Optimization Model and Heuristic Approach for Blocks Retrieval Processes in Warehouses

Christopher Expósito-Izquierdo, Belén Melián-Batista, and **J. Marcos Moreno-Vega**
{cexposit, mbmelian, jmmoreno}@ull.es
Department of Statistics, Operative Research and Computation
University of La Laguna, Spain

Abstract

In this paper we introduce a planning problem termed as Q-Blocks Relocation Problem, which pursues to retrieve a subset of blocks located in a warehouse by minimizing the number of relocation movements. We formalize the problem by means of a Mixed Integer Linear Programming model. However, the high computational burden required by the model encourages us to develop a heuristic algorithm for tackling it. The rationale behind the proposed heuristic is both to retrieve the requested blocks as soon as possible while reducing the number of blocks placed above another one with a higher priority. The computational results indicate that the heuristic reports near-optimal solutions for realistic instances by short computational times, which makes it attractive to be applied by management systems.

Introduction

Over the last decades, the management of supply chains has gained increasing interest within the research community (Breiter et al. 2009). In this environment, the warehouses are facilities aimed at storing goods temporarily at all phases in the supply chain. Their relevance lies in their essential role as intermediate points in the flow of goods, from producers to customers (Melo, Nickel, and da Gama 2009).

The block stacking is the most widespread storage strategy used in warehouses when storing goods packed in robust load units (Gu, Goetschalckx, and McGinnis 2010). In this strategy, homogeneous blocks are piled up one on each other in stacks in such a way that they make up three-dimensional storage structures, that is, sets of bays arranged in parallel (Gudehus and Kotzab 2009). The number of tiers is usually established by the physical features of the warehouse, the load strength (crushability), and the stability of the stacks. The storage location of a block is therefore given by the bay, stack, and tier in which is placed (Park and Kim 2010).

Order-picking is known to be the most labor-intensive and time-consuming process in warehouse logistics (Tompkins et al. 2010)(Grosse, Glock, and Jaber 2013). An order-picking can be understood as the retrieval of one or several blocks from their storage locations in a warehouse (de Koster, Le-Duc, and Roodbergen 2007). In this context, the time involved when retrieving the blocks requested by the customers constitutes the main indicator of the competitiveness of a warehouse (Chan and Chan 2011). However, it is worth mentioning that the main shortcoming associated with the block stacking arises from the Last In First Out (LIFO) policy used to access the blocks. Consequently, the retrieval time is short whenever the requested blocks are currently placed at the top of the stacks. Furthermore, relocation movements must be performed with the goal of freeing up the target blocks when they are buried below other ones (Caserta, Schwarze, and Voβ 2011).

The block relocation movements give rise to low performance in warehouses. As a result, the labor costs are increased whereas the customer satisfaction is highly impaired (Dekker, Voogd, and Asperen 2007). In order to avoid this scenario, suitable planning strategies are required to minimize the number of relocation movements during a given time horizon (Choe et al. 2011). These planning strategies pursue to place the incoming blocks in storing locations in which they will be accessible for their future retrieval. In spite of the exhaustive analysis of the most likely order-pickings, the high level of unpredictability inherent in the supply chains precludes to guarantee a constant retrieval time of the blocks (Zhao and Goodchild 2010).

The previous discussion draws forth the need of efficient approaches to retrieve the relevant blocks associated with incoming order-pickings. With this goal in mind, in this work we introduce the Q-Blocks Relocation Problem (Q-BRP) as a generalization of the known Blocks Relocation Problem (Caserta, Schwarze, and Voβ 2012) with practical application in warehouses. The Q-BRP pursues to retrieve a subset of blocks from a bay by minimizing the number of relocation movements. We firstly formalize this problem by a mixed integer linear programming model. The computational burden required by the model when solving real scenarios has also encouraged us to develop a heuristic algorithm.

The remainder of this work is organized as follows. We firstly introduce the Q-BRP and review the related works from the literature. We then proceed with a mathematical formulation for the Q-BRP. Afterwards, we propose a heuristic approach for the Q-BRP. The computational experiments carried out are later discussed. We finish the work with the main concluding remarks and several directions for further research.

Q-Blocks Relocation Problem

The Q-Blocks Relocation Problem (Q-BRP) is a combinatorial optimization planning problem in which we are given a bay composed of W stacks, H tiers, and N blocks. Each block c has a given priority, $p(c)$, which indicates its order in the retrieval sequence. The stack in which the block c is currently placed is denoted by $s(c)$, whereas its tier is denoted by $t(c)$. The number of blocks in the stack s is denoted by $h(s)$, whereas $\max(s)$ and $\min(s)$ report the highest and the lowest priority of s, respectively. The goal of the Q-BRP is to determine the sequence of movements with minimum length that should be performed in order to retrieve a subset of K blocks from the bay, where $K \leq N$. The retrieval of blocks must be carried out according to their decreasing order of priorities, in such a way that, the block with the highest priority, 1, must be retrieved before the block with priority 2, the block 2 must be retrieved before the block 3, and so forth, until all the K blocks with the highest priorities are retrieved from the bay. Note that after finishing the retrieval process $M = N - K$ blocks are still placed into the bay. At each step, the block with the highest priority among those currently placed into the bay is denoted by c^*. The technical features of the handling equipment (*e.g.* gantry crane, forklift truck, etc.) used in the storage environment establish the following types of block operations to perform:

1. *Retrieval operation*. The block with the highest priority, c^*, currently placed at the top of some stack into the bay is retrieved from it.
2. *Relocation operation*. A subset of at most Q blocks placed at the top of some stack is moved from its current stack to another one. It should be noted that, if several blocks are relocated at once their relative order in the target stack has to be exactly the same as in the source one.

