of several blocks, and each block comprises several bays. Each bay has the same number of stacks. Each stack has the same maximum capacity of containers. As in [6], in order to keep the problem simple, we have assumed that all the containers are of the same size. Each container is assigned a number, which represents its priority over others. Lower numbers indicate higher priorities for leaving the bay. Exactly one crane is used to transfer the containers from stacks to the truck. To this purpose, akin to [6], the Rail Mounted Gantry Crane (RMGC) is employed as one of the most widely used types of cranes. This crane transfers the containers one at a time and can access the ones that are located on top of the stacks. As shown in Figure 1, the bay is one-sided. That is, only one side of the bay is allotted to the retrieval of containers, which is shown next to the stack.

We suppose that at the beginning of the process, the RMGC is located along the bay and above the truck lane, next to the stack. For relocating a container from a stack such as s_1 to another one such as s_2, we use the same procedure applied in [6], which is as follows. First, the trolley should be moved above the stack s_1; this operation is called the trolley movement. Then, using the spreader, the container will be picked up; this operation is called the spreader movement. After lifting the container up, it should be transferred to stack s_2 using the abovementioned movements. In this paper, we consider the time associated with these operations to be deterministic. In other words, working time of the crane is proportional to the distance it has moved during its operations. Although there are other realistic issues, which affect crane working time, we chose above a procedure from the literature [6] to make our results comparable with other results available in the literature.

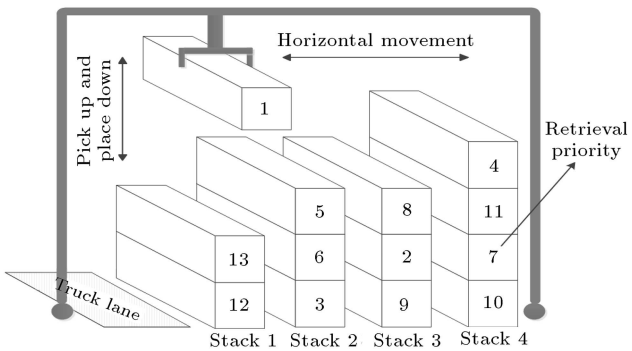

Figure 1. The configuration of a typical bay.

In Figure 1, the configuration of a typical bay is shown. For a given block of containers, a solution to this CRP is a sequence of movements, which satisfies the following assumptions that are common in the CRP literature [5,6]:

- Only those containers that are located on top of the stacks are directly accessible;
- In each stage, only the container with the highest priority can be retrieved;
- Containers could only be relocated to the stacks of the same bay;
- The maximum capacity of the stacks could not be exceeded;
- During the retrieval process, no container could enter into the bay;
- The crane transfers containers one at a time.

In this study, we aim at minimizing the working time of the crane, i.e. T_{CW}, as the main objective function of the problem. In each stage of the retrieval process, the container with the highest priority and the stack that includes it are called the present container and the present stack, respectively. Possible movements are of two types: the relocation and the remove. In relocations, a container is transferred from one stack to another within the same bay. In removes, the present container is removed from the bay or, equivalently, it is relocated to the truck lane. In each stage of the retrieval process, in order to remove the present container, other containers that have blocked it must first be relocated to the other stacks.

3. Definitions and notations

In this section, the definitions and the notations used in developing the proposed heuristic algorithms are presented. The parameters W and H refer to the number of the bay's stacks and the number of tiers in each stack, respectively. The third parameter, i.e. N, refers to the number of the containers present in the initial configuration of the bay. The notations used throughout this paper along with their corresponding definitions are summarized in Table 1.

At any stage of the retrieval process, akin to [5], we classify each container as either a good container or a bad container.

Definition 1. *If a container has blocked a container with a higher priority, then it is called a bad container; otherwise, it is called a good container.*

Furthermore, similar to [5], we propose to classify the relocations according to the placement status of the moved item before and after the relocation operation.

