Les Cahiers du GERAD ISSN: 0711–2440

Online heuristics for unloading boxes off a gravity conveyor

P. Bastiste, R. Bürgy,
A. Hertz, D. Rebaïne

G–2016–20

March 2016

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Online heuristics for unloading boxes off a gravity conveyor

Pierre Baptiste [a]
Reinhard Bürgy [a]
Alain Hertz [a]
Djamal Rebaïne [b]

[a] *GERAD & Département de mathématiques et de génie industriel, Polytechnique Montréal, Montréal (Québec) Canada, H3C 3A7*

[b] *GERAD & Département d'informatique et de mathématique, Université du Québec à Chicoutimi, Saguenay (Québec) Canada, G7H 2B1*

pierre.bastiste@polymtl.ca
reinhard.burgy@gerad.ca
alain.hertz@gerad.ca
drebaine@uqac.ca

March 2016

Les Cahiers du GERAD
G–2016–20

Abstract: This paper addresses the problem of minimizing the number of moves to unload a set of boxes off a gravity conveyor by a forklift. If the input data is known in advance, the problem is efficiently solvable with a dynamic programming approach. However, this method is rarely applicable in practice for two reasons. First, the problem generally occurs in a real-time environment where the input data is revealed over time. Second, computing devices are in most cases not available in forklifts for decision making. Online approaches that can easily be applied by human operators are therefore sought in practice.

With this in mind, we first propose some intuitive approaches and analyze their performance through an extensive experimental study. The results show that these approaches are quite inefficient as they are on average between 14.7% and 59.3% above the optimum. A less intuitive but still simple approach is then designed that consistently produces good results with an average gap of 6.1% to the optimum.

Key Words: Gravity conveyor; online optimization; sequencing heuristics.

Acknowledgments: The second author's work was partially funded by the Swiss National Science Foundation under Grant P2FRP2_161720.

1 Introduction

The problem addressed in this paper was introduced by Baptiste, Rebaïne, and Brika (2011), and was met in a company that produces household appliances. It can be described as follows. Different versions of a product are assembled on assembly lines, and after a quality inspection, each product is packed in a standardized box. A single gravity conveyor line transports these boxes to an unloading zone where a human operator moves them with a forklift to their destination, which is for example a semi-trailer or a truck. The conveyor consists of three areas, namely an accessible area where the operator can pick boxes, a visible area where the operator can see but not pick the boxes, and an invisible area where the boxes are hidden. Each box has a predefined destination. For simplicity, we assign to each destination a distinct color, and depict each box with the color of its destination. The forklift cannot pick more than F boxes at a time, and these boxes must be of the same color and contiguous on the conveyor. In the example of Figure 1, the accessible area contains nine boxes and the visible area three boxes, there are three destinations, and the forklift can pick at most three boxes at a time. As a contiguous set of boxes is unloaded, a gap is momentarily formed. It is immediately filled by the boxes to its right which are positioned at a higher height and slide down, thus leading to new boxes entering the accessible and visible areas.

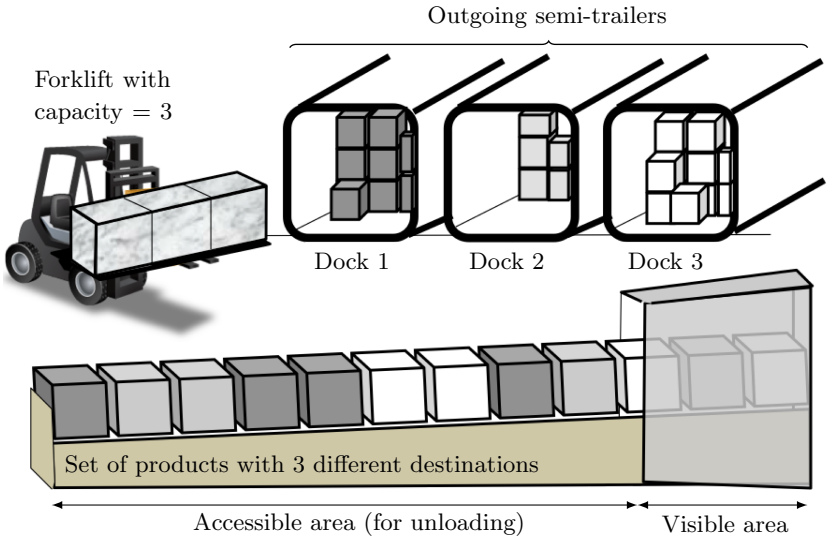

Figure 1: Unloading boxes off a gravity conveyor.

The sequence in which the boxes are revealed has a great influence on the number of moves needed to unload all the boxes. We aim at developing strategies that unload all boxes in as few moves as possible. Clearly, the sequence in which the boxes are placed on the gravity conveyor has a significant influence on the minimum number of moves needed to unload them. Unfortunately, this sequence must be taken as given, as it is an outcome of decisions made by the production and quality control departments, which generally do not consider the consequences on the gravity conveyor unloading problem.

Some decision problems arising before and after unloading boxes off a conveyor are well studied. There exists for example a large body of research on assembly line sequencing problems (see e.g., Sumichrast and Russell, 1990; Becker and Scholl, 2006), on quality inspection sequencing problems (see e.g., Raz and Thomas, 1983; Ding, Greenberg, and Matsuo, 1998), and on truck loading problems (see e.g., Morabito, Morales, and Widmer, 2000; Lodi, Martello, and Vigo, 2002). There is also a considerable amount of papers addressing the retrieval of specific items from conveyors. For example, Bozma and Kalaılıoğlu (2012) consider a problem where a set of robots pick items from a moving conveyor band. Bartholdi and Platzman (1986) developed heuristic retrieval strategies for picking a set of items from a carousel conveyor, and Ghosh and Wells (1992) proposed optimal retrieval strategies for some of these problems.

In contrast, conveyor unloading problems received little attention in the literature. Baptiste et al. (2013) studied an offline version of our gravity conveyor unloading problem, where the complete sequence of the boxes is assumed to be known in advance, i.e., there is no invisible area. They developed an efficient dynamic programming based approach that finds an optimal solution in time $O(n^3 A \log F)$, where n is the total number of boxes, A the size of the accessible area, and F the forklift capacity. However, this approach is rarely applicable in practice, mainly for the following two reasons. First, only a part of the boxes is visible at each time, while the other part is hidden. The boxes actually arrive at the gravity conveyor in a continuous fashion without knowing in advance the order of their arrival; this is due to various unknowns inherent to the quality inspection process. Second, there is no computing device embedded in the forklift in order to generate the next move to be executed. As a consequence, simple online approaches that can be applied by human operators are sought in practice.