The Q-BRP introduced in this work is a generalization of the BRP in which only one block can be relocated at once. Consequently, this problem is $\mathcal{NP}$-hard by reduction to the BRP (Caserta, Schwarze, and Voβ 2012).

A solution for the Q-BRP is a sequence of tuples with the shape (a, b, q), where a is the source stack, b is the target stack, and $q \leq Q$ is the number of blocks to move at each step. In this context, the retrieval movements are represented as $(a, -, 1)$, which indicates that one block is retrieved from the top of the stack a. It is worth mentioning that, each feasible solution for the Q-BRP contains K retrieval operations and, therefore, the objective of the problem can be tackled as minimizing the number of relocation movements.

In order to illustrate the Q-BRP let us consider the example depicted in Figure 1. This example shows a bay with 6 stacks, 4 tiers, and 16 blocks with exclusive priorities that are going to be retrieved on the basis of the prescribed priority order. The priority of each block is represented through a number in it. A solution for the example at hand when setting $Q = 1$ is as follows: ((4, 3, 1)(4, 3, 1)(4, 6, 1)(4, -, 1)(2, 5, 1)(2, 4, 1)(2, -, 1)(1, -, 1)(6, 4, 1)(6, -, 1)(3, 2, 1)(3, -, 1)(1, -, 1)(2, -, 1)(5, -, 1)(6, 3, 1)(6, -, 1)(3, -, 1)(2, -, 1)(4, 6, 1)(4, -, 1)(6, -, 1)(5, -, 1)(3, 1, 1)(3, -, 1)(1, -, 1)). As can be checked, this solution is composed of 10 relocation movements. However, the number of relocation movements can be greatly reduced whenever the features of the available handling equipment allow to move several blocks at once. For instance, a solution when setting $Q = 2$ is as follows: ((4, 3, 2)(4, 6, 1)(4, -, 1)(2, 4, 2)(2, -, 1)(1, -, 1)(6, 2, 1)(6, -, 1)(3, -, 1)(1, -, 1)(3, -, 1)(4, -, 1)(6, 1, 1)(6, -, 1)(1, -, 1)(2, 6, 1)(2, -, 1)(4, -, 1)(6, -, 1)(5, -, 1)(3, 1, 1)(3, -, 1)(1, -, 1)), which is composed of 7 relocation movements. Finally, a solution when setting $Q = 3$ is as follows: ((4, 5, 3)(4, -, 1)(2, 3, 2)(2, -, 1)(1, -, 1)(6, -, 1)(5, -, 1)(1, -, 1)(5, -, 1)(3, -, 1)(6, 4, 1)(6, -, 1)(4, -, 1)(2, -, 1)(3, -, 1)(5, -, 1)(5, -, 1)(3, 1, 1)(3, -, 1)(1, -, 1)), which is composed of only 4 relocation movements. The reduction in the number of relocation movements stemming from handling several blocks at once encourages us to develop optimization techniques aimed at exploiting this potential.

A straightforward analysis of the aforementioned example draws forth the following classification of the blocks according to their priorities and the slots in which they are currently placed into the bay:

1. *Non-located block*. That block currently placed above another one with a higher priority. See those blocks with gray background in Figure 1. It should be noted that this type of blocks has to be relocated in a different stack of the bay in order to allow the retrieval of the block with higher priority placed below it. The set of non-located blocks in the stack s is defined as follows:

$$\Omega(s) = \{c \mid (s(c) = s) \wedge \exists\, c' : (s(c') = s) \wedge (t(c') < t(c)) \wedge (p(c') < p(c))\}. \quad (1)$$

   The set of non-located blocks into the bay is denoted as follows:

$$\Omega = \bigcup_{1 \leq s \leq S} \Omega(s). \quad (2)$$

2. *Well-located block*. That block not currently placed above any other with a higher priority. See those blocks with white background in Figure 1. This type of blocks does not require any relocation movement before its retrieval. The set of well-located blocks in the stack s is defined as follows:

$$\Upsilon(s) = \{c \mid (s(c) = s) \wedge (c \notin \Omega(s))\}. \quad (3)$$

In the following we present a method for computing an upper bound for the Q-BRP. This upper bound can be used to provide a preliminary solution for the Q-BRP or setting a planning horizon as illustrated in the next section.

Algorithm 1 is the pseudocode of the method. It goes over each block into the bay according to the decreasing order of priorities (lines 3 - 14). At each step, that block with the highest priority, c^*, is found (line 4). If c^* is placed at the top of its stack, it can be directly retrieved from the bay (line 13). Otherwise, while it is not freed up, the method identifies those blocks currently placed on it (line 6), denoted by $O(c^*)$ and formally defined as follows:

$$O(c^*) = \{c \mid (s(c) = s(c^*)) \wedge (t(c) > t(c^*))\}. \quad (4)$$

This set is composed of non-located blocks (Equation 2) and, therefore, they have to be relocated in a different stack.

