Heuristic Methods for the Unrestricted Dynamic Container Relocation Problem

M. Hakan AKYÜZ

Abstract—**In this work, we address the unrestricted Dynamic Container Relocation Problem (DCRP). The DCRP is an extension of the Container Relocation Problem (CRP) that is NP-hard. The CRP aims to empty a single yard-bay, which contains containers with a given retrieval sequence from the yard-bay, such that the total number of relocations is minimized. The DCRP extends the CRP so that containers both arrive and depart at the yard-bay. The unrestricted DCRP relaxes a common assumption which enforces container relocations are performed only to retrieve containers that needs to leave the yard-bay. A mathematical formulation is suggested for the unrestricted DCRP. Then, three efficient heuristic methods are offered. An extensive set of computational experiments is performed on standard test instances. Our results indicate that savings can be achieved by using the suggested heuristic methods.**

Index Terms—**Container relocation, cleaning moves, heuristics, integer programming.**

I. Introduction

THE global containerized cargo flow have increased to a total of 171 million Twenty-Equivalent Units (TEUs) in 2014 [1]. Excluding the effect of the negative growth of world's economy in 2008, the volume of container trade has an average annual growth of over 5% since 1996 [1]. This implies that container terminals are expected to operate more efficiently to meet increasing demand for cargo handling. Additionally, container liner shipping companies continue to order mega container vessels requiring container terminal operations to be managed well. The container terminal area can be divided into two areas: quay side and yard side. Berth allocation, quay crane assignment and scheduling, and vessel storage planning can be considered as quay side operations. Transfer of containers from quay side, scheduling of yard cranes, and storage allocation and placement of containers at the yard storage area constitute yard side operations. This work focuses on the yard side operations. In particular, we address the unrestricted Dynamic Container Relocation Problem (DCRP) which arises in a single yard-bay in container terminal yard areas.

The DCRP generalizes the so called "static" Container Relocation Problem (CRP) which is shown to be NP-hard [2]. The CRP tries to minimize the total number of relocations required to clear out a single yard-bay with a container storage capacity of C columns (stacks) and H rows (tiers) where initially N containers exist in the yard-bay with known retrieval order. In addition to container retrievals, the DCRP allows container arrivals at the yard-bay resulting in a more realistic problem than the CRP. The DCRP aims to minimize the total number of relocations in a yard-bay with given arrival and retrieval sequences of N containers.

Manuscript received December 22, 2016; revised January 11, 2017. This work was fully supported by the Turkish Scientific and Technological Research Council (TÜBİTAK) with Research Grant No: 214M222.

M. Hakan AKYÜZ is with the Department of Industrial Engineering, Galatasaray University, Ortaköy, İstanbul, 34357, TÜRKİYE (phone: +90212-2274480(436); fax: 90212-2595557; e-mail: mhakyuz@gsu.edu.tr).

The CRP is classified as the restricted and unrestricted CRP with respect to the existence of its original assumptions given by [3]. The restricted problem assumes that the relocations can only be performed to take a container out of the yard-bay. On the other hand, unrestricted CRP relaxes this assumption and allows pre-marshaling operations to reduce future relocations. In this work, we address the unrestricted DCRP. To the best of our knowledge, this is the first study addressing the unrestricted DCRP. The contribution of this study can be summarized as follows. First, we propose a mathematical programming formulation for the unrestricted DCRP. Second, we develop three heuristic approaches for the unrestricted DCRP. Third, we perform our computational experiments on a standard set of test instances from the literature. Our results imply that the suggested heuristic methods are able to produce promising outcomes for the unrestricted DCRP. In what follows, we give a brief review of the relevant literature for the CRP and the DCRP.