Table 1. The definitions of the notations used in this study.

Notations	Definitions
PS	Present Stack
PC	Present Container
w	The stack number of the PS
q	The container which is located on top of the PS
S	The set of all unfilled stacks of the bay except the present stack
N_m	The current number of movements
LB	The lower bound for the number of movements required to empty the current bay
UB	The sum of the N_m and the LB
t_{cw}	The current working time of the crane
T_{best}	The crane's working time associated to the best found solution
t_{rem}	The time required to remove the PC
t_{rel}	The time required to relocate q to the destination stack

Definition 2. *If a bad container is still a bad container after the relocation, then this relocation is called a Bad-Bad (in short BB) relocation. Analogously, if a bad container becomes a good container after the relocation, then this relocation is called a Bad-Good (BG) relocation.*

The two aforesaid definitions were also applied by Forster and Bortfeldt [5]. For example, in Figure 1, the containers 4 and 7 are good containers and the containers 5 and 8 are bad containers. Relocating container 5 to stack will be a BG relocation, but relocating container 5 into stack 4 will be a BB relocation. In this study, the good containers are not allowed to be relocated. To distinguish the best BB relocation or the best BG relocation for a given container q, the following definition is developed.

Definition 3. *The best BB stack, $s \in S$, for a given container, q, is the stack onto which relocating the container q is a BB relocation and the lowest number in stack s is larger than the lowest numbers in all other stacks of the set S. Analogously, the best BG stack, $s \in S$, for a given container, q, is the stack onto which relocating the container q is a BG relocation and the lowest number in stack s is lower than the lowest number in all other stacks of the set S.*

According to Definition 3, some relocations are called the best because we suppose they will decrease the possible future relocations. For example, in Figure 1, after retrieving container 1, the PC and the PS will be container 2 and stack 3, respectively. For retrieving container 2, $q = 8$ and $S = \{1, 2, 4\}$. The best BB stack and the best BG stack for q in S are stacks 4 and 1, respectively. It is worthy to mention that in Definition 3, the relocation is considered to be the best BG as it can minimize the difference between the relocated container and the lowest number in the destination stack, which can in turn reduce the chance of finding a better relocation to that destination stack in subsequent relocations. An analogous reasoning underlies the definition of the best BB.

In the above definition, if stack s is empty, in order to make the relocation of containers onto it possible, the smallest container number for it should be set equal to $N + 1$. Also, if more than one empty stack satisfies the condition stated in Definition 3, the one with the lowest stack number would be selected.

4. The heuristic approaches

In this section, we first introduce the proposed approach by Kim and Hong [3] for calculating a lower bound on the number of movements required to remove the containers of a bay. Then, we present a heuristic, called the Good-Bad Heuristic (GBH), which attempts to minimize the N_{MOV}. Finally, we propose our main heuristic, called constant summation (CSUM), which is developed based upon the GBH and attempts to minimize the main objective, i.e. T_{CW}.

4.1. Lower bound

Different approaches for calculating a lower bound on the number of movements required to remove the containers of a bay have been proposed in the literature [3-5]. Since the approach proposed by Kim and Hong [3] is simple and efficient, in this study, we have decided to use it, which is as follows. If x and y refer to the numbers of bad and good containers existing in the current bay (according to Definition 1), respectively, then a lower bound on the number of movements can be calculated by $2x + y$.