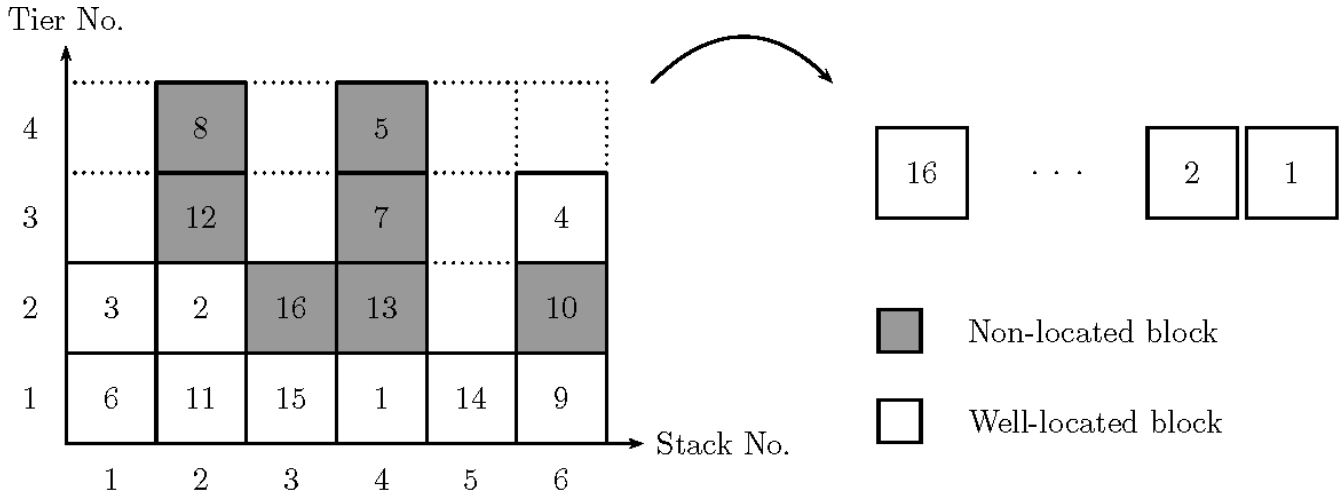

Figure 1: Example of a bay with 6 stacks, 4 tiers and 16 blocks.

With this goal in mind, the method pursues to define at this step a subset of blocks included into $O(c^*)$ to be relocated (line 8). According to the definition of the Q-BRP, the number of blocks that can be potentially relocated in each movement ranges from 1 up to Q. However, this range is restricted in order to fulfill those environments in which the number of blocks included into $O(c^*)$ is lower than Q (line 7). An example of this fact is presented when the block 1 is about to be retrieved from the bay and $Q > 3$. In spite of the fact that more than 3 blocks can be handled at once, at most 3 blocks can be moved in the next relocation movement. Once defined the blocks to relocate at each step, denoted by $O(c^*)'$, the method identifies all the feasible target stacks for them, denoted by Θ (line 9). A stack, s, could be used as target of $O(c^*)'$ if, and only if, the number of empty slots is at least equal to the cardinality of $O(c^*)'$. That is:

$$H - h(s) \geq |O(c^*)'|. \tag{5}$$

One stack, s', is selected at random from Θ (line 10) and $O(c^*)'$ is relocated in s' (line 11), by means of the movement $(s(c^*), s', |O(c^*)'|)$. Finally, in order to provide a tight upper bound the method is embedded into a restarting strategy (lines 2 - 16) which iterates max times, where max is a parameter whose value is set by the user.

Algorithm 1 Method for calculating the upper bound of the Q-BRP

Require: Bay with the blocks to retrieve
Require: max, number of iterations
Ensure: Solution of the Q-BRP
1: $BestSolution \leftarrow \emptyset$
2: **for** $i = 1 \rightarrow max$ **do**
3: **for** $p = 1 \rightarrow K$ **do**
4: $c^* \leftarrow$ Find block with priority p
5: **while** !(c^* is topmost) **do**
6: $O(c^*) \leftarrow$ Blocks placed on c^* (Equation 4)
7: $q' \leftarrow$ Select random number in $[1..\min\{Q, |O(c^*)|\}]$
8: $O(c^*)' \leftarrow$ Select q' topmost blocks from $O(c^*)$
9: $\Theta \leftarrow$ Feasible target stacks for $O(c^*)'$
10: $s' \leftarrow$ Select stack from Θ at random
11: Relocate $O(c^*)'$ from $s(c^*)$ to stack s'
12: **end while**
13: Retrieve c^* from the top of its stack
14: **end for**
15: Update $BestSolution$
16: **end for**
17: **return** $BestSolution$

Literature Review

The literature contains multitude works in which reducing the time required to fulfill the incoming order-pickings is addressed (Tompkins et al. 2010). Many authors, *e.g.* (Kang, Ryu, and Kim 2006), (van Zelst et al. 2009), (Rei and Pedroso 2012), and (Rei and Pedroso 2013), have propose different stacking policies aimed at storing blocks in warehouses in such a way that they are eventually retrieved efficiently. The works by (Lee and Hsu 2007), (Bortfeldt and Forster 2012), and (Expósito-Izquierdo, Melián-Batista, and Moreno-Vega 2012) focus on reshuffling the blocks stacked into a bay before their retrieval is started with the goal of piling they up according to their precedence relationships.

The BRP has been studied previously in several works, *e.g.* (Kim and Hong 2006), (Caserta, Voβ, and Sniedovich 2011), (Caserta, Schwarze, and Voβ 2009), (Zhang et al. 2010), and (Caserta, Schwarze, and Voβ 2012). A clear trend stemming from the related literature is to develop heuristic approaches aimed at solving the BRP. As indicated by (Caserta, Schwarze, and Voβ 2012), despite the effort carried out in developing exact techniques, these have not proved to be sufficiently efficient when solving real-world instances so far. Examples of this fact are both the Branch & Bound proposed by (Kim and Hong 2006) and the iterative deepening A* search algorithm designed by (Zhang et al. 2010).

