On the integer programming formulation for the relaxed restricted container relocation problem

Bo Jin[a,b]

[a]*Institute of Future Networks, Southern University of Science and Technology, Shenzhen, China*
[b]*Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen, China*

Abstract

In a recent paper entitled "A new binary formulation of the restricted container relocation problem based on a binary encoding of configurations", Galle et al. (2018) introduced a new variant of the container relocation problem (CRP), named the relaxed restricted CRP, where every container can be relocated at most once for retrieving each target container. The authors also proposed a binary integer programming model for formulating the relaxed restricted CRP. In this paper, it is first shown that the proposed model contains two deficiencies in formulating the "last in, first out" (LIFO) policy. These deficiencies will cause the solutions obtained by the model to correspond to infeasible configurations or infeasible relocation sequences. Then, the LIFO policy is analyzed in detail and reformulated as linear constraints correctly. Lastly, the corrected integer programming formulation is presented. Computational experiments show that the corrected model dramatically reduces complexity and improves performance.

Keywords: Combinatorial optimization, Container relocation problem, Integer programming

1. Introduction

A container terminal provides temporary storage for containers to connect maritime and land transport in logistics chains. Due to space limitations, containers are piled up vertically in container yards. Each vertical pile of containers is referred to as a stack, and a row of stacks is referred to as a bay. If a container to be retrieved is not at the top of a stack, the blocking containers placed above must be relocated to other stacks first. Reduction of such unproductive relocations is crucial for improving the throughput of container handling in a container terminal. For this reason, the *container relocation problem* (CRP), also known as the *block(s) relocation problem* (BRP), has been extensively studied in the literature.

Following the seminal work of Kim & Hong (2006), most of the existing studies on the CRP concentrate on minimizing the total number of relocations for retrieving containers from a single bay. The bay consists of S stacks numbered $1, \dots, S$ from left to right and T tiers numbered $1, \dots, T$ from bottom to top. The initial bay configuration contains C containers, with each container being identified by a unique number $1, \dots, C$ indicating its retrieval order. At any moment in the retrieval process, the *target* container refers to the container currently in the bay with the smallest retrieval order. There are two main variants of the CRP in the existing literature. The *restricted* variant only allows the relocation of the containers above the target container, while the *unrestricted* variant does not impose such restriction.

In a recent paper, Galle et al. (2018) proposed a novel binary integer programming formulation called CRP-I for the restricted CRP, which was shown to significantly outperform other existing integer programming models from the literature. The key to its remarkable performance is that CRP-I uses an enhanced binary encoding based on the support relations between containers, which was originally introduced by Caserta et al. (2009). Inspired from the unrestricted CRP that allows arbitrary relocations, the authors introduced an intermediary variant called the *relaxed restricted CRP*. In the relaxed restricted CRP, the entire retrieval process is divided into C time periods corresponding to C retrievals. In each time period $n \in \{1, \dots, C\}$ lasting from time n to time $n + 1$, the task is to retrieve the target container n. Every container is allowed to be relocated at most once in each time period, including the ones placed below the target container, if any. The authors then extended the CRP-I model for formulating the relaxed restricted CRP. For the ease of description, we refer to the formulation as CRP-Ir ("r" for "relaxed") hereinafter.

In this paper, it is first shown that the CRP-Ir model contains two deficiencies in formulating the "last in, first out" (LIFO) policy. Then, the LIFO policy is analyzed in detail and reformulated as linear constraints with new variables. Lastly, the corrected integer programming formulation named CRP-Irc ("c" for "corrected") is presented. The rest of this paper is organized as follows. Section 2 recalls the formulation of CRP-Ir and Section 3 discusses its deficiencies. Section 4 reformulates the LIFO policy with new integer variables and presents the corrected integer programming formulation named CRP-Irc. Section 5 reports the computational results of the CRP-Ir and CRP-Irc models. Finally, Section 6 concludes the paper.

⋆This work is partially supported by the project "PCL Future Regional Network Facilities for Large-Scale Experiments and Applications" (PCL2018KP001).

Email address: `jinb@sustech.edu.cn` (Bo Jin)

2. Formulation of CRP-Ir

The CRP-Ir model considers only the first $C - S$ time periods, as the remaining problem becomes "trivial" when there are only S containers left. The optimal solution to the remaining retrieval process is equal to the number of blocking containers in the residual configuration at time N. Note that container d is said to be *blocking* if there exists a container c in a lower tier of the same stack such that $c < d$.

Defining $N = C - S + 1$, the CRP-Ir model uses three sets of binary decision variables:

- *Support relation variables:* For $n \in \{1, \dots, N\}$, $c \in \{n, \dots, C + S\}$, $d \in \{n, \dots, C\}$,
$$a_{n,c,d} = \begin{cases} 1, & \text{if container } c \text{ is placed below } d \text{ at time } n, \\ 0, & \text{otherwise;} \end{cases}$$
- *Blocking indicator variables:* For $d \in \{N + 1, \dots, C\}$,
$$b_d = \begin{cases} 1, & \text{if container } d \text{ is blocking at time } N, \\ 0, & \text{otherwise;} \end{cases}$$
- *Relocation indicator variables:* For $n \in \{1, \dots, N - 1\}$, $c \in \{n + 1, \dots, C\}$,
$$x_{n,c} = \begin{cases} 1, & \text{if container } c \text{ is relocated in time period } n, \\ 0, & \text{otherwise.} \end{cases}$$

Note that the ground of stack $s \in \{1, \dots, S\}$ is regarded as an artificial container identified by a number $C + s$, so that the second subscript index of a takes values $1, \dots, C + S$. Given the binary encoding A of the initial configuration at time 1, the formulation of CRP-Ir is as follows:

$$\min \sum_{n=1}^{N-1} \sum_{c=n+1}^{C} x_{n,c} + \sum_{d=N+1}^{C} b_d \qquad \text{(CRP-Ir)}$$

s.t.