4.2. The good-bad heuristic

The Good-Bad Heuristic (GBH) is developed to minimize the number of movements, i.e. N_{MOV}. In GBH, if a remove is possible it is done; otherwise, GBH follows at most two new branches, which are the best BG stack and the best BB stack for the uppermost container of the PS, respectively. If no BG relocation is available, the first and the second best BB relocations are followed. In addition, the relocations are limited to originate only from the PS. In this approach, by allowing for the evaluation of at most two branches, i.e. the best BB and BG relocations, which limit the number of branches, as well as by limiting the relocations only to one source stack, it is tried to speed up the GBH heuristic.

```
BEGIN
  UB := H × N;   N_m = 0;   GBH();      N_MOV = UB + 1;   Print the final solution;
END.

GBH (): begin
    if  the time limit is reached then      return;      endif;
    if  PC is not blocked then                          //Retrieval
        if only one container is remained then
           Retrieve it;   N_m++;   UB := N_m − 1;   save the solution;
           Undo the last retrieving; N_m−−;
        else
           Retrieve it;   GBH();   Undo the last retrieving;
        endif;
    else                                                //Relocation
        q := the container which is located on top of the present stack;
        S := the set of all non-full stacks of the bay except the present stack;
        if there is any BG relocation for q then
           FDS := best BG in S for q;      SDS := best BB in S for q;
        else
           FDS := best BB in S for q;      S = S − {FDS};    SDS := best BB in S for q;
        endif;
        for k ∈ {FDS, SDS} do
            if k = ∅ or the time limit is reached then      return;      endif;
            Transfer the container q to the stack k; N_m++;
            if N_m + LB <= UB then   GBH();   endif;
           Go back to the last movement; N_m−−;
        endfor;
    endif;
End;
```

Figure 2. The pseudo code of the GBH algorithm.

Figure 2 represents the pseudo code of the GBH algorithm. Before starting the GBH, we set UB equal to $H \times N$ that is obviously an upper bound on N_{MOV} as well as $N_m = 0$ and then the final solution found by GBH will be reported. Regarding the $\mathcal{NP}$-hardness of the problem [7], the computational time of the GBH may increase exponentially, so we have imposed a time limit on the search process. At the first step, we check whether the time limit is reached or not. If it is reached, the search will be aborted. Otherwise, we check whether the PC is blocked or not. If it is not blocked and it is the only remaining container in the block, we first retrieve (remove) it from the block and then set the UB equal to one unit less than the length of the current solution. We save the current solution as the best achieved solution so far and, finally, in order to search other branches, we go back to the last upper level.

If the PC is neither blocked nor the last container in the bay, after retrieving it, we call for GBH to continue the search process. However, in order to carry on a recursive search, after searching the current branch, we have to return one level back from the current branch that is equivalent to its removal.

If the PC is blocked, in order to releases it, the blocking containers should be transferred to the other stacks. Let q be the uppermost container in PS and S be the set of all the stacks of the bay except PS. In making new branches, at most two alternatives will be examined. If there is a possible BG relocation for q in S, the First Destination Stack (FDS) is the best BG relocation in S for q and the Second Destination Stack (SDS) is the best BB relocation in S for q. If there is no BG relocation for q in S, the first one (FDS) is the best BB relocation in S for q and the second alternative (SDS) is the best BB in $S-\{\text{FDS}\}$ for q. Then, for the stacks FDS and SDS (if available), we do as follows.

If neither FDS nor SDS is available or the time limit defined by the analyst is reached, the search is ceased. Otherwise, the container q will be moved to the destination stack, which is either FSD or SDS. Now, if N_m plus Lower Bound (LB) is less than or equal to UB, we follow this branch by calling for GBH procedure. Otherwise, it is aborted and the last movement will be returned.