Optimization Model

This section is devoted to model the Q-BRP by means of a Mixed Integer Linear Programming (MILP). Let us assume we have a planning horizon, T, obtained by some optimization technique (*e.g.* Algorithm 1) and, without loss of generality, that all the blocks placed in the bay are going to be retrieved, that is, $K = N$. In the following we firstly introduce the families of variables used by the model:

$$b_{ijnt} = \begin{cases} 1, & \text{if block } n \text{ is in } (i,j) \text{ in time period } t \\ 0, & \text{otherwise} \end{cases}$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T$$

$$x_{ijklqt} = \begin{cases} 1, & \text{if } q \text{ blocks are relocated from } (i,j) \\ & \text{in } (k,l) \text{ in time period } t \\ 0, & \text{otherwise} \end{cases}$$
$$\forall i,k = 1..W, j,l = 1..H, n = 1..N, t = 1..T$$

$$y_{ijnt} = \begin{cases} 1, & \text{if block } n \text{ is retrieved from } (i,j) \text{ in} \\ & \text{time period } t \\ 0, & \text{otherwise} \end{cases}$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T$$

$$v_{nt} = \begin{cases} 1, & \text{if block } n \text{ has been retrieved before} \\ & \text{time period } t \\ 0, & \text{otherwise} \end{cases}$$
$$\forall n = 1..N, t = 1..T$$

$$a_{ijnt} = \begin{cases} 1, & \text{if block } n \text{ is relocated from } (i,j) \text{ in} \\ & \text{time period } t \\ 0, & \text{otherwise} \end{cases}$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1$$

$$c_{ijnt} = \begin{cases} 1, & \text{if block } n \text{ is relocated in } (i,j) \text{ in time} \\ & \text{period } t \\ 0, & \text{otherwise} \end{cases}$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1$$

d_{ijnt}, number of blocks placed on the block n before its relocation from (i,j) in time period t,
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1$$

e_{ijnt}, number of blocks placed on the block n after its relocation in (i,j) in time period t,
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1$$

The objective function of the MILP is to maximize the periods that the last block to retrieve (N) is out of the bay:

$$maximize \sum_{t=1}^{T} v_{Nt} \tag{6}$$

Each block has to be in or out the bay:

$$\sum_{i=1}^{W}\sum_{j=1}^{H} b_{ijnt} + v_{nt} = 1, \forall n = 1..N, t = 1..T \tag{7}$$

In each slot has to be at most one block:

$$\sum_{n=1}^{N} b_{ijnt} \leq 1, \forall i = 1..W, j = 1..H, t = 1..T \tag{8}$$

The blocks have to be placed one on each other and, therefore, empty slots between them are not allowed:

$$\sum_{n=1}^{N} b_{ijnt} \geq \sum_{n=1}^{N} b_{ij+1nt},$$
$$\forall i = 1..W, j = 1..H-1, t = 1..T \tag{9}$$

At each time period at most one block operation is allowed:

$$\sum_{i=1}^{W}\sum_{j=1}^{H}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{\substack{q=1 \\ j+q-1\leq H \\ l+q-1\leq H}}^{Q} x_{ijklqt} +$$
$$\sum_{i=1}^{W}\sum_{j=1}^{H}\sum_{n=1}^{N} y_{ijnt} \leq 1, \forall t = 1..T \tag{10}$$

The retrieval operations have to be carried out according to the block priority order:

$$\sum_{t=1}^{T} v_{nt} \geq \sum_{t=1}^{T} v_{n+1t} + 1, \forall n = 1..N-1 \tag{11}$$

The blocks are out of the bay if they have been previously retrieved:

$$v_{nt} = \sum_{i=1}^{W}\sum_{j=1}^{H}\sum_{t'=1}^{t-1} y_{ijnt'}, \forall n = 1..N, t = 1..T \tag{12}$$

The block n is relocated from the slot (i,j) in time period t if it is placed in that slot ($b_{ijnt} = 1$) and there is one relocation movement associated with one of the slots (i,j), $(i,j-1)$,..., $(i,j-Q-1)$. In these cases, at least 1, 2,..., Q blocks have to be moved to other slots, respectively:

$$a_{ijnt} \leq b_{ijnt},$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1 \tag{13}$$

$$a_{ijnt} \leq \sum_{\substack{j'=0 \\ j-j'\geq 1}}^{Q-1}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{\substack{q=1 \\ j-j'+q-1\leq H \\ l+q-1\leq H \\ q>j'}}^{Q} x_{ij-j'klqt},$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1 \tag{14}$$

$$a_{ijnt} \geq b_{ijnt} +$$
$$\sum_{\substack{j'=0 \\ j-j'\geq 1}}^{Q-1}\sum_{k=1}^{W}\sum_{l=1}^{H}\sum_{\substack{q=1 \\ j-j'+q-1\leq H \\ l+q-1\leq H \\ q>j'}}^{Q} x_{ij-j'klqt} - 1,$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1 \tag{15}$$

The location of each relocated block depends on its previous position and the relocation movement carried out:

$$c_{ijnt} \leq b_{ijnt+1},$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1 \quad (16)$$

$$c_{ijnt} \leq \sum_{\substack{j'=0 \\ j-j' \geq 1}}^{Q-1} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{\substack{q=1 \\ j-j'+q-1 \leq H \\ l+q-1 \leq H \\ q > j'}}^{Q} x_{klij-j'qt},$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1 \quad (17)$$

$$c_{ijnt} \geq b_{ijnt+1} + \sum_{\substack{j'=0 \\ j-j' \geq 1}}^{Q-1} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{\substack{q=1 \\ j-j'+q-1 \leq H \\ l+q-1 \leq H \\ q > j'}}^{Q} x_{klij-j'qt} - 1,$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1 \quad (18)$$

The location of each block depends on its previous location and the movements carried out:

$$b_{ijnt} = b_{ijnt-1} + c_{ijnt-1} - a_{ijnt-1} - y_{ijnt-1},$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 2..T \quad (19)$$