$$a_{1,c,d} = A_{c,d}, \qquad c \in \{1, \dots, C + S\},\ d \in \{1, \dots, C\}; \quad (1)$$

$$\sum_{s=1}^{S} a_{n,C+s,c} = 1, \qquad n \in \{2, \dots, N\},\ c \in \{n, \dots, C\}; \quad (2)$$

$$a_{n,c,c} = 0, \qquad n \in \{2, \dots, N\},\ c \in \{n, \dots, C\}; \quad (3)$$

$$a_{n,c,d} + a_{n,d,c} \le 1, \qquad n \in \{2, \dots, N\},\ c \in \{n, \dots, C\},\ d \in \{n, \dots, C\} \setminus \{c\}; \quad (4)$$

$$a_{n,c,d} + a_{n,d,c} \ge a_{n,C+s,c} + a_{n,C+s,d} - 1, \qquad n \in \{2, \dots, N\},\ s \in \{1, \dots, S\},\ c \in \{n, \dots, C\},\ d \in \{n, \dots, C\} \setminus \{c\}; \quad (5)$$

$$a_{n,c,d} + a_{n,d,c} \le 2 - a_{n,C+s,c} - \sum_{\substack{r=1 \\ r \ne s}}^{S} a_{n,C+r,d}, \qquad n \in \{2, \dots, N\},\ s \in \{1, \dots, S\},\ c \in \{n, \dots, C\},\ d \in \{n, \dots, C\} \setminus \{c\}; \quad (6)$$

$$\sum_{d=n}^{C} a_{n,C+s,d} \le T, \qquad n \in \{2, \dots, N\},\ s \in \{1, \dots, S\}; \quad (7)$$

$$b_d \ge a_{N,c,d}, \qquad c \in \{N, \dots, C - 1\},\ d \in \{c + 1, \dots, C\}; \quad (8)$$

$$a_{n+1,d,c} \le a_{n,d,c} + x_{n,c}, \qquad n \in \{1, \dots, N - 1\},\ c \in \{n + 1, \dots, C\},\ d \in \{n + 1, \dots, C + S\} \setminus \{c\}; \quad (9)$$

$$a_{n+1,d,c} \ge a_{n,d,c} - x_{n,c}, \qquad n \in \{1, \dots, N - 1\},\ c \in \{n + 1, \dots, C\},\ d \in \{n + 1, \dots, C + S\} \setminus \{c\}; \quad (10)$$

$$x_{n,c} + a_{n,C+s,c} + a_{n+1,C+s,c} \le 2, \qquad n \in \{1, \dots, N - 1\},\ s \in \{1, \dots, S\},\ c \in \{n + 1, \dots, C\}; \quad (11)$$

$$x_{n,c} + x_{n,d} + a_{n,c,d} + a_{n+1,c,d} \le 3, \qquad n \in \{1, \dots, N - 1\},\ c \in \{n + 1, \dots, C\},\ d \in \{n + 1, \dots, C\} \setminus \{c\}; \quad (12)$$

$$x_{n,c} \ge a_{n,n,c}, \qquad n \in \{1, \dots, N - 1\},\ c \in \{n + 1, \dots, C\}; \quad (13)$$

$$a_{n,c,d} + a_{n,e,f} + a_{n+1,e,d} + a_{n+1,c,f} \le 3 + a_{n,e,d}, \qquad n \in \{1, \dots, N - 1\},\ c, e \in \{n + 1, \dots, C + S\},\ d, f \in \{n + 1, \dots, C\},\ c \ne d \ne e \ne f; \quad (14)$$

$$a_{n,c,d} \in \{0, 1\}, \qquad n \in \{1, \dots, N\},\ c \in \{n, \dots, C + S\},\ d \in \{n, \dots, C\}; \quad (15)$$

$$b_d \in \{0, 1\}, \qquad d \in \{N + 1, \dots, C\}; \quad (16)$$

$$x_{n,c} \in \{0, 1\}, \qquad n \in \{1, \dots, N - 1\},\ c \in \{n + 1, \dots, C\}. \quad (17)$$

The objective function of CRP-Ir minimizes the total number of relocations. Constraint (1) initializes the variables a for $n = 1$ with the input binary encoding A. Constraint (2) ensures that a container is in one stack and Constraint (3) ensures that a container cannot support itself. Constraints (4)–(6) together enforce the support relations between containers. Constraint (7) imposes the height limitation of stacks. Constraint (8) enforces the relation between variables a and b. Constraints (9)–(11) enforce the relation between variables a and x. Constraint (12) ensures the LIFO policy for two containers relocated from the same source stack to the same destination stack. Constraint (13) enforces that containers placed above the target container must be relocated. Constraint (14) ensures the LIFO policy for containers from two different stacks. The number of variables is of the order of C^3 and the number of constraints is of the order of C^5.

3. Deficiencies of CRP-Ir

In this section, we show that the CRP-Ir model contains two deficiencies in formulating the LIFO policy. The first deficiency is related to *support relation cycles* of containers, which result in infeasible configurations. The second deficiency is related to *precedence relation cycles* of relocations, which make it impossible to construct a feasible relocation sequence.

3.1. Support relation cycles

Support relations between containers should be transitive, that is, $a_{n,c,d} = 1$ and $a_{n,d,e} = 1$ implies $a_{n,c,e} = 1$. However, Constraint (12) only ensures the LIFO policy for containers relocated from the same source stack to the same destination stack. If multiple containers are relocated from different source stacks to the same destination stack, the resultant support relations may form a cycle, violating the transitive property and leading to an infeasible configuration.

Figure 1 gives an example of a support relation cycle. Assuming that containers c, d, e are relocated to the same destination stack in a certain time period n, the resultant support relations $a_{n+1,c,d} = a_{n+1,d,e} = a_{n+1,e,c} = 1$ form a cycle, while Constraint (12) is still satisfied.

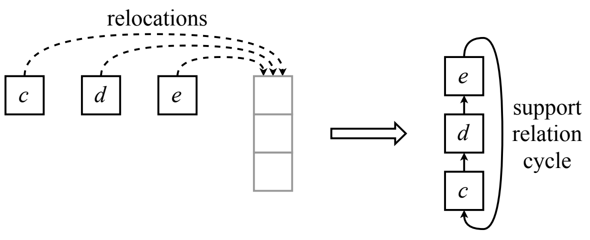

Figure 1: Support relation cycle.

We would like to remark that this deficiency only leads to infeasible configurations, but does not affect the quality of the optimal objective value obtained by the CRP-Ir model. The statement is formally presented in Claim 1 and the proof will be given in Appendix A.

Claim 1. *Any optimal solution with support relation cycles obtained by the CRP-Ir model can be transformed to another optimal solution without support relation cycles with the same objective value.*

Nevertheless, an optimal solution without support relation cycles of the CRP-Ir model is not necessarily an optimal solution to the relaxed restricted CRP, due to the existence of precedence relation cycles.

3.2. Precedence relation cycles

Constraint (14) only ensures the LIFO policy for containers from two different stacks but is insufficient for the case when more stacks are involved. Figure 2 gives an example, where the support relations are $a_{n,c,d} = a_{n,e,f} = a_{n,g,h} = 1$ before the relocations and $a_{n+1,c,h} = a_{n+1,e,d} = a_{n+1,g,f} = 1$ after the relocations. It is easy to notice that container d has to be relocated before the relocation of container h, or "d precedes h". Similarly, h precedes f, and f precedes d. Such precedence relation cycle violates the LIFO policy but still satisfies Constraint (14).