4.3. Constant summation heuristic

Pseudo-code of the constant summation (CSUM) heuristic is shown in Figure 3. This heuristic utilizes the GBH in its structure so that the problem is first solved using the GBH and the obtained solution is used as an initial solution to CSUM. According to this initial solution and with respect to the configuration of the bay, we divide all the containers into two different sets, i.e. A and B. If the number of relocations for the container q in the initial solution is more than one, then this container belongs to set A; otherwise, it belongs to set B. Then, according to the initial solution, we

```
BEGIN
    solve the problem using GBH by considering a time limit T and save the initial solution; A := ∅; B := ∅;
    for each container n in the initial solution do
        if there is at least one BB relocation for container n in the initial solution then
            add container n to the set A;
        else
            add container n to the set B;
        endif;
    endfor;
    N_m = 0;  N_MOV := the number of movements in the initial solution;  t_cw = 0;  T_best := ∞;  call for CSUM ();
END.

CSUM (): begin
    if the time limit is reached then  return;  endif;
    if PC is not blocked then                                   //Retrieval
        if only one container is remained then
            remove it from the block;  t_rem :=removal time;  t_cw := t_cw + t_rem;  N_m++;
            if t_cw <= T_best then
                N_MOV := N_m;  T_best := t_cw;  Backup the solution;
            endif;
            t_cw := t_cw − t_rem; go back to the last removal; N_m−−;
        else
            remove the PC;  t_rem := removal time;  t_cw := t_cw + t_rem; N_m++;
            CSUM ();  t_cw := t_cw − t_rem;  Undo the last retrieving;  N_m−−;
        endif;
    else                                                        //Relocations
        q := the container which is located on top of the present stack;  Q := ∅;
        if q ∈ A then
            if there is any BG for q then
                add the best BG for q to the list Q;  add the best BB for q to the list Q;
            else
                add the first and second best BB relocations for q in the list Q in order;
            endif;
        else
            G1 := the set of stacks such that their number is less than the number associated to the present stack;
            G2 := the set of stacks such that their number is greater than the number associated to the present stack;
            s := the best BG relocation for q in G1;
            if s = ∅ then  bound := ∞;
            else bound := the lowest number in the stack s; add s to the list Q; endif;
            for each s ∈ G2 in increasing order do
                if moving q to s is a BG relocation then
                    m :=the lowest container in stack s;
                    if m < bound then add s to the bottom of list Q; m := bound; endif;
                endif;
            endfor;
        endif;
        if the time limit is reached then  return;  endif;
        for every stack s ∈ Q from up to down in order do
            t_rel := the time needed to relocate q to stack s;  t_cw := t_cw + t_rel; N_m++;
            if N_m <= N_MOV then transfer q to stack s; CSUM(); go back to the last relocation;  endif;
            t_cw := t_cw − t_rel; N_m−−;
        endfor;
    endif;
End;
```

Figure 3. The pseudo code of the CSUM heuristic.

assign the initial values to the variables N_m, N_{MOV}, t_{cw}, and T_{best}. Afterwards, we call for CSUM, which is a recursive procedure.

In performing the CSUM, if the PC is not blocked, we do the same as we did in GBH. If it is blocked and q belongs to set A, the same branching strategy as the one used in GBH will be employed. The main difference between GBH and CSUM is when q belongs to the set B. In this case, if the PS number is w, the first destination stack is the best BG stack for q in the set $\{1, \cdots, w-1\}$. Otherwise, alternative destination stacks are selected from the set $\{w+1, \cdots, W\}$ so that the stack m will be selected if and only if for each stack $n \in \{1, \cdots, w-1, w+1, \cdots, m-1\}$, the lowest number in stack m is lower than that of the stack n where relocating q to stack m is known as a BG relocation. In this case, if more than one stack satisfies such conditions, the stack with less number is

The smallest numbers in each stack:	14	13	PS	12	9	15	11	N+1
Stack number:	1	2	3	4	5	6	7	8

Figure 4. The smallest numbers in the stacks of a typical bay.

chosen to be searched sooner.

After the list Q is completed, the new children will be created so that for each stack $s \in Q$, the time of relocating q to the stack s is added to t_{cw} and the N_m is increased by one unit. If $N_m \leq UB$, then the relocation will be done and we have to return one level back afterwards; otherwise, t_{cw} and N_m must be updated.

To illustrate the branching strategy in CSUM, an example of a typical bay is considered next. Let $q = 10$, $W = 8$, and $w = 3$ and also assume that the smallest number in each stack of the bay is as given in Figure 4. According to CSUM, the alternative destination stacks are 2, 4, and 7. Note that although stacks 1, 6, and 8 provide BG relocations, none of them is selected. In this example, stack 8 is empty and, therefore, according to Definition 3, the lowest number for it will be $N + 1$.