Number of blocks placed on each one to be relocated:

$$d_{ijnt} \leq Q \cdot a_{ijnt},$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1,$$
$$\text{subject to } Q - 1 \geq 1 \quad (20)$$

$$d_{ijnt} \leq \sum_{j'=j+1}^{H} \sum_{n'=1}^{N} a_{ij'n't},$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1 \quad (21)$$

$$d_{ijnt} \geq \sum_{j'=j+1}^{H} \sum_{n'=1}^{N} a_{ij'n't} - Q \cdot (1 - a_{ijnt}),$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1,$$
$$\text{subject to } Q - 1 \geq 1 \quad (22)$$

Number of blocks placed on each block after a relocation movement is performed:

$$e_{ijnt} = c_{ijnt} \sum_{j'=j+1}^{H} \sum_{n'=1}^{N} c_{ij'n't},$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1 \quad (23)$$

$$e_{ijnt} \leq Q \cdot c_{ijnt},$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1,$$
$$\text{subject to } Q - 1 \geq 1 \quad (24)$$

$$e_{ijnt} \leq \sum_{j'=j+1}^{H} \sum_{n'=1}^{N} c_{ij'n't},$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1 \quad (25)$$

$$e_{ijnt} \geq \sum_{j'=j+1}^{H} \sum_{n'=1}^{N} c_{ij'n't} - Q \cdot (1 - c_{ijnt}),$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1,$$
$$\text{subject to } Q - 1 \geq 1 \quad (26)$$

The number of blocks placed on each block involved in the relocation movements have to be equal before and after the movement:

$$\sum_{i=1}^{W} \sum_{j=1}^{H} d_{ijnt} = \sum_{i=1}^{W} \sum_{j=1}^{H} e_{ijnt},$$
$$\forall n = 1..N, t = 1..T-1 \quad (27)$$

Relocation of q blocks from the slot (i, j) is allowed only when there are exactly q blocks being removed:

$$\sum_{i=1}^{W} \sum_{j=1}^{H} \sum_{n=1}^{N} a_{ijnt} = \sum_{i=1}^{W} \sum_{j=1}^{H} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{\substack{q=1 \\ j+q-1 \leq H \\ l+q-1 \leq H}}^{Q} q \cdot x_{ijklqt},$$
$$\forall t = 1..T-1 \quad (28)$$

Relocation movements within a stack are not allowed:

$$x_{ijilqt} = 0,$$
$$\forall i = 1..W, j, l = 1..H, q = 1..Q, t = 1..T$$
$$\text{subject to } j + q - 1 \leq H \text{ and } l + q - 1 \leq H \quad (29)$$

Domain of the variables:

$$b_{ijnt} \in \{0, 1\},$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T \quad (30)$$

$$x_{ijklqt} \in \{0, 1\},$$
$$\forall i, k = 1..W, j, l = 1..H, q = 1..Q, t = 1..T$$
$$\text{subject to } j + q - 1 \leq H \text{ and } l + q - 1 \leq H \quad (31)$$

$$y_{ijnt} \in \{0, 1\},$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T \quad (32)$$
$$v_{nt} \in \{0, 1\}, \forall n = 1..N, t = 1..T \quad (33)$$

$$a_{ijnt} \in [0..1],$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1 \quad (34)$$

$$c_{ijnt} \in [0..1],$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1 \quad (35)$$

$$d_{ijnt} \in [0..Q-1],$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1 \quad (36)$$

$$e_{ijnt} \in [0..Q-1],$$
$$\forall i = 1..W, j = 1..H, n = 1..N, t = 1..T-1 \quad (37)$$

Heuristic Approach

As described in the next section, the large number of variables of the mathematical formulation previously introduced constricts its scope of application to only small instances. In order to overcome this limitation, in the following we propose a heuristic approach which allows us to obtain high quality solutions by means of short computational times.

The rationale behind the proposed heuristic is both to retrieve the requested blocks as soon as possible while reducing the number of non-located blocks into the bay (Equation 2). It should be noted that the shorter the number of non-located blocks into the bay the shorter the probability of requiring additional relocation movements in the future.

The pseudocode of the heuristic is depicted in Algorithm 2. As done in Algorithm 1, it iterates over all the requested priorities (lines 1 - 21). At each step, the next block to retrieve, c^*, is found (line 2). If c^* is currently placed at the top of some stack into the bay, it can be retrieved directly (line 20). Otherwise, the set of blocks placed above c^*, denoted by $O(c^*)$ (Equation 4), is checked (line 3). See block $c^* = 1$ in Figure 1, for which $O(1) = \{5, 7, 13\}$.

The main objective is to reduce the number of non-located blocks by building stacks with only well-located blocks (Equation 3). That is, stacks in which all the blocks are placed following their increasing priority order, in such a way that, the topmost block has the highest relative priority. With this goal in mind, the heuristic identifies the largest subset of $O(c^*)$ placed at the top of $s(c^*)$ that already satisfies the increasing priority order, denoted by $O(c^*)^+$ (line 8). For instance, if $Q = 2$ is set in Figure 1, we obtain $O(c^*)^+ = \{5, 7\}$. Note that the maximum size of $O(c^*)^+$ is constrained by Q and the cardinality of $O(c^*)$ (line 5).