To the best of our knowledge, the only contribution tackling the online version of our problem is (Baptiste, Rebaïne, and Brika, 2012). They propose three simple heuristics based on intuitive rules for selecting the next boxes to be moved. They observe that the obtained results are sometimes more than 20% above the optimal value, and conclude that rules based on common sense may have an unsatisfactory performance. These results were the motivation and the starting point for the approaches developed in this article.

This paper is organized as follows. Section 2 gives a formal description of the problem. Section 3 addresses valid, dominant, and optimal moves. Section 4 proposes and evaluates the performance of an online algorithm. Based on the obtained results, a variant of the online algorithm using an improved rule is developed and evaluated in Section 5. Concluding remarks are given in Section 6.

2 Problem formulation

Consider a gravity conveyor with three areas, namely an accessible area containing the first A boxes, a visible area preceding the accessible area and containing the next V boxes, and an invisible area of boxes preceding the visible area. The forklift has a fork of size F.

Let c_i be the color of the box at position i, and let $C = (c_1, \ldots, c_{|C|})$ be the sequence of colors on the conveyor. We identify a box by its position in C, which means that box i has color c_i. We say that the positions are increasing from *left to right*, and we denote an instance by (C, A, V, F). Figure 2 illustrates the example (C, 10, 3, F) with C = (1, 2, 3, 1, 4, 3, 1, 2, 2, 4, 1, 1, 2, 3, 3, 1, 1, 2, 1, 2, 3, 1, 1, 2, 4, 2, 2, 1, 4, 2), representing color 1, 2, 3, and 4 by white, light grey, dark grey, and black, respectively. We will use this example in the sequel of the paper with a fork size $F = 3$. The removal of a group of contiguous boxes is called *a move*. Valid moves are defined as follows.

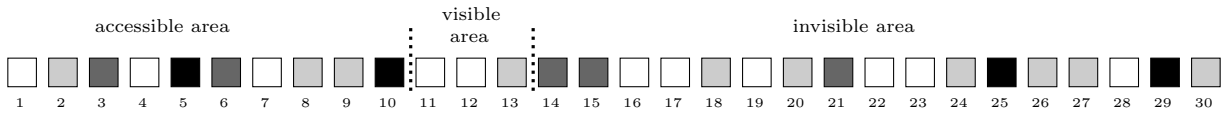

Figure 2: An example with 30 boxes and 4 different destinations indicated with the colors white, light grey, dark grey and black. The numbers below the boxes indicate the position of the boxes.

Definition 1 *Given instance (C, A, V, F), a move is specified by a pair (p, ℓ), where p is the position of the leftmost removed box and ℓ is the number of removed boxes. A move (p, ℓ) is* valid *if*

a) $p + \ell - 1 \leq \min\{|C|, A\}$,
b) $c_p = c_{p+i}$ *for all* $i \in \{0, \ldots, \ell - 1\}$, *and*
c) $\ell \leq F$.

Note that the validity conditions in the above definition state that a) the removed boxes must exist and be in the accessible area, b) the removed boxes must be all of the same color, and c) the number of removed

boxes cannot be larger than F. We denote by $C \odot (p, \ell)$ the ordered set of colors that results from a valid move (p, ℓ) applied to $C = (c_1, \ldots, c_{|C|})$, i.e., $C \odot (p, \ell) = (c_1, \ldots, c_{p-1}, c_{p+\ell}, \ldots, c_{|C|})$.

A *strategy* for an instance (C, A, V, F) consists of a sequence of valid moves that clear the conveyor. The objective is to determine a strategy that generates a minimum number of moves.

An offline version of the problem is studied in (Baptiste et al., 2013) where it is assumed that all the boxes are visible from the beginning. They developed a polynomial time algorithm that finds an optimal strategy for this problem. Denote by $OPT(C, A, F)$ the optimal value for (C, A, V, F). Clearly, $OPT(C, A, F)$ does not depend on V and is also an optimal value for the online version of the problem.

3 Valid, dominant, and optimal moves

Let $\Sigma(C, A, F)$ be the set of valid moves for instance (C, A, V, F). The number $|\Sigma(C, A, F)|$ of valid moves does not depend on V, and is bounded by $F \max\{|C|, A\}$. Indeed, each box $p \in \{1, \ldots, \max\{|C|, A\}\}$ can be the leftmost removed box of a group of size $\ell \in \{1, \ldots, \min\{|C| - p + 1, F\}\}$. Among all moves in $\Sigma(C, A, F)$, we aim at selecting a move (p^*, ℓ^*) that minimizes the total number of moves needed to remove all the remaining boxes. In other words, (p^*, ℓ^*) is a move in $\Sigma(C, A, F)$ such that

$$OPT(C \odot (p^*, l^*), A, F) = \min\{OPT(C \odot (p, l), A, F) : (p, l) \in \Sigma(C, A, F)\}.$$

Note that $OPT(C, A, F) = 1 + OPT(C \odot (p^*, l^*), A, F)$. This justifies the following definition.

Definition 2 *A move $(p, l) \in \Sigma(C, A, F)$ is optimal if $OPT(C \odot (p, l), A, F) = OPT(C, A, F) - 1$.*

For a subset $R \subseteq \{1, \ldots, |C|\}$ of box positions, let $C' = (c'_1, \ldots, c'_{|C|-|R|})$ be the subsequence obtained from $C = (c_1, \ldots, c_{|C|})$ by removing all c_i with $i \in R$. For every $i \notin R$, we thus have $c_i = c'_{i-\Delta_i}$, with $\Delta_i = |R \cap \{1, \ldots, i-1\}|$. We write $C' = C - R$ and $C' \sqsubseteq C$.