5. Numerical results

In this section, in order to evaluate the performance of the proposed algorithm, i.e. CSUM, we have compared its performance with the best known approaches presented in the literature through the numerical experiments. To this purpose, the CSUM is implemented on a personal computer with a 2.4 GHz Intel Core i3 CPU with 4 GB of RAM. In this study, in order to compare the performance of the proposed heuristic with the existing approaches, we have performed the comparison over the test problems presented in other studies. Thus, two CRP test problems taken from the literature are considered in this study. The first set of instances is given by Ünlüyurt and Aydın [16], which can be accessed upon request and includes 640 instances. The second problem set is given by Lee and Lee [6] that can be downloaded from http://sites.google.com/site/smallcontainerworld/,which includes 14 instances.

Ünlüyurt and Aydın [16] proposed a Branch and Bound (B&B) algorithm and also three alternative heuristic algorithms, called Grdy2, Diff2, and EAR2. In Table 2, the computational results of the CSUM in comparison to those obtained by the proposed algorithms in [16] are presented. The first column refers to the Instance Type (IT), which corresponds to 40 instances with the same parameters. Columns named PRI (i.e. columns 4, 6, and 8 from the left) refer to the Percent Relative Improvement (PRI) of the

Table 2. Benchmarking CSUM against the approaches proposed by Ünlüyurt and Aydın [16].

				N_{MOV}					
IT	**CSUM**	**Grdy2**	**PRI**	**Diff2**	**PRI**	**EAR2**	**PRI**	**B&B**	**Gap**
b-5-6-65	28.05	28.50	-1.58	28.2	-0.53	29.53	-5.01	28.03	0.07
b-5-7-60	30.18	30.68	-1.65	30.15	0.08	31.63	-4.60	30.15	0.08
b-6-5-55	24.53	25.35	-3.25	24.75	-0.91	26.08	-5.96	24.48	0.18
b-6-6-60	32.65	33.83	-3.49	33.05	-1.21	35.28	-7.45	32.65	0.00
u-5-6-65	28.83	29.58	-2.54	29.05	-0.76	30.63	-5.88	28.83	0.00
u-5-7-60	30.6	31.03	-1.39	30.78	-0.58	32.65	-6.28	30.6	0.00
u-6-5-55	24.63	25.18	-2.20	24.93	-1.22	26.45	-6.90	24.6	0.10
u-6-6-60	32.35	33.55	-3.58	32.58	-0.71	35.15	-7.97	32.3	0.15
Average			-2.46		-0.73		-6.26		0.07
				T_{CW}					
IT	**CSUM**	**Grdy2**	**PRI**	**Diff2**	**PRI**	**EAR2**	**PRI**	**B&B**	**Gap**
b-5-6-65	298.95	304.9	-1.95	314.8	-5.04	314.6	-4.98	289.93	3.11
b-5-7-60	344.88	353.7	-2.50	365.1	-5.53	360.3	-4.28	337.05	2.32
b-6-5-55	241.93	253.4	-4.53	256.9	-5.83	253.5	-4.58	236.00	2.51
b-6-6-60	345.55	361.8	-4.49	373.4	-7.45	361.1	-4.31	331.40	4.27
u-5-6-65	309.73	318.5	-2.76	319.2	-2.95	330.8	-6.37	299.90	3.28
u-5-7-60	350.75	358.5	-2.16	369.7	-5.13	370.9	-5.42	341.20	2.80
u-6-5-55	243.68	253.7	-3.95	259.7	-6.16	258.9	-5.88	238.90	2.00
u-6-6-60	344.15	361.2	-4.73	367.4	-6.34	369.7	-6.91	335.58	2.55
Average			-3.4		-5.6		-5.3		2.86