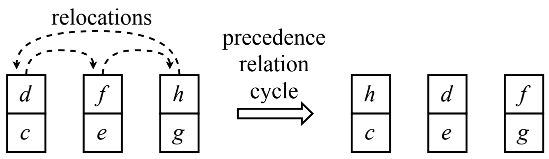

Figure 2: Precedence relocation cycle.

Next, we use a simple instance to illustrate that CRP-Ir may report an incorrect solution with smaller objective value than the true optimum. Given the initial configuration of size $(S, T, C) = (3, 4, 9)$, Figure 3 gives the optimal solution obtained by CRP-Ir with objective value 7. In time period 1, containers 2, 3, 9 are relocated from stack 1 to stack 2, containers 6, 7 are relocated from stack 2 to stack 3, and containers 4, 5 are relocated from stack 3 to stack 1, resulting in a precedence relation cycle. As a comparison, Figure 4 shows one of the true optimal solutions, where the correct optimal objective value should be 8.

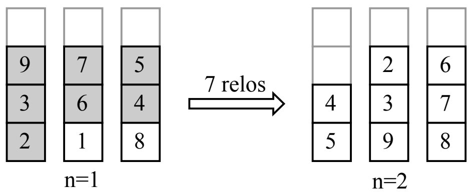

Figure 3: Incorrect solution reported by the CRP-Ir model.

3.3. Verification of CRP-Ir solutions

Next, we present a three-step algorithm for verifying the solution obtained by the CRP-Ir model. The overall process of the algorithm is as follows:

Step 1. Detect support relation cycles in the solution. If no support relation cycle is detected, i.e., all the intermediate configurations are feasible, go to Step 2. Otherwise, the algorithm terminates;

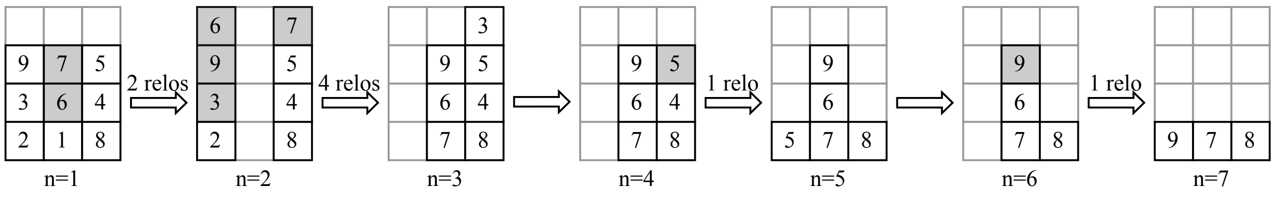

Figure 4: Correct optimal solution for the relaxed restricted CRP.

Step 2. Detect precedence relation cycles in the solution. If no precedence relation cycle is detected, i.e., feasible relocation subsequences between intermediate configurations can be constructed, go to Step 3. Otherwise, the algorithm terminates;

Step 3. Finally, the algorithm completes the relocation sequence and terminates with success.

At the first step, we verify the feasibility of the intermediate configurations as follows:

Procedure 1 (Detection of support relation cycles)**.** For $n \in \{1, \ldots, N\}$, we build a support relation graph $G_n^{\perp} = (V_n^{\perp}, E_n^{\perp})$ with respect to the configuration at time n, where $V_n^{\perp} = \{n, \ldots, C\}$ is the set of vertices representing containers and $E_n^{\perp} = \{(c, d) \in V_n^{\perp} \times V_n^{\perp} \mid a_{n,c,d} = 1\}$ is the set of edges representing support relations between containers. We apply Tarjan's algorithm to identify the strongly connected components of $G_n^{\perp}$. The presence of non-trivial strongly connected components (which contain more than one vertex) implies the presence of support relation cycles. In other words, if the strongly connected components of $G_n^{\perp}$ are all trivial, the corresponding configuration at time n is said to be feasible.

If no support relation cycle exists in the solution, we carry out the second procedure to sort the relocations according to the precedence relations. The precedence relations between relocations depend on how two relocations are related to the same stack.

We first classify the relocations related to a certain stack s in a certain time period n into two types:

- *Outgoing relocation:* If container c is relocated from stack s, the relocation of container c is an outgoing relocation of stack s;
- *Incoming relocation:* If container c is relocated to stack s, the relocation of container c is an incoming relocation of stack s.

Regarding all the outgoing and incoming relocations related to stack s in time period n, the precedence relations are established by the following rules:

- *Outgoing precedence rule:* For containers relocated from the same stack, the higher ones should be relocated before the lower ones;
- *Incoming precedence rule:* For containers relocated to the same stack, the containers to the lower tiers should be relocated before those to the higher tiers;
- *Replacing precedence rule:* For a certain stack, outgoing relocations should be performed before the incoming relocations.

The procedure of detecting precedence relation cycles is presented as follows:

Procedure 2 (Detection of precedence relation cycles)**.** For $n \in \{1, \ldots, N-1\}$, we build a precedence relation graph $G_n^{\prec} = (V_n^{\prec}, E_n^{\prec})$ with respect to the relocations conducted in time period n, where $V_n^{\prec} = \{c = n+1, \ldots, C \mid x_{n,c} = 1\}$ is the set of vertices representing relocated containers and $E_n^{\prec}$ is the set of edges representing precedence relations between relocations such that

$$E_n^{\prec} = \left\{ (c,d) \in V_n^{\prec} \times V_n^{\prec} \;\middle|\; \begin{array}{rl} s_{n,c} = s_{n,d} & \text{and} \quad t_{n,c} > t_{n,d} \\ \text{or } s_{n+1,c} = s_{n+1,d} & \text{and } t_{n+1,c} < t_{n+1,d} \\ \text{or} \quad s_{n,c} = s_{n+1,d} & \end{array} \right\}$$

where $s_{n,c} = \sum_{s=1}^{S} s \cdot a_{n,C+s,c}$ and $t_{n,c} = \sum_{d=n}^{C+S} a_{n,d,c}$ are the position (stack and tier) at which container c is located at time n. Note that a relocated container $c \in V_n^{\prec}$ corresponds to the relocation from $(s_{n,c}, t_{n,c})$ to $(s_{n+1,c}, t_{n+1,c})$. We apply Tarjan's algorithm to identify the strongly connected components of $G_n^{\prec}$. If there exists any non-trivial strongly connected component, then precedence relation cycles must exist. Otherwise, any topological ordering of $V_n^{\prec}$ is a feasible relocation subsequence for time period n.