There exist a considerable amount of studies that consider the CRP within the last two decades. Kim et al. [4] aims to minimize the expected number of relocations for the case of grouped containers based on their weights. Kim and Hong [3] offer a branch and bound (BB) algorithm and employ a rule based heuristic approach for the CRP. Wan et al. [5] is the first study offering a mathematical programming formulation for the CRP where the authors present fast heuristics for the CRP. Caserta et al. [6] use a metaheuristic algorithm named as "corridor method" that is based on dynamic programming. A tree search based solution procedure is given by [7]. The CRP is shown to be NP-hard by [2]. Caserta et al. [2] offer two mathematical programming formulations: the first solves the unrestricted CRP and the second solves the restricted CRP. Ünlüyurt and Aydın [8] consider a different objective of minimizing the total time to empty a yard-bay with heuristic procedures. Petering and Hussein [9] develop an improved formulation and a look-ahead heuristic method for the unrestricted CRP. Jovanovic and Voß [10] suggest chain heuristics that employ the Max-Min (MM) algorithm used by [2] for the CRP. The work by [11] is an excellent survey on CRP and related studies. Jin et al. [12] offer a greedy look-ahead heuristic approach for solving both restricted and unrestricted CRP. Zehendner et al. [13] propose an improved CRP formulation which works efficiently. Recently, Ku and Arthanari [14] employ an exact abstraction method that can significantly reduce the search space of the CRP.

Wan et al. [5] is the first study addressing the DCRP. Rei and Pedroso [15] proves that the DCRP is NP-hard. Akyüz and Lee [16] suggests the first mathematical programming formulation for the DCRP. The authors [16] also implement efficient heuristic procedures for the DCRP. Konig et al. [17] focuses on a similar problem of stacking steel slabs. Casey and Kozan [18] deals with a DCRP having a different objective of minimizing the total processing time of the straddle carrier serving a single yard-bay. Borjian et al. [19] study a variant of the DCRP by introducing a class of flexible service policies to make minor changes in the order of container retrievals. Recently, Zhang et al. [20] focus on a CRP where containers can be handled in batches (i.e., two containers simultaneously).

The rest of this work is organized as follows. In Section 2, we introduce the unrestricted DCRP and its mathematical programming formulation. Section 3 presents three straightforward heuristic approaches for the unrestricted DCRP. This is followed by Section 5 where we present our computational results. Conclusions are presented in Section 6.

II. The Unrestricted Dynamic Container Relocation Problem

A yard-bay contains C container stacks (or *columns*). Each column has a capacity to accumulate H tiers (or *rows*) of containers on top of each other. Slot is the term used to state the space occupied by a container that is defined by its column and row number. A yard crane serves the yard-bay of interest and it has access to the containers only from the top slot of a column. Hence, when a *target container* is to be retrieved from the yard-bay, the yard crane should first remove all the containers above it. The containers above the target containers are misplaced and they called as *blocking containers*. A blocking container needs to be repositioned into another column within the yard-bay in order to reach a target container. This repositioning of a blocking container is denominated as a *relocation*. In addition, a container is an *arrival* (*retrieval*) *container* when a container is to arrive (depart) at the yard bay. Then, the unrestricted DCRP aims to minimize the total number of relocations performed to handle N containers whose arrival/departure times at the yard-bay are known in advance. The restricted DCRP has the following assumptions given by [16]: "*A1: The yard-bay is served by one yard crane which can handle a single container at a time. A2: The sequences of the events described by arrival and retrieval of containers are known a priori. A3: Relocations are allowed only at the departure time period of the containers and a container can be relocated at most once at a time period. A4: Relocations can occur only within the yard-bay. A5: Containers are of the same type in the yard-bay.*" This work focuses on the unrestricted DCRP in which the assumption *A3* is relaxed. Removing assumption *A3* permits pre-marshaling operations between container arrivals and departures to reduce the number of future relocations that has to be made in the existence of *A3*.

Let $t = 1, \ldots, T$ denotes the maximum allowed number of time periods that a yard crane is in service. Define T_i^a and T_i^d as the arrival and departure time of container i that can join/depart at the yard-bay. Here, it is assumed that a container handling operation is completed within a predefined amount of time shown as p. That is, total service time for the yard crane is $p \cdot T$. The following decision variables are used. x_{ic}^t is a binary decision variable which takes a value of 1 if and only if container i is at column c at time t and zero otherwise. Similarly, y_{ih}^t equals 1 if and only if container i is at height h at time t and zero otherwise. The binary variable v_{ich}^t becomes 1 if and only if container i is in column c at height h at time t which keeps the position of a container i using binary variables x_{ic}^t and y_{ih}^t. Lastly, the binary variables r_i^t, a_i^t and d_i^t take a value of 1 if and only if container i is relocated at time t, container i arrives to the yard-bay after time t and container i leaves the yard-bay before time t, respectively, and zero otherwise. Now, the mathematical programming formulation of the unrestricted DCRP can be given as follows.