Proposition 3 *Let (C, A, V, F) and (C', A, V, F) be two instances with $C' = C - R$ for some $R \subseteq \{1, \ldots, |C|\}$. Then $OPT(C', A, F) \leq OPT(C, A, F)$.*

Proof. Let $(p, \ell) \in \Sigma(C, A, F)$ be a valid move for (C, A, V, F). If $\{p, \ldots, p+\ell-1\} \subseteq R$ then $C' \sqsubseteq C \odot (p, \ell)$. Otherwise, let p' be the smallest index in $\{p, \ldots, p+\ell-1\}$ such that $p' \notin R$ and let $\ell' = \ell - |R \cap \{p, \ldots, p+\ell-1\}|$. Clearly, $(p' - \Delta_{p'}, \ell')$ is a valid move for (C', A, V, F) and $C' \odot (p' - \Delta_{p'}, \ell') \sqsubseteq C \odot (p, \ell)$. It follows that, given any strategy S for (C, A, V, F), a corresponding strategy S' can be obtained for (C', A, V, F) by replacing each move in S with the removal of the same set of contiguous boxes, except those in R. If a move in S removes only boxes in R, then no move is performed for (C', A, V, F). Hence, S' contains at most as many moves as S, and $OPT(C', A, F) \leq OPT(C, A, F)$ follows. □

In the online version of the problem, it is generally not possible to decide if a valid move is optimal due to the missing information on the boxes in the invisible area. Nevertheless, some valid moves can be considered as non-interesting as they are dominated by others. We now show how to determine a subset of $\Sigma(C, A, F)$ that necessarily contains an optimal move.

The ordered set of boxes in the accessible area of an instance (C, A, V, F) can be partitioned into maximal ordered subsets of contiguous boxes having the same color. We denote by $\mathcal{B}(C, A)$ the resulting partition, and every element $b \in \mathcal{B}(C, A)$ is called a *block*. For the example of Figure 2, we have $\mathcal{B}(C, A) = ((1), (2), (3), (4), (5), (6), (7), (8, 9), (10))$.

Let $\Sigma^{\mathrm{R}}(C, A, F)$ be the subset of valid moves (p, ℓ) such that p is the leftmost box of a block $b = (p, \ldots, p + |b| - 1)$ and $\ell = \min\{|b|, F\}$. We will prove that $\Sigma^{\mathrm{R}}(C, A, F)$ necessarily contains an optimal move. Note that the size of $\Sigma^{\mathrm{R}}(C, A, F)$ is at most A. For the example of Figure 2, we have $\Sigma^{\mathrm{R}}(C, A, F) = \{(1,1), (2,1), (3,1), (4,1), (5,1), (6,1), (7,1), (8,2), (10,1)\}$.

Proposition 4 *$\Sigma^R(C, A, F)$ contains an optimal move for (C, A, V, F).*

Proof. Let (p^*, ℓ^*) be an optimal move for (C, A, V, F), and let $b = (p, \ldots, p+|b|-1)$ be the block in $\mathcal{B}(C, A)$ such that $p^* \in b$. By construction, we have $(p^*, \ldots, p^* + \ell^* - 1) \subseteq b$. Let (p, ℓ) be the move in $\Sigma^{\mathrm{R}}(C, A, F)$ with $\ell = \min\{|b|, F\}$. Clearly, $\ell^* \leq F$ and $\ell^* \leq |b|$, which implies $\ell \geq \ell^*$. Hence $C \odot (p, l) \sqsubseteq C \odot (p^*, \ell^*)$, and it follows from Proposition 3 that $OPT(C \odot (p, l), A, F) \leq OPT(C \odot (p^*, \ell^*), A, F)$. Since (p^*, ℓ^*) is an optimal move, (p, ℓ) is also optimal. □

The two following propositions identify two types of optimal moves. They slightly generalize Proposition 2 and 3 of Baptiste, Rebaïne, and Brika (2011).

Proposition 5 *If $b = (p, \ldots, p + |b| - 1)$ is a block in $\mathcal{B}(C, A)$ for (C, A, V, F) with $|b| \geq F$, then (p, F) is an optimal move in $\Sigma^R(C, A, F)$*

Proof. Let S be an optimal strategy for (C, A, V, F), and assume that the j-th move in S is the first one that removes at least one box of b. Assume also that this j-th move of S removes ℓ contiguous boxes (not necessarily all in b). Let S' be the strategy obtained from S by first removing boxes $p, \ldots, p + \ell - 1$ in one move (i.e., performing move (p, ℓ)), and then executing all moves of S, except the j-th one. Clearly, all moves in S' are valid, and the sequence resulting from the first j moves of S is equal to the sequence resulting from the first j moves of S'. Hence, S' is also optimal. Consider now move (p, F) in $\Sigma^{\mathrm{R}}(C, A, F)$. Since $C \odot (p, F) \sqsubseteq C \odot (p, \ell)$, it follows from Proposition 3 that $OPT(C \odot (p, F), A, F) \leq OPT(C \odot (p, \ell), A, F)$, which means that (p, F) is optimal. □

Proposition 6 *If (p, ℓ) is a move in $\Sigma(C, A, F)$ such that $c_i \neq c_p$ for all $i \in \{1, \ldots, |C|\} \setminus \{p, \ldots, p + \ell - 1\}$ then $(p, \ell) \in \Sigma^R(C, A, F)$ and is optimal.*

Proof. Note first that the two assumptions $(p, \ell) \in \Sigma(C, A, F)$ and $c_i \neq c_p$ for all $i \in \{1, \ldots, |C|\} \setminus \{p, \ldots, p + \ell - 1\}$ imply that $b = (p, \ldots, p + \ell - 1)$ is a block in $\mathcal{B}(C, A)$, which means that $(p, \ell) \in \Sigma^{\mathrm{R}}(C, A, F)$. Let S be an optimal strategy for (C, A, V, F), and assume that the j-th move in S is the first one that removes at least one box of b. Assume also that this j-th move of S removes ℓ' contiguous boxes. Clearly, $\ell' \leq \ell$ since $c_i \neq c_p$ for all $i \in \{1, \ldots, |C|\} \setminus b$. Let S' be the strategy obtained from S by first removing boxes $p, \ldots, p + \ell' - 1$ in one move (i.e., performing move (p, ℓ')), and then executing all moves of S, except the j-th one. All moves in S' are valid, and the sequence resulting from the first j moves of S is equal to the sequence resulting from the first j moves of S'. Hence, S' is also optimal. Since $C \odot (p, \ell) \sqsubseteq C \odot (p, \ell')$, it follows from Proposition 3 that $OPT(C \odot (p, \ell), A, F) \leq OPT(C \odot (p, \ell'), A, F)$, which means that (p, ℓ) is optimal. □