The target stack of $O(c^*)^+$ is selected by using the following scoring function over each stack s:

$$f(O(c^*)^+, s) = \begin{cases} 0, & \text{if } (h(s) + |O(c^*)^+| > H) \vee \\ & (s = s(c^*)) \\ \max(s), & \text{otherwise} \end{cases} \quad (38)$$

The previous function $f(\cdot, \cdot)$ measures the attractiveness of the stacks by estimating the time in which they would require a relocation movement if $O(c^*)^+$ is placed on them. Stacks with no empty slots and the source stack are not included into the set of feasible target stacks, denoted by Φ (line 9). The target stack of $O(c^*)^+$, denoted by s^*, is selected at random among those δ stacks with the highest value (line 11), where $\delta \in [1..(W-1)]$ is a parameter whose value is set by the user. It is worth mentioning that if any feasible stack is found for relocating $O(c^*)^+$, the heuristic reduces the length of the subset of blocks to relocate (line 16). The process finishes when all the blocks initially included into $O(c^*)$ are relocated in another stack.

After relocating $O(c^*)^+$ in s^*, there could be empty slots, denoted by $e(s^*) = H - (h(s^*) + |O(c^*)^+|)$. These slots can be exploited in order to reduce the number of non-located blocks. Before relocating $O(c^*)^+$, the heuristic i) identifies all the topmost non-located blocks of some stack s ($s \neq s(c^*)$) and ii) selects that with the lowest priority

Algorithm 2 Pseudocode of the heuristic approach aimed at solving the Q-BRP

Require: Bay with the blocks to retrieve
Require: δ, range of target stacks
Ensure: Solution of the Q-BRP
1: **for** $p = 1 \rightarrow K$ **do**
2: $c^* \leftarrow$ Find block with priority p
3: $O(c^*) \leftarrow$ Blocks placed on c^* (Equation 4)
4: **while** $(O(c^*) \neq \emptyset)$ **do**
5: $q^* \leftarrow \min\{Q, |O(c^*)|\}$
6: $placed \leftarrow false$
7: **while** $(!placed)$ **do**
8: $O(c^*)^+ \leftarrow$ Find the largest subset of $O(c^*)$ ($|O(c^*)^+| \leq q^*$) with increasing priority order
9: $\Phi \leftarrow$ Feasible target stacks for $O(c^*)^+$
10: **if** $(\Phi \neq \emptyset)$ **then**
11: $s^* \leftarrow$ Select stack from Φ by using δ
12: Fill s^* with non-located blocks
13: Relocate $O(c^*)^+$ from $s(c^*)$ in s^*
14: $placed \leftarrow true$
15: **else**
16: $q^* \leftarrow q^* - 1$
17: **end if**
18: **end while**
19: **end while**
20: Retrieve c^* from the top of its stack
21: **end for**

that would satisfy the increasing priority order. The reason is that we pursue to maximize the number of potential blocks to be placed in s^* without requiring new relocations in the future. Figure 2 illustrates the filling strategy. In this case, the feasible target stacks of $O(1)^+ = \{5, 7\}$ are 1, 3, and 5. The attractiveness measure reported by Equation (38) is depicted at the top of the figure. Before relocating $O(1)^+$ in the stack 5, the potential non-located blocks to fill the empty slot ($e(5) = 1$) are identified, blocks 8, 16, and 4. As we can see, the block 8 is relocated due to the fact that it is that with the lowest priority that satisfy the increasing priority order.

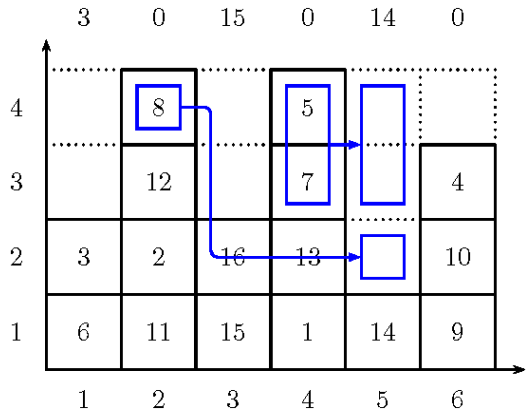

Figure 2: Filling strategy applied before relocating $O(c^*)^+ = \{5, 7\}$

Computational Results

The experiments presented along this section were carried out on a PC with an Intel Core 2 Duo E8500 3.16 GHz and 4 GB of RAM, the MILP was implemented by using CPLEX 12.4[1] and the heuristic algorithm was programmed by the Java SE 7 language. In the experiments we have used the most extended benchmark suite for the BRP (Caserta, Voβ, and Sniedovich 2011). It is composed of 840 realistic instances generated at random in which each slot contains one block. In all the cases we include two empty rows at the top of the bays with the aim of allowing relocation movements.

The first computational experiment pursues to evaluate the performance of the proposed MILP. Table 1 shows the results obtained for the 3x3 instances when $Q = 1$ and $Q = 2$ by using the solution reported by Algorithm 1 ($max = 100$) as initial solution. In spite of the fact we are addressing the smallest instances, the computational time (several days in some cases) required by the MILP increases exponentially with the value of Q and makes it impractical in real-world scenarios. The reason is found in the large number of variables involved, which are aimed at identify the state of each block in each slot during the whole planning horizon.

In the following we compare the results obtained by our heuristic and those reported by that proposed by (Kim and Hong 2006) for the BRP ($Q = 1$). It is worth mentioning that the computational times required by the heuristic by (Kim and Hong 2006) are shorter than 0.1 seconds in all the cases. We have executed our heuristic with $\delta = 3$ and embedded it into a restarting strategy that finished when 100 iterations have been executed without any improvement in the objective function value of the best solution found so far. Table 2 shows the comparison between both heuristics when solving the first 5 instances of 9 groups from the benchmark suite with $Q = 1$. Additionally, we report the results obtained when setting $Q = 2$ and $Q = \infty$ (all the blocks in a stack can be relocated at once) with future comparative purposes.