UDCRP:

$$\min z = \sum_{i=1}^{N}\sum_{t=1}^{T} d_i^t - a_i^t \tag{1}$$

$s.t.$

$$\sum_{i=1}^{N} v_{ich}^t \le 1 \qquad c = 1, ..., C; h = 1, ..., H; t = 1, ..., T \tag{2}$$

$$\sum_{i=1}^{N} x_{ic}^t \le H \qquad c = 1, ..., C; t = 1, ..., T \tag{3}$$

$$\sum_{i=1}^{N} y_{ih}^t \le C \qquad h = 1, ..., H; t = 1, ..., T \tag{4}$$

$$x_{ic}^t + y_{ih}^t \le v_{ich}^t + 1 \qquad i = 1, ..., N; c = 1, ..., C; h = 1, ..., H; t = 1, ..., T \tag{5}$$

$$v_{ich}^t \le x_{ic}^t \qquad i = 1, ..., N; c = 1, ..., C; h = 1, ..., H; t = 1, ..., T \tag{6}$$

$$v_{ich}^t \le y_{ih}^t \qquad i = 1, ..., N; c = 1, ..., C; h = 1, ..., H; t = 1, ..., T \tag{7}$$

$$\sum_{c=1}^{C} x_{ic}^t = d_i^t - a_i^t \qquad i = 1, ..., N; t = 1, ..., T \tag{8}$$

$$\sum_{h=1}^{H} y_{ih}^t = d_i^t - a_i^t \qquad i = 1, ..., N; t = 1, ..., T \tag{9}$$

$$\sum_{i=1}^{N} v_{ic(h+1)}^t \le \sum_{i=1}^{N} v_{ich}^t \qquad c = 1, ..., C; h = 1, ..., H-1; t = 1, ..., T \tag{10}$$

$$\sum_{i=1}^{N} r_i^t \le 1 \qquad t = 1, ..., T \tag{11}$$

$$x_{ic}^t - x_{ic}^{t+1} \le r_i^t \qquad i = 1, ..., N; c = 1, ..., C; t = 1, ..., T-1 \tag{12}$$

$$x_{ic}^{t+1} - x_{ic}^t + \le r_i^t \qquad i = 1, ..., N; c = 1, ..., C; t = 1, ..., T-1 \tag{13}$$

$$y_{ih}^t - y_{ih}^{t+1} \le r_i^t \qquad i = 1, ..., N; h = 1, ..., H; t = 1, ..., T-1 \tag{14}$$

$$y_{ih}^{t+1} - y_{ih}^t \le r_i^t \qquad i = 1, ..., N; h = 1, ..., H; t = 1, ..., T-1 \tag{15}$$

$$x_{ic}^t + x_{ic}^{t+1} + r_i^t \le 2 \qquad i = 1, ..., N; c = 1, ..., C; t = 1, ..., T-1 \tag{16}$$

$$x_{ic}^t + a_i^t \le 1 \qquad i = 1, ..., N; c = 1, ..., C; t = 1, ..., T \tag{17}$$

$$y_{ih}^t + a_i^t \le 1 \qquad i = 1, ..., N; h = 1, ..., H; t = 1, ..., T \tag{18}$$

$$x_{ic}^t \le d_i^t \qquad i = 1, ..., N; c = 1, ..., C; t = 1, ..., T \tag{19}$$

$$y_{ih}^t \leq d_i^t \quad i=1,...,N; h=1,...,H; t=1,...,T \tag{20}$$

$$\sum_{t=1}^{T} a_i^t \leq \left\lceil \frac{T_i^a}{p} \right\rceil \quad i=1,...,N \tag{21}$$

$$\sum_{t=1}^{T} d_i^t \leq \left\lceil \frac{T_i^d}{p} \right\rceil \quad i=1,...,N \tag{22}$$