The last step completes the relocation sequence, if neither support relation cycles nor precedence relation cycles are detected in previous steps.

Procedure 3 (Completion of the relocation sequence)**.** The problem of handling the residual configuration encoded by $a_{N,c,d}$ is trivial and can be solved optimally by relocating blocking containers to any empty stacks. By concatenating all the relocation subsequences obtained above, we get the complete relocation sequence.

4. CRP-Irc: The corrected model

4.1. Reformulating the LIFO policy

In order to correctly formulate the LIFO policy as linear constraints, new auxiliary variables are introduced:

- *Precedence relation variables:* For $n \in \{1, \ldots, N-1\}$, $c \in \{n+1, \ldots, C\}$, $d \in \{n+1, \ldots, C\} \setminus \{c\}$,

$$y_{n,c,d} = \begin{cases} 1, & \text{if container } c \text{ has to be relocated before } d \\ & \text{in time period } n, \\ 0, & \text{otherwise;} \end{cases}$$

- *Relocation order variables:* For $n \in \{1,\dots,N-1\}$, $c \in \{n+1,\dots,C\}$,

  $z_{n,c} \in \{1,\dots,C-n\}$, representing the relative order of relocating container c in time period n.

Incorporating the new variables y and z, we reformulate the LIFO policy as follows:

$$y_{n,c,d} \geq a_{n,d,c} + x_{n,c} + x_{n,d} - 2, \quad n \in \{1,\dots,N-1\}, \; c \in \{n+1,\dots,C\}, \; d \in \{n+1,\dots,C\} \setminus \{c\}; \tag{18}$$

$$y_{n,c,d} \geq a_{n+1,c,d} + x_{n,c} + x_{n,d} - 2, \quad n \in \{1,\dots,N-1\}, \; c \in \{n+1,\dots,C\}, \; d \in \{n+1,\dots,C\} \setminus \{c\}; \tag{19}$$

$$y_{n,c,d} \geq a_{n,C+s,c} + a_{n+1,C+s,d} + x_{n,c} + x_{n,d} - 3, \quad n \in \{1,\dots,N-1\}, \; s \in \{1,\dots,S\}, \; c \in \{n+1,\dots,C\}, \; d \in \{n+1,\dots,C\} \setminus \{c\}; \tag{20}$$

$$z_{n,c} + 1 \leq z_{n,d} + (C-n) \cdot (1 - y_{n,c,d}), \quad n \in \{1,\dots,N-1\}, \; c \in \{n+1,\dots,C\}, \; d \in \{n+1,\dots,C\} \setminus \{c\}. \tag{21}$$

These constraints are explained as follows:

- Constraint (18) ensures the outgoing precedence rule. If container c is placed above container d at time n and they are both relocated in time period n, then $y_{n,c,d}$ is forced to take value 1;
- Constraint (19) ensures the incoming precedence rule. If containers c and d are both relocated in time period n, and c will be placed below d at time $n+1$, then $y_{n,c,d}$ is forced to take value 1;
- Constraint (20) ensures the replacing precedence rule. If containers c is relocated from stack s and container d is relocated to stack s in time period n, then $y_{n,c,d}$ is forced to take value 1;
- Constraint (21) enforces the relation between variables y and z. If container c is forced to be relocated before container d, then the relocation order of d must be greater than that of c. Specifically, if $y_{n,c,d} = 1$, $z_{n,c} + 1 \leq z_{n,d}$ is enforced; otherwise, the inequality $z_{n,c} + 1 \leq z_{n,d} + C - n$ always holds for any values of $z_{n,c}$ and $z_{n,d}$.

4.2. *Formulation of CRP-Irc*

First of all, we simplify Constraints (4)–(6) of CRP-Ir as follows:

- Since indices c and d are symmetric in Constraints (4)–(6), their ranges can be narrowed down to $c \in \{n,\dots,C-1\}$, $d \in \{c+1,\dots,C\}$ to remove redundancy;
- Constraint (6) can be rewritten in a simpler form $a_{n,c,d} + a_{n,d,c} \leq a_{n,C+s,c} - a_{n,C+s,d} + 1$.

Furthermore, by replacing Constraints (12) and (14) by the new constraints (18)–(21), we correct the formulation of CRP-Ir and present the new formulation named CRP-Irc as follows:

$$\min \sum_{n=1}^{N-1} \sum_{c=n+1}^{C} x_{n,c} + \sum_{d=N+1}^{C} b_d \tag{CRP-Irc}$$

s.t. (1)–(3), (7)–(11), (13), (15)–(17), (18)–(21), and

$$a_{n,c,d} + a_{n,d,c} \leq 1, \quad n \in \{2,\dots,N\}, \; c \in \{n,\dots,C-1\}, \; d \in \{c+1,\dots,C\}; \tag{4'}$$

$$a_{n,c,d} + a_{n,d,c} \geq a_{n,C+s,c} + a_{n,C+s,d} - 1, \quad n \in \{2,\dots,N\}, \; s \in \{1,\dots,S\}, \; c \in \{n,\dots,C-1\}, \; d \in \{c+1,\dots,C\}; \tag{5'}$$

$$a_{n,c,d} + a_{n,d,c} \leq a_{n,C+s,c} - a_{n,C+s,d} + 1, \quad n \in \{2,\dots,N\}, \; s \in \{1,\dots,S\}, \; c \in \{n,\dots,C-1\}, \; d \in \{c+1,\dots,C\}; \tag{6'}$$

$$y_{n,c,d} \in \{0,1\}, \quad n \in \{1,\dots,N-1\}, \; c \in \{n+1,\dots,C\}, \; d \in \{n+1,\dots,C\} \setminus \{c\}; \tag{22}$$

$$z_{n,c} \in \{1,\dots,C-n\}, \quad n \in \{1,\dots,N-1\}, \; c \in \{n+1,\dots,C\}. \tag{23}$$

The number of variables is of the order of C^3 and the number of constraints is of the order of $C^3 S$. It is evident that the number of constraints is decreased dramatically in the CRP-Irc model compared to the CRP-Ir model. As a result, the computational efficiency has a remarkable improvement accordingly. Relevant experimental results will be demonstrated in detail in Section 5.