As can be seen, when $Q = 1$ our heuristic reports solutions with at least the same or better quality than those obtained by (Kim and Hong 2006) in most of the cases, with an overall average improvement of 1.38 relocation movements. The computational times are short (less than 1 second) in all the cases, in such a way that, the heuristic can be efficiently embedded into integrated warehouse management systems. In those cases in which we have not obtained the best known solution (instances 4x5-1 and 4x5-5) our heuristic reports high quality solutions. It should be noted that we have obtained the optima solutions for the first 5 instances belonging to the group 3x3 (see Table 1).

On the other hand, if we carry out an analysis concerning the impact of increasing the number of blocks handled by the equipment we can see that the relocations perform during the retrieval process is highly influenced. The number of relocation movements decreases fastly as the value of Q increases, from 8.75 relocation movements on average down to 7.02 and 6.58 relocations when we set $Q = 2$ and $Q = \infty$, respectively.

[1] http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

Index	MILP (Q=1)		MILP (Q=2)	
	f_{opt}	t (s.)	f_{opt}	t (s.)
1	6	5680.46	4	18609.78
2	5	5268.28	4	37707.09
3	2	16.62	2	217.17
4	4	1284.81	2	142.75
5	1	0.55	1	12.29
6	6	31760.24	5	374588.30
7	6	20871.43	3	5198.82
8	2	30.33	2	4640.57
9	7	160620.96	5	756859.12
10	5	2832.22	4	8841.69
11	3	216.78	2	2392.90
12	5	3003.08	4	7260.20
13	8	231314.47	5	613968.37
14	7	16483.72	4	15293.46
15	6	66376.00	3	9073.46
16	7	37603.17	5	66814.56
17	5	3934.84	4	42402.27
18	2	3.97	2	32.07
19	8	224421.26	4	17181.60
20	7	65574.23	4	19781.15
21	7	75849.64	4	137415.12
22	4	9552.62	4	20611.03
23	6	2541.81	4	577138.87
24	6	16114.17	5	232760.50
25	4	269.68	3	2141.35
26	4	82.80	3	5520.55
27	5	4749.58	4	15336.56
28	5	6015.75	5	137143.41
29	7	20577.23	5	68665.67
30	6	14026.59	3	6562.33
31	5	6972.97	4	26523.38
32	2	25.68	2	507.22
33	3	1930.35	3	4172.73
34	6	9843.51	4	7302.84
35	5	4636.64	4	14539.76
36	7	136872.48	6	1828854.31
37	5	2606.56	4	54184.31
38	4	743.46	4	28461.44
39	0	0.48	0	2.28
40	6	72599.96	3	7968.58
	4.98	31582.73	3.58	129420.75

Table 1: Performance of the MILP over 40 instances with 3 stacks, 3 tiers and 2 empty rows at the top of the bay

Concluding Remarks and Future Lines

The time required when retrieving blocks from a warehouse is a critical factor. This time is increased when the blocks are not placed according with their retrieval order and, therefore, relocations must be carried out. In this work we model this planning problem by a MILP. Reducing its computational burden is an open direction for further research. On the other hand, we have proposed a heuristic algorithm aimed at minimizing the number of non-located blocks. In future works we are going to study the integration of the Q-BRP with

Instance			Kim & Hong	Heuristic (Q=1)		Heuristic (Q=2)		Heuristic (Q=∞)	
H	W	Index		f_{Heu}	t (s.)	f_{Heu}	t (s.)	f_{Heu}	t (s.)
3	3	1	7	**6**	0.94	4	0.80	3	0.72
		2	7	**5**	0.68	4	0.68	4	0.60
		3	2	2	0.34	2	0.23	2	0.25
		4	4	4	0.33	2	0.21	2	0.21
		5	1	1	0.30	1	0.14	1	0.25
3	4	1	5	5	0.12	5	0.93	5	0.89
		2	4	**3**	0.48	2	0.40	2	0.49
		3	9	**7**	0.99	6	0.75	6	0.82
		4	6	**5**	0.30	5	0.69	5	0.72
		5	7	**6**	0.31	5	0.22	5	0.82
3	5	1	8	**6**	0.16	5	0.19	5	0.81
		2	10	**7**	0.16	5	0.11	5	0.75
		3	9	**8**	0.35	7	0.11	7	0.35
		4	6	6	0.21	5	0.23	5	0.26
		5	12	**9**	0.28	7	0.24	7	0.30
3	6	1	12	**10**	0.28	6	0.20	6	0.18
		2	11	**7**	0.48	7	0.79	7	0.11
		3	11	11	0.47	9	0.75	9	0.58
		4	7	7	0.53	6	0.31	5	0.25
		5	4	4	0.24	4	0.17	4	0.19
3	7	1	7	7	0.23	7	0.26	7	0.23
		2	11	**10**	0.10	8	0.57	8	0.52
		3	10	**9**	0.86	6	0.60	6	0.51
		4	8	8	0.47	7	0.48	7	0.48
		5	12	12	0.61	9	0.81	9	0.65
3	8	1	9	**8**	0.27	7	0.27	7	0.27
		2	10	10	0.10	8	0.11	7	0.15
		3	12	**9**	0.67	8	0.50	8	0.58
		4	11	**10**	0.12	9	0.48	9	0.56
		5	14	**13**	0.64	10	0.82	10	0.53
4	4	1	11	**10**	0.27	8	0.22	7	0.19
		2	15	**10**	0.47	9	0.64	9	0.12
		3	13	**10**	0.43	8	0.36	8	0.33
		4	9	**7**	0.44	7	0.25	7	0.30
		5	12	**9**	0.33	8	0.38	8	0.32
4	5	1	13	17	0.19	12	0.30	10	0.35
		2	11	**10**	0.85	9	0.51	8	0.55
		3	15	**13**	0.11	10	0.42	9	0.44
		4	11	**9**	0.32	7	0.36	6	0.34
		5	14	15	0.10	10	0.50	9	0.44
4	6	1	20	**17**	0.39	11	0.28	9	0.82
		2	8	**7**	0.78	6	0.87	6	0.58
		3	14	**13**	0.59	9	0.82	9	0.84
		4	20	**15**	0.73	11	0.65	9	0.60
		5	24	**17**	0.85	15	0.61	9	0.85
			10.13	8.75	0.41	7.02	0.45	6.58	0.49

Table 2: Comparison between the heuristic proposed by (Kim and Hong 2006) and our heuristic with different values of Q

other logistical problems such as dispatching of vehicles.