4 The online algorithm

The results of Section 3 suggest the following online algorithm: while C is not empty, build the set $\mathcal{B}(C, A)$ of blocks and determine the subset $\Sigma^{\mathrm{R}}(C, A, F)$ of valid moves. If a block $b = (p, \ldots, p + |b| - 1)$ has $|b| \geq F$ boxes, then perform move (p, F). Also, if a block $b = (p, \ldots, p + |b| - 1)$ with $|b| < F$ is such that $c_i \neq c_p$ for all $i \in \{1, \ldots, |C|\} \setminus b$, then perform move $(p, |b|)$. Note that an online algorithm can check the condition $c_i \neq c_p$ for all $i \in \{1, \ldots, |C|\} \setminus b$ only if $|C| \leq A + V$. Indeed, the color of the boxes in the invisible area is not known before their appearance in the visible area. The pseudo-code of a generic online algorithm is described below, where $\mathcal{R}$ is any rule that chooses a move in $\Sigma^{\mathrm{R}}(C, A, F)$.

```
1  Input: instance (C, A, V, F);
2  while C ≠ ∅ do
3      Determine B(C, A) and Σ^R(C, A, F);
4      if there is a move (p, ℓ) ∈ Σ^R(C, A, F) with ℓ = F then
5          set (p*, ℓ*) to (p, ℓ);
6      else if |C| ≤ A + V and there is b = (p, ..., p + |b| − 1) ∈ B(C, A)
7              such that c_i ≠ c_p for all i ∈ {1, ..., |C|} \ b then
8          set (p*, ℓ*) to (p, |b|);
9      else determine a move (p*, ℓ*) ∈ Σ^R(C, A, F) with a rule R;
10     Perform move (p*, ℓ*) and replace C by C ⊙ (p*, ℓ*);
11 end_while
```

Note that the moves $(p^*, \ell^*) \in \Sigma^{\mathrm{R}}(C, A, F)$ determined by rule $\mathcal{R}$ at line 9 are such that $\ell^* = |b|$, where b is the block in $\mathcal{B}(C, A)$ that contains p^*. For ease of reading, we write that $\mathcal{R}$ chooses a block in $\mathcal{B}(C, A)$ (instead of a move in $\Sigma^{\mathrm{R}}(C, A, F)$). Also, the color $c(b)$ of a block b is the color of the boxes that it contains. We now describe six simple and intuitive rules $\mathcal{R}$, where ties are broken by choosing the block that removes the leftmost boxes. Some of these rules were introduced in (Baptiste, Rebaïne, and Brika, 2012). They are illustrated in Figure 3 using the example of Figure 2. So let $\mathcal{B}(C, A) = (b_1, \ldots, b_n)$ be the ordered set of blocks.

first Select block b_1.

rand Choose r in $\{1, \ldots, n\}$ with a uniform distribution, and select block b_r.

large Choose a block b_r such that $|b_r| = \max_{i=1}^{n} |b_i|$.

largePlus Choose a block b_r such that $(|b_r| + \lambda_r) = \max_{i=1}^{n}(|b_i| + \lambda_i)$, where λ_i is determined as follows: if $1 < i < n$ and $c(b_{i-1}) = c(b_{i+1})$, then set $\lambda_i = |b_{i-1}| + |b_{i+1}|$, else set $\lambda_i = 0$. In words, λ_i is a bonus on the value $|b_i|$ used by the previous rule, to reflect the fact that the boxes in b_{i-1} and b_{i+1} are merged into a single block after the removal of block b_i.

pop Let n_c be the number of boxes of color c in the accessible area, Choose a block b_r such that $n_{c(b_r)} = \min_{i=1}^{n} n_{c(b_i)}$

locOpt Let $m = \min\{|C|, A + V\}$ and $C' = \{c_1, \ldots, c_m\}$. The instance (C', A, V, F) has all boxes in the accessible or in the visible area. Determine an optimal strategy S for (C', A, V, F) with the algorithm of Baptiste et al. (2013), and choose b_r as the block that contains all boxes removed in the first move of S.

4.1 Computational results

In order to evaluate the performance of the online algorithms of the previous section, we consider instances with 400 boxes generated as follows. The size A of the accessible area belongs to $\{10, 20, 30, 40, 50\}$, the capacity F of the forklift belongs to $\{2, 3, 4, 5\}$, the number of different colors belongs to $\{3, 5, 8\}$, and the instance type is a, b, or c, where a means that the number of boxes of each color is the same (up to a rounding difference of one unit), b means that 50% of the boxes have color 1 while the number of boxes of the other colors are the same, and c means that each of the colors 1 and 2 represent 25% of the boxes while the number of boxes of the other colors are the same. For each combination of these parameters, we have randomly generated five sequences, which gives a total of $5 \times 4 \times 3 \times 3 \times 5 = 900$ instances.

4.1.1 No visible area

We first set the size V of the visible area to 0. For each instance and each rule, we compute the total number of moves needed to remove all boxes. For each instance we also determine the optimum value produced by the offline algorithm of Baptiste et al. (2013), where it is assumed that all boxes are visible (but not necessarily accessible).