$$a_i^t \leq a_i^{t-1} \quad i=1,...,N; t=2,...,T \tag{23}$$

$$d_i^t \leq d_i^{t-1} \quad i=1,...,N; t=2,...,T \tag{24}$$

$$a_i^t \leq a_j^t \quad i,j=1,...,N : T_i^a < T_j^a; t=1,...,T \tag{25}$$

$$d_i^t \leq d_j^t \quad i,j=1,...,N : T_i^d < T_j^d; t=1,...,T \tag{26}$$

$$1+\sum_{t=1}^{T} a_i^t \leq \sum_{t=1}^{T} a_j^t \quad i,j=1,...,N : T_i^a < T_j^a \tag{27}$$

$$1+\sum_{t=1}^{T} d_i^t \leq \sum_{t=1}^{T} d_j^t \quad i,j=1,...,N : T_i^d < T_j^d \tag{28}$$

$$ta_i^t \leq \sum_{\bar{t}=1}^{T} a_i^{\bar{t}} \quad i=1,...,N; t=1,...,T \tag{29}$$

$$td_i^t \leq \sum_{\bar{t}=1}^{T} d_i^{\bar{t}} \quad i=1,...,N; t=1,...,T \tag{30}$$

$$x_{ic}^t \in \{0,1\} \quad i=1,...,N; c=1,...,C; t=1,...,T \tag{31}$$

$$y_{ih}^t \in \{0,1\} \quad i=1,...,N; h=1,...,H; t=1,...,T \tag{32}$$

$$v_{ich}^t \in \{0,1\} \quad i=1,...,N; c=1,...,C; \\ h=1,...,H; t=1,...,T \tag{33}$$

$$r_i^t, a_i^t, d_i^t \in \{0,1\} \quad i=1,...,N; t=1,...,T \tag{34}$$

The objective function (1) tries to minimize the total handling time of the yard crane to serve N containers. Constraints (2) ensure that at most one container is placed within a slot of the yard-bay. Constraints (3) and (4) are capacity constraints stating that the maximum height of each column is limited by the number of rows H and the maximum number of columns at each tier is C, respectively. Constraints (5), (6) and (7) define the relationship between binary variables indicating the location of containers. Constraints (8) and (9) indicate that a container occupies a space within the yard-bay if and only if it has joined and did not leave the yard-bay. Thus, if $d_i^t - a_i^t$ is positive, then container i exists within the yard-bay. Otherwise, it does not occupy a slot. Constraints (10) guarantee that containers do not float in the air. A slot becomes available for placement as long as there are other containers underneath or it is at the ground level. Constraints (11) imply that at most one relocation can be performed in a time period t. Constraints (12), (13), (14) and (15) enforce that a container maintains its current position in the yard-bay for the next time period $t+1$ if it is not relocated. Constraints (16) state that when a container is relocated at time t, then it can not be located at the same column in the next time period $t+1$. When a container has not arrived at the yard-bay (i.e., $a_i^t = 1$), it can not occupy a space within the yard-bay (i.e. $x_{ic}^t = 0$ for all $c = 1, ..., C$) by constraints (17) and (18). Similarly, if a container has left the yard-bay (i.e., $d_i^t = 0$), then it can not exists within the yard-bay (i.e., $x_{ic}^t = 0$) by constraints (19) and (20). Constraints (21) and (22) respectively restrict the joining time of a container to be earlier than or equal to its arrival time and the retrieving time of a container to be later than or equal to its departure time at the yard-bay. Once a container joins the yard-bay at time t (i.e., $a_i^t = 0$) constraints (23) prevent that it can join the yard-bay again for the time periods after t (i.e., $t+1, t+2$ etc.). Similarly, constraints (24) is for the case of container departures from the yard-bay. Constraints (25), (26), (27) and (28) help maintaining the sequence to join and leave the yard bay with respect to the arrival and retrieval times of containers. Namely, when container i would join the yard-bay earlier than a container j, i.e., $T_i^a < T_j^a$, it guarantees that container j is accepted later than container i. When a container i joins the yard-bay at time t (i.e., $a_i^t = 0$), constraints (29) provide that $a_i^t = 0$ for all remaining time periods until T. When a container i leaves the yard-bay at time t (i.e., $d_i^t = 0$), constraints (30) ensure that $d_i^t = 0$ for all remaining time periods until T. Lastly, the binary restrictions are given by constraints (31)–(34).

III. Heuristic Methods for the Unrestricted DCRP

The UDCRP is a binary programming formulation which quickly becomes intractable as the number of time periods T, the number of containers N to be served and the size of the yard-bay increase. As a remedy, we suggest three heuristic methods in the following.