Since the new constraints already prevent support relation cycles and precedence relation cycles, the construction of complete relocation sequence for a CRP-Irc solution is quite simple as follows:

Step 1. First of all, define $s_{n,c} = \sum_{s=1}^{S} s \cdot a_{n,C+s,c}$ and $t_{n,c} = \sum_{d=n}^{C+S} a_{n,d,c}$ for $n \in \{1,\dots,N\}$ and $c \in \{n+1,\dots,C\}$, so that every relocated container c in time period n corresponds to the relocation from $(s_{n,c}, t_{n,c})$ to $(s_{n+1,c}, t_{n+1,c})$;

Step 2. For each time period $n \in \{1,\dots,N-1\}$, sort the relocated containers $\{c = n+1,\dots,C \mid x_{n,c} = 1\}$ in the ascending order of $z_{n,c}$;

Step 3. Then, finish the remaining retrieval process by relocating blocking containers to any empty stacks;

Step 4. Finally, construct the complete relocation sequence by concatenating all the relocation subsequences described above.

5. Computational experiments

5.1. Experiment setting

We test and compare the two integer programming models, CRP-Ir and CRP-Irc, on the instances introduced by Caserta et al. (2012). These instances are classified by two values (T', S), where T' is the number of containers per stack and S is the number of stacks in the initial configuration. Each class consists of 40 instances, and each of them contains a different initial configuration with $C = T' \times S$ containers. Since the height limitation is not indicated in the original dataset, we adopt the commonly used setting $T = T' + 2$ from the literature.

The two models are solved by IBM ILOG CPLEX 12.8.0 on a server running CentOS 7.6 with two Intel Xeon E5-2640 v3 CPUs and 64 gigabytes of RAM. The programs are written in Scala 2.12.8 and run with Java 8. In keeping with the experiments of Galle et al. (2018), we employ the same enhancement and limitation as follows. Both models are enhanced by an absolute MIP gap of 0.99 and an MIP start obtained by the heuristic proposed by Caserta et al. (2012). A time limit of 3600 seconds is used per instance for the solver. A model size limit of 8.0×10^{10} is imposed on the product of the number of variables and the number of constraints.

5.2. Comparison of model sizes

As mentioned in Section 2, the number of constraints in the CRP-Ir model is of the order of C^5. The vast majority is from Constraint (14), which has been shown to be insufficient for ensuring the LIFO policy. As a comparison, the number of constraints in the CRP-Irc model is of the order of C^3S.

Table 1 compares the model sizes between CRP-Ir and CRP-Irc, expressed by the number of variables and the number of constraints. The cells marked with a light gray background are the instance classes where the model size limit is exceeded. We notice that the number of variables of CRP-Irc slightly increases to about $1.5 \sim 1.8$ times that of CRP-Ir due to the introduction of new variables y and z (see column "$\gamma_{\#\text{vars}}$"). On the other hand, the number of constraints of CRP-Irc greatly decreases to about $4.0/CT'$ times that of CRP-Ir (see column "$\gamma_{\#\text{cons}} \times CT'$").

5.3. Comparison of performance

Table 2 reports the computational results of CRP-Ir and CRP-Irc on the 21 instance classes. The cells marked with a light gray background are the instance classes where the model size limit is exceeded. By examining the solutions obtained by the CRP-Ir model on the instances of Caserta et al. (2012), we did not find incorrect solutions that have smaller objective value than the true optimum. Nevertheless, we can still compare the computational efficiency between the two integer programming models.

The columns under the head "CRP-Ir" show the number of instances that are solved to optimality and the average time used in seconds, for each instance class. Note that for the instances solved optimally by CRP-Ir, CRP-Irc can also find the optimal solutions within the same time limit. For a more specific comparison, the next columns under the head "CRP-Irc (selected)" give the average time used by CRP-Irc on the same instances solved optimally by CRP-Ir. From the comparison, we see that CRP-Irc is significantly more efficient than CRP-Ir for most instance classes. The last columns under the head

Table 1: Comparison of model sizes between CRP-Ir and CRP-Irc.

Instances	CRP-Ir			CRP-Irc			$\gamma_{\#\text{vars}}$	$\gamma_{\#\text{cons}} \times CT'$
(T', S)	#vars	#cons	#vars × #cons	#vars	#cons	#vars × #cons		
(3, 3)	441	10,741	4.7×10^6	640	2,434	1.6×10^6	1.45	6.12
(3, 4)	987	54,218	5.4×10^7	1,479	6,962	1.0×10^7	1.50	4.62
(3, 5)	1,859	185,978	3.5×10^8	2,844	15,865	4.5×10^7	1.53	3.84
(3, 6)	3,133	501,377	1.6×10^9	4,863	31,277	1.5×10^8	1.55	3.37
(3, 7)	4,885	1,148,315	5.6×10^9	7,664	55,748	4.3×10^8	1.57	3.06
(3, 8)	7,191	2,338,884	1.7×10^{10}	11,375	92,244	1.0×10^9	1.58	2.84
(4, 4)	2,119	220,916	4.7×10^8	3,345	17,096	5.7×10^7	1.58	4.95
(4, 5)	4,024	743,137	3.0×10^9	6,464	38,855	2.5×10^8	1.61	4.18
(4, 6)	6,821	1,974,636	1.3×10^{10}	11,090	76,464	8.5×10^8	1.63	3.72
(4, 7)	10,681	4,473,812	4.8×10^{10}	17,520	136,115	2.4×10^9	1.64	3.41
(5, 4)	3,859	662,446	2.6×10^9	6,315	33,982	2.1×10^8	1.64	3.28
(5, 5)	7,364	2,195,596	1.6×10^{10}	12,234	77,070	9.4×10^8	1.66	4.39
(5, 6)	12,525	5,773,819	7.2×10^{10}	21,025	151,459	3.2×10^9	1.68	3.93
(5, 7)	19,662	12,982,627	2.6×10^{11}	33,256	269,353	9.0×10^9	1.69	3.63
(5, 8)	29,095	26,073,324	7.6×10^{11}	49,495	444,940	2.2×10^{10}	1.70	3.41
(5, 9)	41,144	48,080,382	2.0×10^{12}	70,310	694,392	4.9×10^{10}	1.71	3.25
(5, 10)	56,129	82,938,817	4.7×10^{12}	96,269	1,035,865	1.0×10^{11}	1.72	3.12
(6, 6)	20,677	13,957,406	2.9×10^{11}	35,532	263,822	9.4×10^9	1.72	4.08
(6, 10)	93,109	197,397,710	1.8×10^{13}	163,034	1,800,770	2.9×10^{11}	1.75	3.28
(10, 6)	86,405	169,372,338	1.5×10^{13}	156,560	1,238,154	1.9×10^{11}	1.81	4.39
(10, 10)	393,029	2,314,430,282	9.1×10^{14}	721,094	8,414,390	6.1×10^{12}	1.83	3.64

Table 2: Comparison of performance between CRP-Ir and CRP-Irc.