CSUM in comparison to the Grdy2, Diff2, and EAR2 methods, respectively, for both objective functions. The last column indicates the percent of the relative gap between the CSUM and the B&B algorithm. As it can be observed from this table, CSUM has outperformed Grdy2, Diff2, and EAR2, since there are 3.4%, 5.6%, and 5.3% decreases on average in the objective T_{CW} for CSUM in comparison to Grdy2, Diff2, and EAR2, respectively. Similarly, there are 2.46%, 0.73%, and 6.26% decreases on average in the objective N_{MOV} for CSUM in comparison to Grdy2, Diff2, and EAR2, respectively. In addition, the average optimality gap for CSUM over all groups of instances, which is calculated in comparison to B&B, are 0.07% and 2.86% for T_{CW} and N_{MOV}, respectively, which are quite small. The interesting point here is that although the optimality gap for objective N_{MOV} is negligible, it is considerable for objective T_{CW}. This result supports our approach in using objective T_{CW} rather than objective N_{MOV}, in particular when the problem size increases, e.g., when considering blocks with more than one bay. The allotted time to address each category of the instances was one second. The computational times reported throughout this paper are all in seconds.

Furthermore, in this paper, CSUM is also benchmarked against the approach proposed by Lee and Lee [6], briefly LL, and the comparative results are provided in Table 3. In these instances, the time limit allotted to find an initial solution using GBH is considered to be three seconds. After an initial solution is found, CSUM is allowed at most 5 seconds to improve the current solution. Therefore, in Tables 3 and 4, the summation of times spent by GBH and CSUM together is reported, which is at most $3 + 5$ seconds. In Table 3, the second columns indicate the lower-bound values of the test problems for the initial layouts and the columns named PRI represent the percent relative improvements in T_{CW} and N_{MOV}, respectively. In Tables 3 and 4, the CT columns indicate the computational time. CSUM has resulted in fewer movements in all 14 instances and the PRI for N_{MOV} is on average higher than 12%. With respect to the considered main objective, i.e. T_{CW}, CSUM has on

Table 3. Benchmarking CSUM against the approach proposed by Lee and Lee [6].

ID	LB	N_{MOV}		PRI	T_{CW} (s)		PRI	CT (s)	
		LL [6]	CSUM		LL [6]	CSUM		LL [6]	CSUM
R011606_0070_001	100	118	107	-9.3	10832	8010	-26.05	6304	2.0
R011606_0070_002	104	117	108	-7.7	10840	8116	-25.13	11081	2.2
R011606_0070_003	104	110	108	-1.8	10326	8243	-20.17	5502	0.2
R011606_0070_004	108	158	115	-27.2	13823	8629	-37.58	9026	0.3
R011606_0070_005	106	124	110	-11.3	11406	8252	-27.65	9108	0.7
U011606_0070_001	125	125	125	0	11341	9090	-19.85	17326	2.4
U011606_0070_002	124	130	128	-1.5	11737	9397	-19.94	11243	5.1
R011608_0090_001	143	190	151	-20.5	16773	11351	-32.33	13269	5.1
R011608_0090_002	139	191	151	-20.9	16883	11250	-33.36	11135	5.4
R011608_0090_003	142	216	155	-28.2	18857	11653	-38.20	21583	8.0
R011608_0090_004	143	178	151	-15.2	15922	11214	-29.57	7042	5.1
R011608_0090_005	143	182	149	-18.1	16281	11202	-31.20	13738	5.1
U011608_0090_001	164	175	166	-5.1	15715	12371	-21.28	21587	5.1
U011608_0090_002	164	180	168	-6.7	16236	12613	-22.31	8021	5.1
Average		157	135	-12.4	14069	10099.4	-27.47	11855	3.7

Table 4. Benchmarking CSUM against the approach proposed by Forster and Bortfeldt [15].