A. The Unrestricted Dynamic Reshuffling Index Heuristic

Reshuffling Index (RI) heuristic is originally proposed by [21] for the CRP. When a container is to be relocated, a reshuffling index, i.e., RI, is calculated for each column c. RI is determined as the number of blocking containers in a column. Notice that, blocking containers need not consist of the ones blocking only the earliest container that leaves the yard-bay. Once the retrieval container is taken out of the yard-bay, there can be other blocking containers underneath. Therefore, RI of a column c is the total number of blocking containers in column c. The RI is adapted for the restricted DCRP by [16] such that it also considers the arrival containers to obtain RI of columns. For the unrestricted DCRP, relocations can be performed not only to retrieve a container from the yard-bay, but also between the retrieval (or arrival) of containers. The term *cleaning move* is used to define the relocations executed to clean the blocking containers within the yard-bay and place them in a column where they are no longer a blocking container. Recall that, blocking containers are misplaced so that they have to be relocated to reach a target container. Cleaning moves have a chance to reduce the number of future relocations by allowing pre-marshaling of containers. Then, the Unrestricted Dynamic Reshuffling Index (UDRI) heuristic performs cleaning moves by letting the relocation of containers between retrieval and arrival of containers. The RI of a container that is going to be cleared (i.e., by a cleaning move) is also calculated as described. Similarly, for a retrieval or arrival container RI is determined for each column. The location of a container is decided such that it is assigned to the column which has the lowest RI value among all available columns. Clearly, a column c is not available if it is full or it is the same column with the container to be relocated. When there is a tie among the

RI values of the columns, the container is assigned to the highest column. In case of a further tie for the height of the columns, then the selection of a column is made randomly among the tie columns. The UDRI heuristic attempts to find if there exists a cleaning move before the retrieval (or arrival) of a target container. When there exist such a cleaning move, UDRI first performs the cleaning move and then focuses on the retrieval (or arrival) container. We limit ourselves with up to two cleaning moves in between retrieval (or arrival) of containers. The RI values are determined for each container relocation as mentioned. The UDRI heuristic ends when all containers are handled.

B. The Unrestricted Dynamic Min-Max Heuristics

The Min-Max (MM) heuristic is first suggested by [2] for the CRP. Then, it is further improved by [10]. The MM heuristic aims to prevent new relocations that may be created by blocking containers. That is to say, a blocking container may continue to be a blocking container after it is relocated in reaching a target container. So the MM heuristic tries to refrain from creating new blocking containers as much as possible. For that purpose, the MM heuristic employs priorities that are defined as the inverse order of the retrieval sequence for containers. A container has a higher priority when its departure time is earlier than other containers. The relocation of blocking containers are performed depending on their priorities. The columns are also assigned a priority level that is evaluated as the priority of the earliest departing container (i.e. the priority of the container with the highest priority) within the column. Empty columns has the lowest priority and the heuristic tries to save empty columns for later departing containers when possible. A blocking container is first relocated to a column with a lower priority than the blocking container. It is possible that there are more than one such columns. In this case, the blocking container is placed in the column having the highest priority. On the other hand, when there is no column with a lower priority than the blocking container, this implies that the blocking container needs to be relocated one more time after it is repositioned. In such a case, the blocking container is placed to the column having the lowest priority (i.e., the column whose earliest container leaves the yard-bay latest among all columns). Thus, the next relocation of the blocking container occurs as late as possible until the earliest departing container has to leave the yard-bay.

The first Unrestricted Dynamic Min-Max (UDMM1) heuristic inherits the MM heuristic ideas. In the unrestricted DCRP, arrival of the containers are taken into account such that the MM heuristic rule is applied on the arrival container. Namely, deciding the position of arrival containers need additional effort. Arrival containers can be placed to all columns having an empty slot is evaluated. Here, notice that, the column of a blocking is excluded from consideration when it is to be relocated. Similar to the UDRI heuristic, UDMM1 heuristic allows cleaning moves between container retrievals and arrivals. The containers on which cleaning move is applied undergo the MM heuristic rules as described. The number of cleaning moves is restricted to be at most two moves at each container retrieval and arrivals.