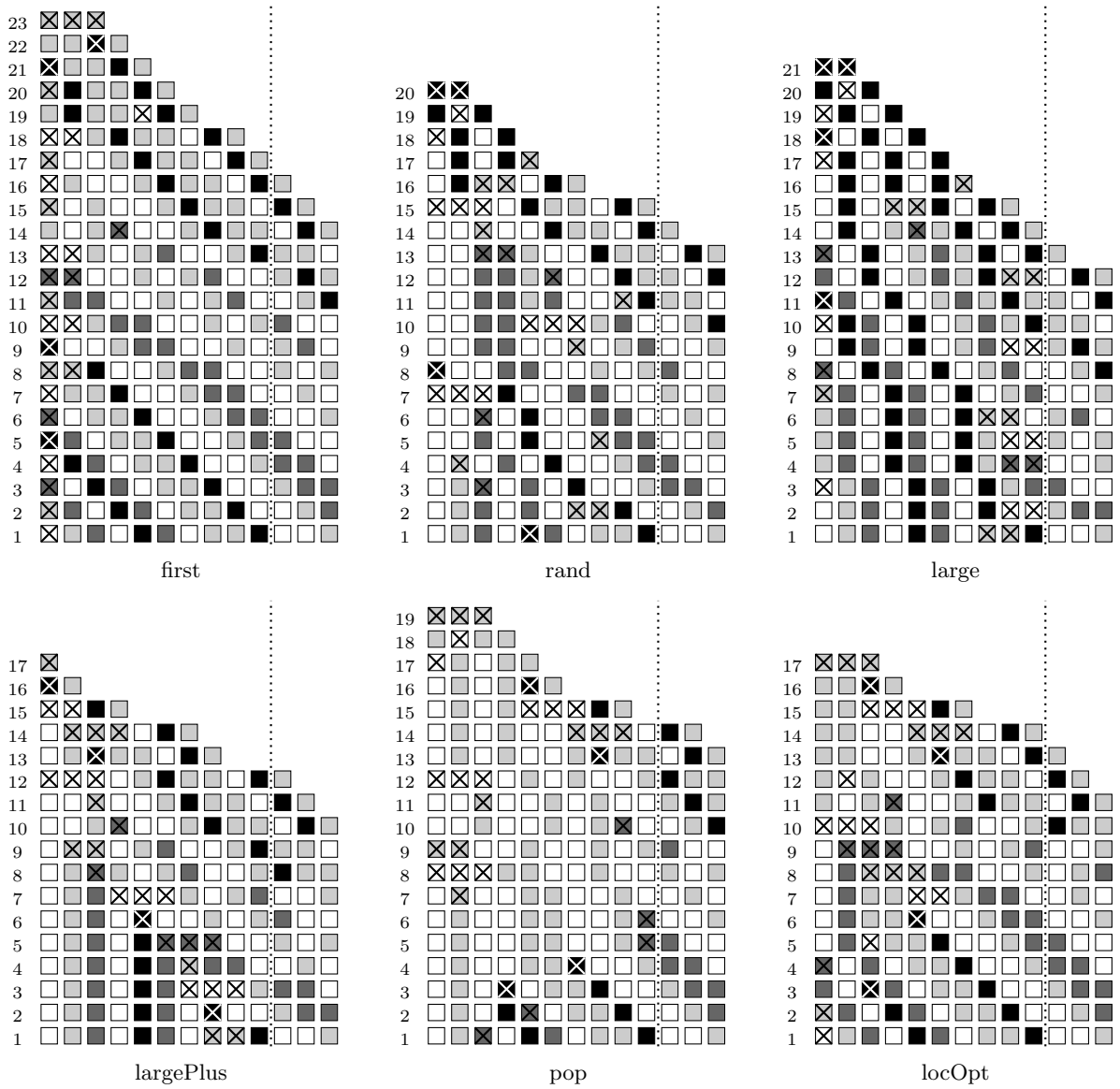

Figure 3: Illustration of the six rules using the example in Figure 2. A run of the online algorithm is depicted as follows. The numbers on the left are counting the moves. Boxes on the left of the dotted vertical line are in the accessible area while boxes on the right are in the visible area. The boxes in the invisible area are not shown. Each move removes the boxes with a black or white cross.

In order to compare the performance of the various rules, we indicate for each of them the average relative percent deviation (ARPD) from the optimal value. More precisely, let E be the set of 900 instances, and for an instance $e \in E$, let b_e be the optimal value produced by the offline algorithm, and let a_e be the number of moves found by the online algorithm. The ARPD is then defined as follows:

$$\text{ARPD} = \frac{100}{|E|} \sum_{e \in E} \frac{a_e - b_e}{b_e}.$$

Table 1 presents these values. The following can be observed. Rule 'first' has the weakest performance with an ARPD of 59.3%. Slightly better is rule 'large' with an ARPD of 51.7%. Rules 'rand' and 'pop' are substantially better than 'first' and 'large' with an ARPD of 31.7% and 30.6%, respectively. The best rules are 'largePlus' and 'locOpt' which have a much lower ARPD of 14.7% and 15.7%, respectively.

Table 1: Comparison of the various rules without visible area.

rule	first	rand	large	largePlus	pop	locOpt
ARPD	59.3%	32.7%	51.7%	14.7%	30.6%	15.7%

These results may be interpreted as follows. Rule 'first' always removes the first block. In order to do better than this trivial strategy, one has to remove a block b_i with a preceding block b_{i-1} and a succeeding one b_{i+1} of the same color. If $|b_{i-1}| + |b_{i+1}| \leq F$ the single block resulting from the merge of these two blocks can then be removed in a single move instead of two. With rule 'large', such a profitable merge rarely happens. Indeed, once the largest blocks have been removed, most of the remaining blocks are of size 1, and the algorithm then often removes the first block (if all of them are of size 1) or the last one (if the boxes that just entered the accessible area have created a block with more than one box). This can clearly be observed in Figure 3 where rule 'large' removes the first or the last block in 16 of the 21 moves.

Profitable merges are more frequent with the 'rand' and 'pop' rules as they occasionally merge two blocks of the same color, by chance. Extending rule 'large' by including a bonus for the merge of two blocks of the same color makes a big difference in terms of performance quality. Indeed, the so obtained 'largePlus' rule performs quite well. On average, its performance is even comparable to the 'locOpt' rule which is executing 'optimal' moves if there were no boxes in the invisible area.

Figure 4 compares the two best rules 'largePlus' and 'locOpt' for different subsets of instances. The following can be observed. The performance of the 'largePlus' rule substantially decreases as the number of colors increases. This result may be explained as follows. In instances with a large number of colors, the probability to be able to merge two blocks with the 'largePlus' rule is generally lower than in instances with fewer colors. In contrast, the performance of the 'locOpt' rule seems independent of the number of colors. For both rules, instances with a uniform distribution of the colors (instance type a) seem to be slightly more difficult than instances of the other types. The performance of the 'locOpt' rule improves as the size of the accessible area increases. It may be argued that the more boxes the 'locOpt' rule sees, the less myopic the chosen moves are. The performance of the 'largePlus' rule seems to be quite independent of the size of the accessible area. For both rules, the instances become more difficult as the fork size increases.