Acknowledgments

This work has been partially funded by the European Regional Development Fund, the Spanish Ministry of Economy and Competitiveness (project TIN2012-32608). Christopher Expósito-Izquierdo thanks the Canary Government the financial support he receives through his doctoral grant.

References

Bortfeldt, A., and Forster, F. 2012. A tree search procedure for the container pre-marshalling problem. *European Journal of Operational Research* 217(3):531 – 540.

Breiter, A.; Hegmanns, T.; Hellingrath, B.; and Spinler, S. 2009. Coordination in supply chain management - review and identification of directions for future research. In Voβ, S.; Pahl, J.; and Schwarze, S., eds., *Logistik Management*. Physica-Verlag HD. 1–35.

Caserta, M.; Schwarze, S.; and Voβ, S. 2009. A new binary description of the blocks relocation problem and benefits in a look ahead heuristic. In Cotta, C., and Cowling, P., eds., *Evolutionary Computation in Combinatorial Optimization*, volume 5482 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. 37–48.

Caserta, M.; Schwarze, S.; and Voβ, S. 2011. Container rehandling at maritime container terminals. In Böse, J. W.; Sharda, R.; and Voβ, S., eds., *Handbook of Terminal Planning*, volume 49 of *Operations Research/Computer Science Interfaces Series*. Springer New York. 247–269.

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

Caserta, M.; Voβ, S.; and Sniedovich, M. 2011. Applying the corridor method to a blocks relocation problem. *OR Spectrum* 33:915–929.

Chan, F. T., and Chan, H. 2011. Improving the productivity of order picking of a manual-pick and multi-level rack distribution warehouse through the implementation of class-based storage. *Expert Systems with Applications* 38(3):2686 – 2700.

Choe, R.; Park, T.; Oh, M.-S.; Kang, J.; and Ryu, K. 2011. Generating a rehandling-free intra-block remarshaling plan for an automated container yard. *Journal of Intelligent Manufacturing* 22(2):201–217.

de Koster, R.; Le-Duc, T.; and Roodbergen, K. J. 2007. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research* 182(2):481 – 501.

Dekker, R.; Voogd, P.; and Asperen, E. 2007. Advanced methods for container stacking. In Kim, K. H., and Günther, H.-O., eds., *Container Terminals and Cargo Systems*. Springer Berlin Heidelberg. 131–154.

Expósito-Izquierdo, C.; Melián-Batista, B.; and Moreno-Vega, M. 2012. Pre-marshalling problem: Heuristic solution method and instances generator. *Expert Systems with Applications* 39(9):8337 – 8349.

Grosse, E. H.; Glock, C. H.; and Jaber, M. Y. 2013. The effect of worker learning and forgetting on storage reassignment decisions in order picking systems. *Computers & Industrial Engineering*.

Gu, J.; Goetschalckx, M.; and McGinnis, L. F. 2010. Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research* 203(3):539 – 549.

Gudehus, T., and Kotzab, H. 2009. Storage systems. In *Comprehensive Logistics*. Springer Berlin Heidelberg. 449–533.

Kang, J.; Ryu, K.; and Kim, K. 2006. Deriving stacking strategies for export containers with uncertain weight information. *Journal of Intelligent Manufacturing* 17(4):399–410.

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

Lee, Y., and Hsu, N.-Y. 2007. An optimization model for the container pre-marshalling problem. *Computers & Operations Research* 34(11):3295 – 3313.

Melo, M.; Nickel, S.; and da Gama, F. S. 2009. Facility location and supply chain management - A review. *European Journal of Operational Research* 196(2):401 – 412.

Park, T.-K., and Kim, K. H. 2010. Comparing handling and space costs for various types of stacking methods. *Computers & Industrial Engineering* 58(3):501 – 508.

Rei, R. J., and Pedroso, J. P. 2012. Heuristic search for the stacking problem. *International Transactions in Operational Research* 19(3):379–395.

Rei, R., and Pedroso, J. 2013. Tree search for the stacking problem. *Annals of Operations Research* 203(1):371–388.

Tompkins, J.; White, Y.; Bozer, E.; and Tanchoco, J. 2010. *Facilities planning*. John Wiley & Sons, 4th edition edition.

van Zelst, S.; van Donselaar, K.; van Woensel, T.; Broekmeulen, R.; and Fransoo, J. 2009. Logistics drivers for shelf stacking in grocery retail stores: Potential for efficiency improvement. *International Journal of Production Economics* 121(2):620 – 632.

Zhang, H.; Guo, S.; Zhu, W.; Lim, A.; and Cheang, B. 2010. An investigation of ida* algorithms for the container relocation problem. In Garca-Pedrajas, N.; Herrera, F.; Fyfe, C.; Bentez, J.; and Ali, M., eds., *Trends in Applied Intelligent Systems*, volume 6096 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. 31–40.

Zhao, W., and Goodchild, A. V. 2010. The impact of truck arrival information on container terminal rehandling. *Transportation Research Part E: Logistics and Transportation Review* 46(3):327 – 343.