ID	N_{MOV}			T_{CW} (s)			CT (s)	
	FB [15]	CSUM	PRI	FB [15]	CSUM	PRI	CSUM	FB [15]
R011606-0070	108	109.6	1.5	9534.1	8250.0	-13.5	1.08	8.2
R011608-0090	148.4	151.4	2	12501.6	11334	-9.3	5.74	10.1
U011606-0070	125	126.5	1.2	10077.4	9243.6	-8.3	3.75	1.2
U011608-0090	167	167	0	13244.6	12492	-5.7	5.1	10
Average			1.18			-9.2	3.9	7.4

average 27.5% lower working time of crane. In addition, as the last two columns of Table 3 demonstrate, the CSUM algorithm is about 3000 times faster than the LL algorithm in terms of the running time.

Finally, CSUM is benchmarked against the tree search algorithm proposed by Forster and Bortfeldt [15], briefly called FB in this paper, and the comparative results are summarized in Table 4. In this table, the first and the last two rows correspond to a category of 5 and 2 instances, respectively, and the results are reported in terms of the average values. It is worth to mention that in spite of minor increase in the average of N_{MOV} by 1.18 for CSUM compared to FB algorithm, there is a significant improvement on average by %9.2 in the main objective, i.e. T_{CW}, which in turn shows the superiority of the CSUM.

6. Conclusions and future research directions

The container terminals are very complex systems. Thus, similar to other studies in the literature, this study focuses on the well-defined problem of such environments known as the Container Retrieval Problem (CRP). In this study, we have proposed a heuristic algorithm, called CSUM, for the CRP that has proved efficient in minimizing the overall working time of the crane (T_{CW}) required to complete the retrieval operations. The proposed approach was benchmarked against all the former solution approaches presented in the literature, which proved its superiority in terms of the time needed to complete the retrieval operations. In addition, the number of movements in the final solutions obtained by CSUM was close to their existing lower bounds and the way the containers were moved around was also efficient. Another advantage of the CSUM is that it is a fast heuristic considering its computational time, even in large instances, which may be found in the real-world applications. It should be noted that since we do not have access to the codes of other heuristics, we have performed the comparisons only over the datasets available in other related studies, of which the computational results were accessible.

This research can be extended in the following future research directions. Considering the fact that the proposed CSUM was developed for the blocks within a single bay, developing a solution approach similar to the CSUM for the block of containers stored in multiple bays can be the subject of future research. In addition, considering the effect of having more than one RMGC as well as using different types of cranes such as straddle carriers in a single bay can also be considered an interesting research topic. Moreover, in this study, we supposed that the traveling time of the crane within the bay was independent from other operations that might influence it. However, in the real-world environments, working time of the crane can be dependent upon the time needed to transfer the containers between the bay and the containership. Finally, the proposed algorithm can be modified to be applied in similar applications such as stacking boxes, pallets, and steel plates [3].