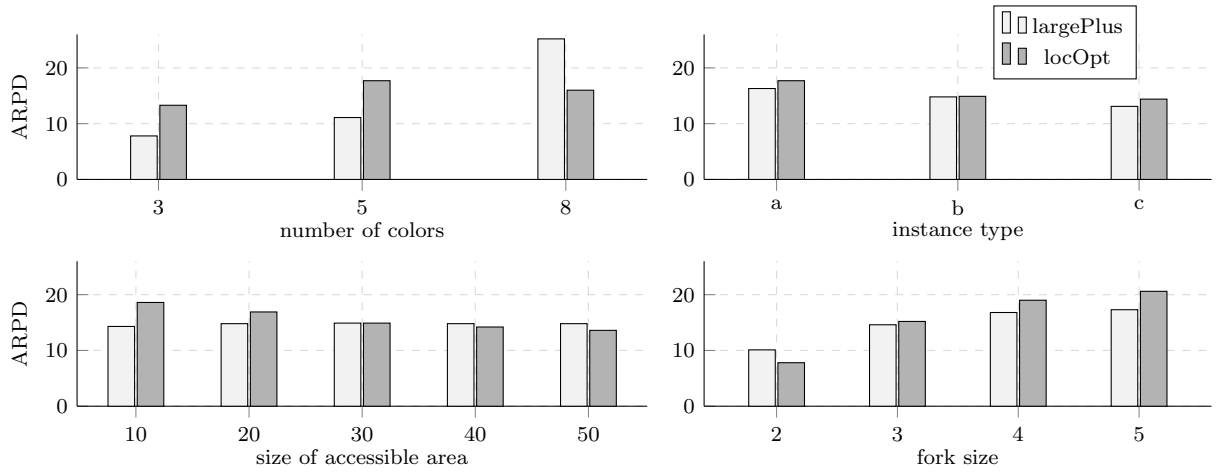

Figure 4: Comparison of the 'largePlus' and 'locOpt' rules for subsets of instances

4.1.2 With a visible area

The only rule that uses information about the boxes in the visible area is the 'locOpt' rule. We now analyze its performance as the size of the visible area increases. For this purpose, we have run the online algorithm with the 'locOpt' rule and with a visible area of size $V \in \{10, 20, 30, 60, 90, 120, 200, 300, 400\}$. Table 2

Table 2: ARPDs for the 'locOpt' rule as size V of the visible area increases.

size V	10	20	30	60	90	120	200	300	400
ARPD	13.5%	12.5%	11.7%	9.8%	8.3%	7.0%	4.3%	1.6%	0.0%

indicates the obtained ARPDs. The following can be observed. Starting with an ARPD of 15.7% with $V = 0$ (see Table 1), the gap decreases as V increases until it is 0 with $V = 400$. Clearly, if all the boxes are in the accessible or in the visible area, then the online algorithm is equal to the optimal offline algorithm. In conclusion, the 'locOpt' rule performs better than the 'largePlus' rule if the visible area consists of about 10 or more boxes.

5 The near rule

Motivated by the quite good performance of the 'largePlus' rule and on the basis of the above discussion, we have developed another rule called 'near' that also tries to merge two blocks of the same color when removing a block, but with a slightly different philosophy. After merging two blocks of the same color, the newly formed block can be removed by a single move only if this new block is not larger than the fork size. Hence, no move is saved if the newly formed block is larger than the fork size, and we take this into account. Also, it is often impossible to merge two blocks by the removal of a single block. In this case, we try to bring closer together two blocks of the same color that are somewhat near to each other in the sequence. We therefore define a distance between blocks of the same color. Finally, if there is a visible area (i.e., $V > 0$), and the color of the first visible box is the same as the color of the last box in the accessible area, then we do not remove the last block of the accessible area since this may lead to an additional required move.

Formally, the 'near' rule can be defined as follows. For an instance (C, A, V, F), let $\mathcal{B}(C, A) = (b_1, \ldots, b_n)$ be the usual ordered set of blocks in the accessible area, and let $\mathcal{B}(C, A + V) = (b'_1, \ldots, b'_r)$ be the similar partition of the set of boxes that lie in the accessible *or* the visible area. If $V > 0$, $|C| > A$, and $c_A = c_{A+1}$, then the color of the first box of the visible area is the same as the color of the last accessible box, and we define $q = n - 1$; otherwise we set $q = n$. We then have $b_i = b'_i$ for $i = 1, \ldots, q$, and we only consider the removal of one of the first q blocks. For the example of Figure 2, we have $\mathcal{B}(C, A) =$ ((1),(2),(3),(4),(5),(6),(7),(8,9),(10)), $\mathcal{B}(C, A + V) =$ ((1),(2),(3),(4),(5),(6),(7),(8,9),(10),(11,12),(13)), and $q = 9$.

For every block b_i, $i \leq q - 1$, we compute a distance d_i and a set I_i of indices as follows: let j be the smallest index strictly larger than i such that $c(b'_j) = c(b_i)$. If j does not exist (there is no such block) or if $|b_i| + |b'_j| > F$ (the merge of b_i with b'_j would create a block larger than the fork size), we set $d_i = \infty$ and $I_i = \emptyset$; otherwise, we set $d_i = j - i - 1$ and $I_i = \{i + 1, \ldots, min\{i + d_i, q\}\}$. In words, every block b_ℓ with $\ell \in I_i$ is candidate to be removed between two blocks b_i and b'_j at distance d_i from one another and of the same color. For the example of Figure 5, we have $d_1 = 2$, $d_2 = 5$, $d_3 = 2$, $d_4 = 2$, $d_5 = 3$, $d_6 = \infty$, $d_7 = 2$, $d_8 = 2$, and $I_1 = \{2, 3\}$, $I_2 = \{3, 4, 5, 6, 7\}$, $I_3 = \{4, 5\}$, $I_4 = \{5, 6\}$, $I_5 = \{6, 7, 8\}$, $I_6 = \emptyset$, $I_7 = \{8, 9\}$, $I_8 = \{9\}$.

If $d_i = \infty$ for all $i = 1, \ldots, q - 1$, the 'near' rule selects block b_1. Otherwise, let $d^{\min} = \min_{i=1}^{q-1} d_i$ and let $I^{\min}$ be the union of the sets I_i with $d_i = d^{\min}$. The 'near' rule chooses a block b_r such that $|b_r| = \max_{i \in I^{\min}} |b_i|$. For the example of Figure 5, we have $d^{\min} = 2$, $I^{\min} = \{2, 3, 4, 5, 6, 8, 9\}$, and b_8 is selected.