Instances	CRP-Ir		CRP-Irc (selected)		CRP-Irc	
(T', S)	#opt	time (s)	time (s)	speedup	#opt	time (s)
$(3, 3)$	40	0.3	0.2	1.6	40	**0.2**
$(3, 4)$	40	13.5	1.0	13.7	40	**1.0**
$(3, 5)$	38	115.7	2.4	49.1	**40**	4.3
$(3, 6)$	37	202.6	4.1	49.1	**40**	116.1
$(3, 7)$	30	87.7	3.4	25.9	**39**	216.1
$(3, 8)$	24	164.9	7.1	23.3	**34**	253.9
$(4, 4)$	20	525.5	10.1	52.1	**40**	81.7
$(4, 5)$	10	259.5	7.8	33.3	**30**	457.5
$(4, 6)$	11	431.6	10.8	40.1	**18**	526.4
$(4, 7)$	7	859.8	29.1	29.5	**10**	657.9
$(5, 4)$	4	181.7	8.5	21.3	**15**	746.2
$(5, 5)$	0	–	–	–	**1**	1522.4
$(5, 6)$	1	342.0	21.3	16.1	1	**21.3**
$(5, 7)$					**1**	70.5
$(5, 8)$					0	–
$(5, 9)$					0	–
$(5, 10)$	model size limit exceeded					
$(6, 6)$					0	–
$(6, 10)$						
$(10, 6)$						
$(10, 10)$						

"CRP-Irc" demonstrate the performance of the CRP-Irc model on all the 40 instances for each instance class. Not only in the time used, but also in the number of optimal solutions found, the CRP-Irc model outperforms the CRP-Ir model on all the instance classes. Due to the complexity of the problem, the CRP-Irc model finds very few optimal solutions on instance classes with size larger than $(5, 4)$ within the time limit.

5.4. Gaps to the unrestricted CRP

Galle et al. (2018) mentioned that by comparing the results of CRP-Ir with Petering & Hussein (2013), the optimal solutions of the relaxed restricted CRP were also optimal for the unrestricted CRP on small instances. To further demonstrate this statement, we extend the comparison to more instances in Table 3. Columns under the head "CRP-Irc" show the number of instances solved to optimality by the CRP-Irc model and the total number of relocations for the relaxed restricted CRP. Columns "IDA*-R (selected)" and "IDA*-U (selected)" give the total number of relocations on the same instances, solved optimally by the iterative deepening A* (IDA*) algorithms of Zhu et al. (2012) for the restricted and unrestricted variants, respectively.

Table 3: Comparison of optimal solutions for three CRP variants.

Instances	CRP-Irc		IDA*-R	IDA*-U
(T', S)	#opt	#relos	(selected)	(selected)
$(3, 3)$	40	**199**	200	199
$(3, 4)$	40	**241**	247	241
$(3, 5)$	40	**274**	281	274
$(3, 6)$	40	**331**	336	331
$(3, 7)$	39	**351**	357	351
$(3, 8)$	34	**340**	350	340
$(4, 4)$	40	391	408	389
$(4, 5)$	30	**344**	365	344
$(4, 6)$	18	**211**	220	211
$(4, 7)$	10	**134**	136	134
$(5, 4)$	15	189	195	188

From the comparison results, we notice that the relaxed restricted CRP has a significant improvement from the restricted variant towards the unrestricted variant. On most small and medium instances, the optimal solutions obtained by CRP-Irc are also the optimal solutions for the unrestricted CRP (see the numbers in bold). Next, we give an example in Figure 5 that the optimal solution of the relaxed restricted CRP is not optimal for the unrestricted CRP. In this example, container 13 is relocated twice in time period 1, and the optimal solution of the unrestricted CRP is 9. However, the optimal solution of the relaxed restricted CRP for this instance is 10.

6. Conclusion

This paper deals with a recently introduced variant of the CRP, called the relaxed restricted CRP, where every container can be relocated at most once for retrieving each target container. The major contribution of this work is threefold: 1) two deficiencies of the CRP-Ir model are discussed, 2) the LIFO policy is analyzed in detail and reformulated correctly as linear constraints with new variables, and 3) the corrected integer programming formulation named CRP-Irc is presented. Future research questions include adapting the binary encoding of configurations as well as the correct formulation of the LIFO policy to related problems, such as the unrestricted CRP and the container pre-marshalling problem.

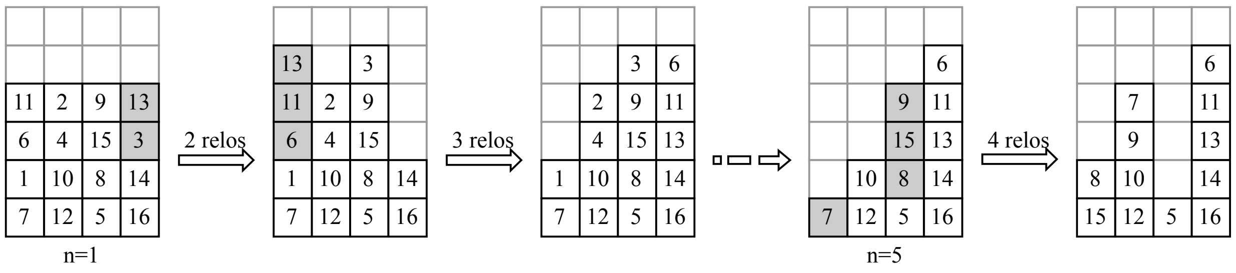

Figure 5: Optimal solution of the unrestricted CRP.

Appendix A. Proof of Claim 1

Suppose that we have identified the strongly connected components of the support relation graph $G_n^{\perp} = (V_n^{\perp}, E_n^{\perp})$ using Tarjan's algorithm for $n \in \{1, \dots, N\}$. Let $P_n^{\perp} = \{(c, d) \in V_n^{\perp} \times V_n^{\perp} \mid c \neq d\}$ denote the set of container pairs in $G_n^{\perp}$, and for $(c, d) \in P_n^{\perp}$, we define

$$\phi_{n,c,d} = \begin{cases} -1, & \text{if containers } c \text{ and } d \text{ are in different stacks at time } n, \\ 0, & \text{if containers } c \text{ and } d \text{ are in the same stack but not in the same strongly connected component of } G_n^{\perp}, \\ 1, & \text{if containers } c \text{ and } d \text{ are in the same strongly connected component of } G_n^{\perp}. \end{cases}$$

Lemma 1. *The shortest cycle in a non-trivial strongly connected component of any support relation graph is of length 3.*