References

1. Tang, L., Xie, X. and Liu, J. "Crane scheduling in a warehouse storing steel coils", *IIE Transactions*, **46**(3), pp. 267-282 (2013).
2. Xie, X., Zheng, Y. and Li, Y. "Multi-crane scheduling in steel coil warehouse", *Expert Systems with Applications*, **41**(6), pp. 2874-2885 (2014).
3. Kim, K.H. and Hong, G.-P. "A heuristic rule for relocating blocks", *Computers & Operations Research*, **33**(4), pp. 940-954 (2006).
4. Zhu, W., Qin, H., Lim, A. and Zhang, H."Iterative deepening A* algorithms for the container relocation problem", *Automation Science and Engineering, IEEE Transactions on*, **9**(4), pp. 710-722 (2012).
5. Forster, F. and Bortfeldt, A. "A tree search procedure for the container relocation problem", *Computers & Operations Research*, **39**(2), pp. 299-309 (2012).
6. Lee, Y. and Lee, Y.-J. "A heuristic for retrieving containers from a yard", *Computers & Operations Research*, **37**(6), pp. 1139-1147 (2010).
7. Caserta, M., Schwarze, S. and Voß, S. "A mathematical formulation and complexity considerations for the blocks relocation problem", *European Journal of Operational Research*, **219**(1), pp. 96-104 (2012).
8. Caserta, M., Schwarze, S. and Voß, S. "A new binary description of the blocks relocation problem and benefits in a look ahead heuristic", *Evolutionary Computation in Combinatorial Optimization*, C. Cotta and P. Cowling, Eds., Springer Berlin/Heidelberg. pp. 37-48 (2009).
9. Caserta, M., Voß, S. and Sniedovich, M. "Applying the corridor method to a blocks relocation problem", *OR Spectrum*, **33**(4), pp. 915-929 (2011).
10. Sniedovich, M. and Voß, S. "The corridor method: a dynamic programming inspired metaheuristic", *Control Cybern.*, **35**, pp. 551-578 (2006).
11. Jovanovic, R. and Voß, S. "A chain heuristic for the blocks relocation problem", *Computer & Industrial Engineering*, **75**, pp. 79-86 (2014).
12. Eskandari, H. and Azari, E. "Notes on "A mathematical formulation and complexity considerations for the blocks relocation problem"", *Scientia Iranica*, **22**(6), pp. 2722-2728 (2015).
13. Petering, M.E.H. and Hussein, M.I. "A new mixed integer program and extended look-ahead heuristic

algorithm for the block relocation problem", *European Journal of Operational Research*, **231**, pp. 120-130 (2013).

14. Zehendner, E., Caserta, M., Feillet, D., Schwarze, S. and Voß, S. "An improved mathematical formulation for the blocks relocation problem", *European Journal of Operational Research*, **242**(2), pp. 415-422 (2015).
15. Forster, F. and Bortfeldt, A. "A tree search heuristic for the container retrieval problem", In *Operations Research Proceedings 2011*, D. Klatte, H.-J. Lüthi, and K. Schmedders, Eds., Springer Berlin Heidelberg, pp. 257-262 (2012).
16. Ünlüyurt, T. and Aydın, C. "Improved rehandling strategies for the container retrieval process", *Journal of Advanced Transportation*, **46**(4), pp. 378-393 (2012).
17. Caserta, M., Schwarze, S. and Voß, S. "Container rehandling at maritime container terminals", In *Handbook of Terminal Planning, Operations Research/Computer Science Interfaces Series*, J. Böse, Ed., **49**, Springer, New York, pp. 247-269 (2011).

Biographies

Esmaeel Azari received a BS degree from the Department of Applied Mathematics at University of Hakim Sabzevari, Iran, in 2009; and an MS degree in Industrial Engineering from Tarbiat Modares University, Tehran, Iran, in 2012. He is currently a Lecturer with the Department of Industrial Engineering at Payame Noor University, Shirvan, North Khorasan, Iran. His research interests include discrete optimization problems related to operations research as well as developing heuristic algorithms.

Hamidreza Eskandari received his BS degree in Electrical Engineering from University of Tehran, Iran, in 1998; MS degree in Socio-Economic Systems Engineering from the Iran University of Science and Technology in 2001; and PhD degree in Industrial Engineering from the University of Central Florida, USA, in 2006. He is currently Assistant Professor of Industrial Engineering and Director of the Advanced Simulation Laboratory at Tarbiat Modares University, Tehran, Iran. His research interests include applied operations research, simulation modeling and analysis, simulation optimization, and evolutionary multiobjective optimization.

Amir Nourmohammadi received his BS In Industrial Engineering from Iran University of Science and Technology, Tehran, Iran, in 2005; and his MS degree in Industrial Engineering from Islamic Azad University, Qazvin, Iran, in 2009. He is currently a PhD candidate of Industrial Engineering at Tarbiat Modares University, Tehran, Iran. His research interests include logistics and supply chain management, assembly line balancing, part feeding, multi-objective modeling and optimization, application of soft computing techniques, multi-criteria decision making, quality engineering, and simulation.