The second UDMM (UDMM2) heuristic benefits from the improvement offered by [10] on the MM heuristic. [10] noticed that when a blocking container is relocated and it continues to be a blocking container, it would be better to reposition the blocking container in a column which has at most $H-2$ containers. This implies that the new column of a blocking container is not full for the next stages. In such a case, the corresponding column maintains its chance to host one more container with the hope that its priority is higher than the column priority. This modification is employed for our implementation of the UDMM2 heuristic. The remaining steps of the UDMM2 heuristic is the same as the UDMM1 heuristic.

IV. Computational Experiments

In this section, we first give our computational results on the performance of the UDCRP formulation. Next, we present the outcome of our heuristic methods on standard test instances. Our experiments are performed on a Dell Precision T5810 workstation with Intel(R) Xeon(R) E5-1650v3 processor of 3.50 GHz and 64 GB RAM operating within Microsoft Windows 7 Pro 64-bit environment. The callable library of Gurobi 5.6.3 with default settings is used to solve the UDCRP formulation and all codings are written in C^{++} programming language.

A. The Performance of the UDCRP Formulation

Table I presents the performance of our UDCRP formulation on standard test instances with heavy container traffic by [16]. The first column states the size of the test instances for which there exists C columns and H rows in the yard-bay as well as the number of time periods T to retrieve five containers from the yard-bay. Here, each cell gives the average of five test instances. The second column stands for the upper bound value obtained. The third column shows the number of relocations performed within the yard-bay. The last column gives the average CPU times in seconds. The test instance, that is marked with *, having 6 columns and 6 rows yielded an output only for three out of five test instances. The performance of the UDCRP formulation quickly deteriorates even on small instances. Next, we proceed to our results with the suggested heuristic methods.

TABLE I
The performance of the UDCRP formulation on standard test instances by [16].

Instance (C, H, T)	UB	Number of Relocations	CPU (s)
(6, 2, 19)	158.8	7.8	62.94
(6, 3, 23)	267.8	8	1037.17
(6, 4, 27)	336.4	8	3662.20
(6, 5, 21)	136.4	8	1541.69
(6, 6, 36)*	428.8	8	4833.53

B. The Performance of the Heuristic Methods for the UD-CRP

The performance of the UDRI, UDMM1 and UDMM2 heuristics is tested on standard test instances by [16]. These instances have Medium and High Density container traffic in the yard-bay and they are shortly indicated as MD and HD, respectively. The yard-bay has $C = 6$ columns and height H selected from the set $H \in \{2, 3, 4, 5, 6\}$. The number of containers that are retrieved from the yard-bay are chosen from the set $\{5, 50, 100, 200, 400, 800\}$. For each combination of density, height and number of containers 20 test instances exists, thus, a total of 1200 test instances are considered. The details on the generation of the test bed and its properties can be obtained from [16]. Table II shows the summary of the performance of the UDRI heuristic. The first column gives the property of the test instances the density as "MD" or "HD", the number of columns and rows of the yard-bay in parenthesis, i.e., (C,H). The UDRI heuristic is considered with at most one or two Cleaning Moves (CM), that are indicated as "1 CM" and "2 CM". The number of relocations made to finish container handling operations is given in the rows under "UB" and their corresponding CPU time is shown under "CPU". The UDRI heuristic is run for 100 times and the best upper bound is reported here. Notice that each cell states the average of 120 test instances in Table II as described. Average values of different densities are given in the row below the MD and HD instances. Note that, the values superior than other heuristic methods are marked in bold characters.

TABLE II
THE PERFORMANCE OF THE UDRI HEURISTIC ON STANDARD TEST INSTANCES BY [16].

Instance	UDRI (1 CM)		UDRI (2 CM)	
(C,H)	UB	CPU	UB	CPU
MD (6,2)	**0.15**	0.04	0.17	0.04
MD (6,3)	**3.83**	0.05	3.95	0.05
MD (6,4)	30.77	0.06	31.95	0.06
MD (6,5)	69.34	0.06	70.86	0.06
MD (6,6)	115.74	0.07	118.89	0.07
Average	43.97	0.06	45.16	0.06
HD (6,2)	40.18	0.05	40.88	0.04
HD (6,3)	102.33	0.09	105.09	0.07
HD (6,4)	159.35	0.08	163.03	0.08
HD (6,5)	241.71	0.09	247.18	0.09
HD (6,6)	318.12	0.10	323.68	0.10
Average	172.34	0.08	175.97	0.08