We can observe in Figure 5 that, when applied to the example of Figure 2, the 'near' rule finds a solution with 16 moves, which is strictly better than all previous strategies. Note that the first or the last block is removed in only two of the 16 moves. Observe also that in contrast to the 'locOpt' rule, which can be applied in practice only with the help of a computing device, the basic principles of the 'near' rule can be easily applied by human operators. Indeed, the blocks can be observed by eye, and the distances d_i are quite simple to calculate exactly, or at least to estimate. The operator must then choose a block which is between two blocks which are as close as possible to one another and of the same color, with a preference for large blocks.

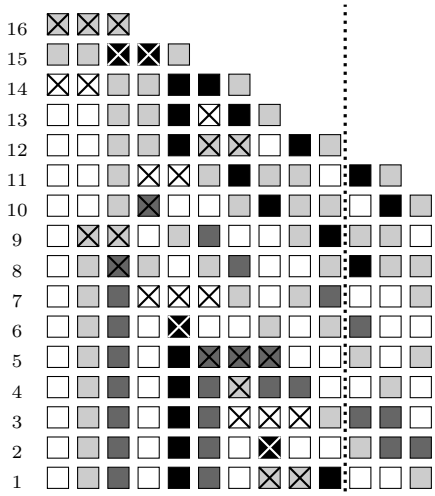

Figure 5: A run of the example with the 'near' rule. The obtained solution with 16 moves is optimal.

5.1 Computational results

In order to evaluate the performance of the 'near' rule, we first consider the 900 instances of Section 4.1 with a visible area of size $V = 0$. Its ARPD is 6.1% which is much better than those of the previous six rules. More details are given Figure 6 where the 'near' rule is compared to the 'largePlus' and 'locOpt' rules for subsets of instances. It can be observed that the 'near' rule clearly outperforms the two other rules, and the results appear as very robust with the various instance characteristics.

Figure 7 further supports these findings by illustrating the empirical cumulative distribution of the relative percent deviations obtained with the 'largePlus', 'locOpt', and 'near' rules. It can be seen that the 'near' rule almost always finds a solution with a relative percent deviation smaller than 10%, while for the 'locOpt' and 'largePlus' rules this is only the case for 22% and 37% of the instances, respectively. Furthermore, while the 'locOpt' and 'largePlus' rules have a relative percent deviation larger than 30% for some instances, this never occurs with the 'near' rule, the maximum being 11%. The standard deviation is 8.8% for the 'largePlus' rule, 6.9% for the 'locOpt' rule, and 1.9% for the 'near' rule.

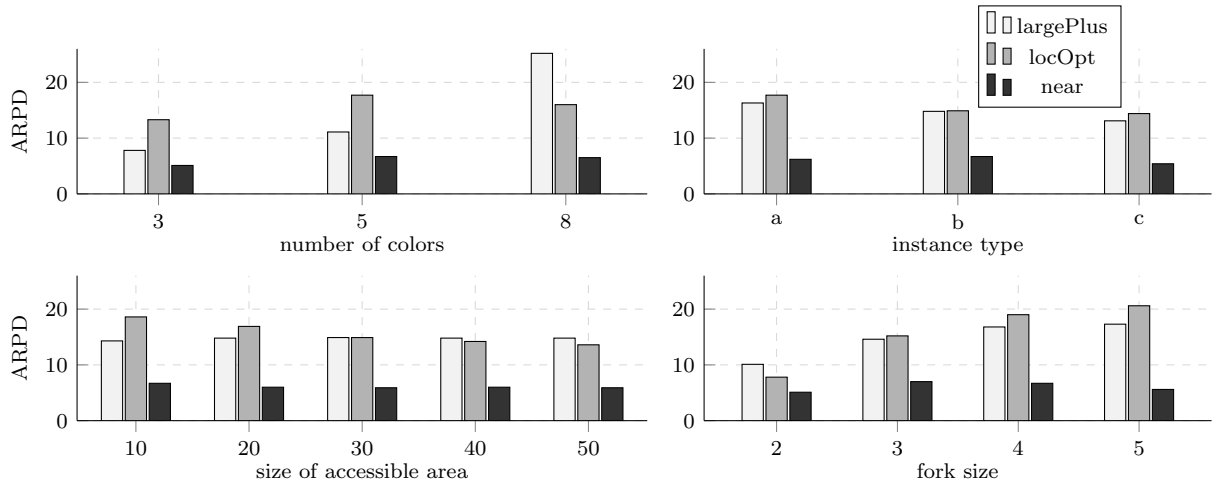

Figure 6: Relative gaps of the 'near', 'largePlus', and 'locOpt' rules averaged over different subsets of instances.

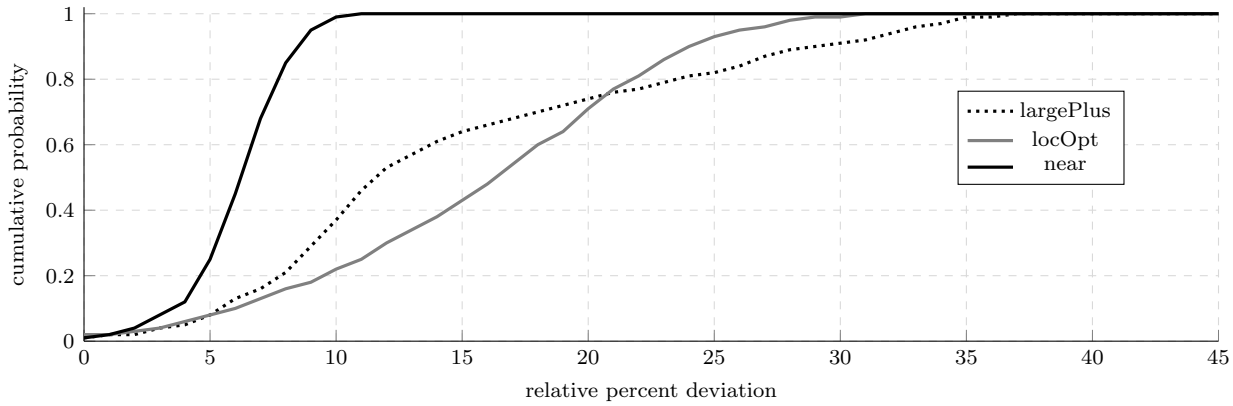

Figure 7: Empirical cumulative distribution of the relative percent deviations for the 'largePlus', 'locOpt', and 'near' rules.