Proof of Lemma 1. Since neither self loops nor parallel edges exist in any support relation graph due to Constraints (3) and (4), any non-trivial strongly connected component contains at least 3 vertices. Suppose that we have a cycle $[c_1, c_2, c_3, \dots, c_k]$ with $k > 3$ such that $a_{n,c_k,c_1} = 1$ and $a_{n,c_i,c_{i+1}} = 1$ for $i \in \{1, \dots, k-1\}$. Since $a_{n,c_1,c_3} + a_{n,c_3,c_1} = 1$, we can always detect a shorter cycle $[c_1, c_3, \dots, c_k]$ (if $a_{n,c_1,c_3} = 1$) or $[c_1, c_2, c_3]$ (if $a_{n,c_3,c_1} = 1$). □

Lemma 2. *Given any $n \in \{2, \dots, N\}$ and $(c, d) \in P_n^{\perp}$ such that $\phi_{n,c,d} = 1$, it must hold that $x_{n-1,c} = x_{n-1,d}$.*

Proof of Lemma 2. If container c is not relocated in time period $n-1$, i.e., $x_{n-1,c} = 0$, all containers placed below container c at time n are not relocated in time period $n-1$ either. By the transitivity of support relation cycles, $x_{n-1,d} = 0$ must hold. On the other hand, if container c is relocated in time period $n-1$, i.e., $x_{n-1,c} = 1$, all containers placed above container c at time n are also relocated in time period $n-1$. By the transitivity of support relation cycles, $x_{n-1,d} = 1$ must hold. □

Lemma 3. *Given any $n \in \{2, \dots, N-1\}$ and $(c, d) \in P_n^{\perp}$ such that $\phi_{n,c,d} = 1$, assuming $x_{n,n} = 1$, it must hold that $x_{n,c} = x_{n,d}$.*

Proof of Lemma 3. If container c is not relocated in time period n, i.e., $x_{n,c} = 0$, all containers placed below container c at time n are not relocated in time period n either. By the transitivity of support relation cycles, $x_{n,d} = 0$ must hold. On the other hand, if container c is relocated in time period n, i.e., $x_{n,c} = 1$, all containers placed above container c at time n are also relocated in time period n. By the transitivity of support relation cycles, $x_{n,d} = 1$ must hold. □

For $n \in \{2, \dots, N\}$ and $(c, d) \in P_n^{\perp}$ such that $\phi_{n,c,d} = 1$, let $\sigma_{n,c,d}$ be the smallest integer and $\tau_{n,c,d}$ be the greatest integer such that $2 \leq \sigma_{n,c,d} \leq n \leq \tau_{n,c,d} \leq N$ and $\phi_{v,c,d} = 1$ for all $v \in \{\sigma_{n,c,d}, \dots, \tau_{n,c,d}\}$.

Lemma 4. *For $n \in \{2, \dots, N\}$ and $(c, d) \in P_n^{\perp}$ such that $\phi_{n,c,d} = 1$, the following propositions hold:*

- *$x_{\sigma_{n,c,d}-1,c} = x_{\sigma_{n,c,d}-1,d} = 1$;*
- *Either $\tau_{n,c,d} = N$ or $x_{\tau_{n,c,d},c} = x_{\tau_{n,c,d},d} = 1$.*

Proof of Lemma 4. From the definition of $\sigma_{n,c,d}$, we have $\phi_{\sigma_{n,c,d}-1,c,d} \neq 1$. It implies that at least one of containers c and d is relocated in time period $\sigma_{n,c,d} - 1$. According to Lemma 2, we have $x_{\sigma_{n,c,d}-1,c} = x_{\sigma_{n,c,d}-1,d} = 1$. From the definition of $\tau_{n,c,d}$, we have either $\tau_{n,c,d} = N$ or $\phi_{\tau_{n,c,d}+1,c,d} \neq 1$. In the latter case, it implies that at least one of containers c and d is relocated in time period $\tau_{n,c,d}$. According to Lemma 3, we have $x_{\tau_{n,c,d},c} = x_{\tau_{n,c,d},d} = 1$. □

Lemma 5. *Given containers c, d, and e that form a support relation cycle at time $n \in \{2, \dots, N\}$, they also form a cycle at time v for all $v \in \{\sigma, \dots, \tau\}$, where $\sigma = \max\{\sigma_{n,c,d}, \sigma_{n,d,e}, \sigma_{n,e,c}\}$ and $\tau = \min\{\tau_{n,c,d}, \tau_{n,d,e}, \tau_{n,e,c}\}$.*

Proof of Lemma 5. Given containers c, d, and e that form a support relation cycle at time n, we have either $a_{n,c,d} = a_{n,d,e} = a_{n,e,c} = 1$ or $a_{n,e,d} = a_{n,d,c} = a_{n,c,e} = 1$. According to Lemma 2 and Constraint (12), the three containers form a support relation cycle at time v for all $v \in \{\sigma, \dots, n-1\}$. Similarly, according to Lemma 3 and Constraint (12), the three containers also form a support relation cycle at time v for all $v \in \{n+1, \dots, \tau\}$. □

In order to eliminate all the support relation cycles, we need to assign new values to the variables $a_{n,c,d}$ for $n \in \{2, \dots, N\}$ and $(c, d) \in P_n^{\perp}$ such that $\phi_{n,c,d} = 1$. According to Constraints (4), (8), and (12), some of the variables are enforced to take their original values as follows.

Corollary 1. *For $n \in \{2, \dots, N\}$ and $(c, d) \in P_n^{\perp}$ such that $\phi_{n,c,d} = 1$, $a_{n,c,d}$ and $a_{n,d,c}$ are enforced to take their original values if at least one of the following conditions is satisfied:*

- *$\phi_{n-1,c,d} = 0$;*
- *$\phi_{n-1,c,d} = 1$, and the values of $\bar{a}_{n-1,c,d}$ and $\bar{a}_{n-1,d,c}$ have been enforced;*
- *$n = N$ and $b_{\max\{c,d\}} = 0$;*
- *$n < N$ and $\phi_{n+1,c,d} = 0$; or*
- *$n < N$, $\phi_{n+1,c,d} = 1$, and the values of $\bar{a}_{n+1,c,d}$ and $\bar{a}_{n+1,d,c}$ have been enforced.*

With the definitions of $\sigma_{n,c,d}$ and $\tau_{n,c,d}$, Corollary 1 can be simplified as follows.