Table III stands for the summary of the performance of the UDMM1 and UDMM2 heuristics. The outline of the Table III is very similar to Table II. The CPU times of UDMM heuristics are negligible, i.e., 0.00, for all test instances. Therefore, we do not include them in Table III for the sake of brevity. The first column shows the instance properties as before. Remaining columns present the number of relocations (i.e., UB) performed to complete container handling operations. The UDMM1 and UDMM2 are also tested for 1 CM and 2 CM as described. Average values are given for each traffic density of the test instances in the row under them. The best performing heuristic is distinguished with bold characters. The performance of the heuristic methods is better for 1 cleaning move at most. It is observed that the performance of our heuristic methods deteriorates when the number of cleaning moves becomes greater than or equal to two CM. The UDRI heuristic with 1 CM yields slightly better results than the other heuristics on medium density instances with height of two and three rows in the yard-bay. Broadly speaking, MM type heuristics significantly outperform the RI based heuristic. The UDMM1 heuristic with 1 CM arises to be the best performing heuristic in high density test instances. Furthermore, the UDMM1 heuristic is the best for medium density instances with height of $H = 6$ rows. Nevertheless, the UDMM2 heuristic is superior than the others for medium density instances with height of four and five rows in the yard-bay. The unrestricted DCRP is a difficult problem to solve to optimality. Indeed, the UDCRP formulation can only be solved for very small instances. The heuristic methods proposed in this work quickly yields promising outcomes for the unrestricted DCRP.

TABLE III
THE PERFORMANCE OF THE UDMM1 AND UDMM2 HEURISTICS ON STANDARD TEST INSTANCES BY [16].

Instance	UDMM1		UDMM2	
(C,H)	(1 CM)	(2 CM)	(1 CM)	(2 CM)
MD (6,2)	0.28	0.28	0.28	0.28
MD (6,3)	3.96	3.97	3.85	3.90
MD (6,4)	23.23	23.38	**22.70**	23.13
MD (6,5)	49.42	49.87	**47.45**	48.70
MD (6,6)	**80.70**	82.96	81.47	81.73
Average	31.52	32.09	31.15	31.55
HD (6,2)	**35.70**	36.26	**35.70**	36.26
HD (6,3)	**84.75**	87.91	113.53	115.15
HD (6,4)	**125.83**	128.78	143.01	144.36
HD (6,5)	**183.96**	186.87	208.00	209.43
HD (6,6)	**236.43**	238.78	254.53	255.07
Average	133.33	135.72	150.95	152.05

V. CONCLUSION

In this work, we addressed the unrestricted DCRP. The unrestricted DCRP tries to handle arrival and retrieval of containers at a single yard-bay while minimizing total number of container relocations. Unlike the restricted DCRP, the unrestricted problem relaxes the assumption that relocations are only made at container retrievals (or arrivals). Therefore, unrestricted DCRP permits pre-marshaling operatios to reduce the number of future container relocations at the yard-bay. For all we know, the unrestricted DCRP is new to the literature. We suggested a mathematical programming formulation for the unrestricted DCRP. The performance of the UDCRP formulation is limited for solving small instances. Then, we offer efficient heuristics for the unrestricted DCRP. An extensive set of computational experiments is performed on standard test instances.

Three efficient heuristic methods are devised for the unrestricted DCRP. The first heuristic UDRI employs the RI idea by [21]. The second and third heuristics are enhancements of the known min-max heuristics for the unrestricted DCRP. Our results suggest that the UDMM1 heuristic yields better performance than the UDRI and UDMM2 heuristics in general. However, the UDRI and UDMM2 heuristics can be superior on medium density test instances. The suggested heuristic methods are very efficient to produce quick solutions at a yard-bay. Considering the fact that the container terminals are very busy and require fast decision making, our solution methods are also promising for practical usage.

Existing mathematical formulations for the unrestricted and restricted DCRP can only be used for small problems. Therefore, designing new formulations and exact solution approaches can be a good further research direction. This study only focused on rule based heuristic methods. Tree search based heuristics, i.e., beam search, for the unrestricted DCRP can be a fruitful future research area.