We next report the results obtained with the 'near' rule when the visible area is of size $V \in \{10,20,30,60, 120,200,300,400\}$. The ARPD for the 'near' rule slightly decreases from 6.1% with $V = 0$ to 5.9% with $V = 10$, and stays at 5.9% for $V > 10$. This indicates that the performance of the 'near' rule slightly increases if there is a visible area, the main reason being that the knowledge of the visible area allows to prevent the removal of the last block of the accessible area if other boxes of the same color immediately follow in the visible area. Clearly, the boxes directly following the accessible area have a larger impact on the selection of the move than boxes that are further away, explaining that there are almost no differences between results with small and large visible areas.

Since the performance of the 'locOpt' rule improves when the size of the visible area increases, we can estimate the size of the visible area with which the 'near' and 'locOpt' rules have a similar performance. We have already obeserved in Table 2 that the ARPD of the 'locOpt' rule is equal to 7% with $V = 120$ and to 4.3% with $V = 200$ while, as mentioned above, it is equal to 6.1% for the 'near' rule. A more detailed comparison is provided in Figure 8, where we display the percentage of instances for which the 'near' rule gives worse, equal, or better results than the 'locOpt' rule. Here again, we can observe that the 'near' rule outperforms the 'locOpt' rule for $V \leq 120$, whereas 'locOpt' is better when $V > 200$, concluding that 'locOpt' needs a large visible area in order to meet the performance of the 'near' rule.

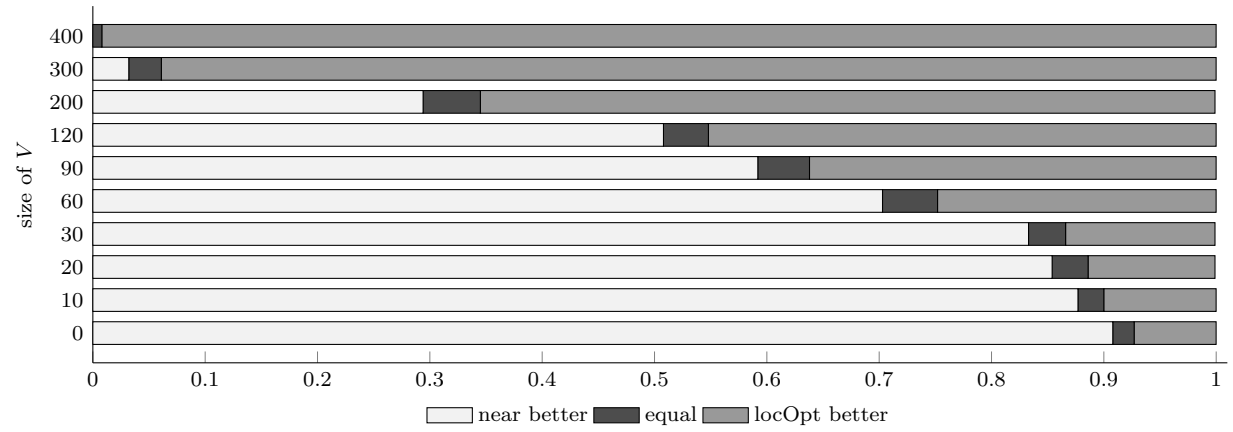

Figure 8: Comparison of the 'near' and 'locOpt' rules as the size of the visible area V increases.

6 Concluding remarks

We addressed an online version of the problem of minimizing the number of moves for unloading boxes off a gravity conveyor. Several online heuristic algorithms were introduced and extensive experimental tests were carried out in order to assess their performance. It turned out that the online algorithm with the 'near' rule performs extremely well. Despite its simplicity and its online character, it generates solutions that are almost optimal, and its performance is robust with respect to different instance characteristics. In addition, its basic principles can be easily applied in practice by human operators.

For further research, it would be interesting to investigate a stochastic version of the present work in which the total number of boxes of each color is known in advance. This information is sometimes available in practice and could be used to improve the results.

References

Baptiste, P., A. Hertz, A. Linhares, and D. Rebaïne. 2013. A polynomial time algorithm for unloading boxes off a gravity conveyor. Discrete Optimization 10(4): 251–262.

Baptiste, P., D. Rebaïne, and Z. Brika. 2011. Chargement de véhicules à l'aide d'un convoyeur. In 9e Congrès International de Génie Industriel, Saint-Saveur, Canada.

Baptiste, P., D. Rebaïne, and Z. Brika. 2012. Chargement de camions d'une ligne de production: comparaison d'heuristiques. In 13 Congrès Annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision, .

Bartholdi, J.J., and L.K. Platzman. 1986. Retrieval strategies for a carousel conveyor. IIE Transactions 18(2): 166–173.

Becker, C., and A. Scholl. 2006. A survey on problems and methods in generalized assembly line balancing. European Journal of Operational Research 168(3): 694–715.

Bozma, H.I., and M.E. Kalalıoğlu. 2012. Multirobot coordination in pick-and-place tasks on a moving conveyor. Robotics and Computer-Integrated Manufacturing 28(4): 530–538.

Ding, J., B.S. Greenberg, and H. Matsuo. 1998. Repetitive testing strategies when the testing process is imperfect. Management Science 44(10): 1367–1378.

Ghosh, J.B., and C.E. Wells. 1992. Optimal retrieval strategies for carousel conveyors. Mathematical and Computer Modelling 16(10): 59–70.

Lodi, A., S. Martello, and D. Vigo. 2002. Recent advances on two-dimensional bin packing problems. Discrete Applied Mathematics 123(1–3): 379–396.

Morabito, R., S.R. Morales, and J.A. Widmer. 2000. Loading optimization of palletized products on trucks. Transportation Research Part E: Logistics and Transportation Review 36(4): 285–296.

Raz, T., and M. Thomas. 1983. A method for sequencing inspection activities subject to errors. IIE Transactions 15(1): 12–18.

Sumichrast, R.T., and R.S. Russell. 1990. Evaluating mixed-model assembly line sequencing heuristics for just-in-time production systems. Journal of Operations Management 9(3): 371–390.