Corollary 2. *For $n \in \{2, \dots, N\}$ and $(c, d) \in P_n^{\perp}$ such that $\phi_{n,c,d} = 1$, $a_{n,c,d}$ and $a_{n,d,c}$ are enforced to take their original values if at least one of the following conditions is satisfied:*

(C1) $\phi_{\sigma_{n,c,d}-1,c,d} = 0$;
(C2) $\tau_{n,c,d} = N$ *and* $b_{\max\{c,d\}} = 0$; *or*
(C3) $\tau_{n,c,d} < N$ *and* $\phi_{\tau_{n,c,d}+1,c,d} = 0$.

Lemma 6. *Given containers c, d, and e that form a support relation cycle at time $n \in \{2, \dots, N\}$, Condition C1 in Corollary 2 will be satisfied at most once for (c, d), (d, e), and (e, c).*

Proof of Lemma 6. Without loss of generality, we assume that $\sigma_{n,c,d}$ is the maximum among $\sigma_{n,c,d}$, $\sigma_{n,d,e}$, and $\sigma_{n,e,c}$. Then, we have $\phi_{\sigma_{n,c,d}-1,c,d} \neq 1$ and $x_{\sigma_{n,c,d}-1,c} = x_{\sigma_{n,c,d}-1,d} = x_{\sigma_{n,c,d}-1,e} = 1$ according to Lemmas 4 and 5. There are the following situations:

- If containers c, d, and e are in the same stack at time $\sigma_{n,c,d} - 1$, they also form a cycle at time $\sigma_{n,c,d} - 1$ according to Constraint (12), and thus $\phi_{\sigma_{n,c,d}-1,c,d} = 1$, which contradicts the premise;
- If two of containers c, d, and e are in the same stack at time $\sigma_{n,c,d} - 1$ and the other one is in another stack, then Condition C1 will be satisfied at most once for (c, d), (d, e), and (e, c);
- If containers c, d, and e are in three different stacks at time $\sigma_{n,c,d} - 1$, then Condition C1 will not be satisfied for any of (c, d), (d, e), and (e, c).

Proved. □

Lemma 7. *Given containers c, d, and e that form a support relation cycle at time* $n \in \{2, \ldots, N\}$*, Conditions C2 and C3 in Corollary 2 will be satisfied at most once in total for* (c, d)*,* (d, e)*, and* (e, c)*.*

Proof of Lemma 7. First, let us first consider the case when $\tau_{n,c,d} = \tau_{n,d,e} = \tau_{n,e,c} = N$. Without loss of generality, we assume that $c > d > e$. There are the following situations:

- If $b_c = 0$, it implies that $a_{n,c,d} = 1$ and $a_{n,c,e} = 1$, which contradicts the premise that containers c, d, and e form a support relation cycle at time n;
- If $b_c = 1$ and $b_d = 0$, it implies $a_{n,d,e} = 1$, and thus Condition C2 will be satisfied for (d, e) only;
- If $b_c = 1$ and $b_d = 1$, Condition C2 will not be satisfied for any of (c, d), (d, e), and (e, c).

Next, we turn to the opposite case. Without loss of generality, we assume that $\tau_{n,c,d} < N$ is the minimum among $\tau_{n,c,d}$, $\tau_{n,d,e}$, and $\tau_{n,e,c}$. Then, we have $\phi_{\tau_{n,c,d}+1,c,d} \neq 1$ and $x_{\tau_{n,c,d},c} = x_{\tau_{n,c,d},d} = x_{\tau_{n,c,d},e} = 1$ according to Lemmas 4 and 5. There are the following situations:

- If containers c, d, and e are in the same stack at time $\tau_{n,c,d} + 1$, they also form a cycle at time $\tau_{n,c,d} + 1$ according to Constraint (12), and thus $\phi_{\tau_{n,c,d}+1,c,d} = \phi_{\tau_{n,c,d}+1,d,e} = \phi_{\tau_{n,c,d}+1,e,c} = 1$, which contradicts the premise;
- If two of containers c, d, and e are in the same stack at time $\tau_{n,c,d} + 1$ and the other one is in another stack, then Condition C3 will be satisfied at most once for (c, d), (d, e), and (e, c);
- If containers c, d, and e are in three different stacks at time $\tau_{n,c,d} + 1$, then Condition C3 will not be satisfied for any of (c, d), (d, e), and (e, c).

Proved. □

Lemma 8. *All the variables enforced to take their original values by Corollary 2 will not form support relation cycles by themselves.*

Proof of Lemma 8. For any containers c, d, and e that form a support relation cycle at time $n \in \{2, \ldots, N\}$, according to Lemmas 6 and 7, we conclude that Corollary 2 will enforce at most two of the variable pairs $(a_{n,c,d}, a_{n,d,c})$, $(a_{n,d,e}, a_{n,e,d})$, and $(a_{n,e,c}, a_{n,c,e})$. □

For the variables whose values are enforced by Corollary 2, we simply keep their original values. For the rest support relation variables that are not enforced by the equations, their values are reassigned arbitrarily as long as the transitive property is not violated. As a result, the support relation cycles are eliminated, while the objective value has not been changed.

Finally, we summarize the procedure of eliminating support relation cycles from a CRP-Ir solution as follows:

- For any support relation graph $G_n^{\perp}$, remove the existing edges (c, d) such that $\phi_{n,c,d} = 1$ and the values of $a_{n,c,d}$ and $a_{n,d,c}$ are not enforced by Corollary 2;
- Find a topological ordering of the vertices in the residual graph, and then insert new edges according to the obtained topological order;
- Repeat the steps until all the support relation graphs are processed.

References

Caserta, M., Schwarze, S., & Voß, S. (2009). A new binary description of the blocks relocation problem and benefits in a look ahead heuristic. In C. Cotta & P. Cowling (Eds.), *Evolutionary Computation in Combinatorial Optimization* (pp. 37–48). Berlin, Heidelberg: Springer Berlin Heidelberg.

Caserta, M., Schwarze, S., & Voß, S. (2012). A mathematical formulation and complexity considerations for the blocks relocation problem. *European Journal of Operational Research*, 219(1), 96–104.

Galle, V., Barnhart, C., & Jaillet, P. (2018). A new binary formulation of the restricted container relocation problem based on a binary encoding of configurations. *European Journal of Operational Research*, 267(2), 467–477.

Kim, K. H. & Hong, G.-P. (2006). A heuristic rule for relocating blocks. *Computers & Operations Research*, 33(4), 940–954.

Petering, M. E. H. & Hussein, M. I. (2013). A new mixed integer program and extended look-ahead heuristic algorithm for the block relocation problem. *European Journal of Operational Research*, 231(1), 120–130.

Zhu, W., Qin, H., Lim, A., & Zhang, H. (2012). Iterative deepening A* algorithms for the container relocation problem. *IEEE Transactions on Automation Science and Engineering*, 9(4), 710–722.