REFERENCES

[1] UNCTAD. Review of Maritime Transportation. (2015). http://unctad.org/en/PublicationsLibrary/rmt2015_en.pdf/ Accessed 01.12.2016.

[2] M. Caserta, S. Schwarze and S. Voß, “A mathematical formulation and complexity considerations for the blocks relocation problem,” *European Journal of Operational Research*, vol. 219, pp. 96-104, 2012.

[3] K. Kim and G. P. Hong, “A heuristic rule for relocating blocks,” *Computers & Operations Research*, vol. 33, pp. 940-954, 2006.

[4] K. H. Kim, T. P. Park and K. R. Ryu, “Deriving decision rules to locate export containers in container yards,” *European Journal of Operational Research*, vol. 124, pp. 89-101, 2000.

[5] Y. W. Wan, J. Liu and P. C. Tsai, “The assignment of storage locations to containers for a container stack,” *Naval Research Logistics*, vol. 56, pp. 699-713, 2009.

[6] M. Caserta, S. Voßand M Sniedovich, “Applying the corridor method to a blocks relocation problem,” *OR Spectrum*, vol. 33, pp. 915-929, 2011.

[7] F. Forster and A. Bortfeldt, “A tree search procedure for the container relocation problem,” *Computers & Operations Research*, vol. 39, pp. 299-309, 2012.

[8] T. Ünlüyurt and C. Aydın, “Improved rehandling strategies for the container retrieval process,” *Journal of Advance Transportation*, vol. 46, pp. 378-393, 2012.

[9] M. E. H. Petering and M. I. Hussein, “A new mixed integer program and extended look-ahead heuristic algorithm for the block relocation problem,” *European Journal of Operational Research*, vol. 231, pp. 120-130, 2013.

[10] R. Jovanovic and S. Voß, “A chain heuristic for the Blocks Relocation Problem,” *Computers & Operations Research*, vol. 75, pp. 79-86, 2014.

[11] J. Lehnfeld and S.Knust, “Loading, unloading and premarshalling of stacks in storage areas: survey and classification‘,” *European Journal of Operational Research*, vol. 239, pp. 297-312, 2014.

[12] B. Jin, W. Zhu and A. Lim, “Solving the container relocation problem by an improved greedy look-ahead heuristic,” *European Journal of Operational Research*, vol. 240, pp. 837-847, 2015.

[13] E. Zehendner, M. Caserta, D. Feillet, S. Schwarze and S. Voß, “An improved mathematical formulation for the blocks relocation problem,” *European Journal of Operational Research*, vol. 245, pp. 415-422, 2015.

[14] D. Ku and T. S. Arthanari, “On the abstraction method for the container relocation problem,” *Computers & Operations Research*, vol. 68, pp. 110-122, 2016.

[15] R. Rei and J. P. Pedroso, “Tree search for the stacking problem,” *Annals of Operations Research*, vol. 203, pp. 371-388, 2013.

[16] M. H. Akyüz, and C.-Y. Lee, “A mathematical formulation and efficient heuristics for the dynamic container relocation problem,” *Naval Research Logistics*, vol. 61, pp. 101-118, 2014.

[17] F. G. Konig, M. Lübbecke, R. Möhring, G. Schäfer and I. Spenke, “Solutions to real-world instances of PSPACE-complete stacking,” in *Lecture Notes in Computer Science: Algorithms - ESA 15th Annual European Symposium 2007*, pp. 729-749.

[18] B. Casey and E. Kozan, “Optimising container storage processes at multimodal terminals,” *Journal of the Operational Research Society*, vol. 63, pp. 1126-1142, 2012.

[19] S. Borjian, V. H. Manshadi, C. Barnhart and P. Jaillet, “Managing Relocation and Delay in Container Terminals with Flexible Service Policies,” *arXiv preprint arXiv: 1503.01535* Accessed 01.12.2016.

[20] R. Zhang, S. Liu and H. Kopfer, “Tree search procedures for the blocks relocation problem with batch moves,” *Flexible Services and Manufacturing Journal*, vol. 28, pp. 397-424, 2016.

[21] K. G. Murty, J. Liu, Y. W. Wan and R. Linn, “A decision support system for operations in a container terminal,” *Decision Support Systems*, vol. 39, pp. 309-332, 2005.