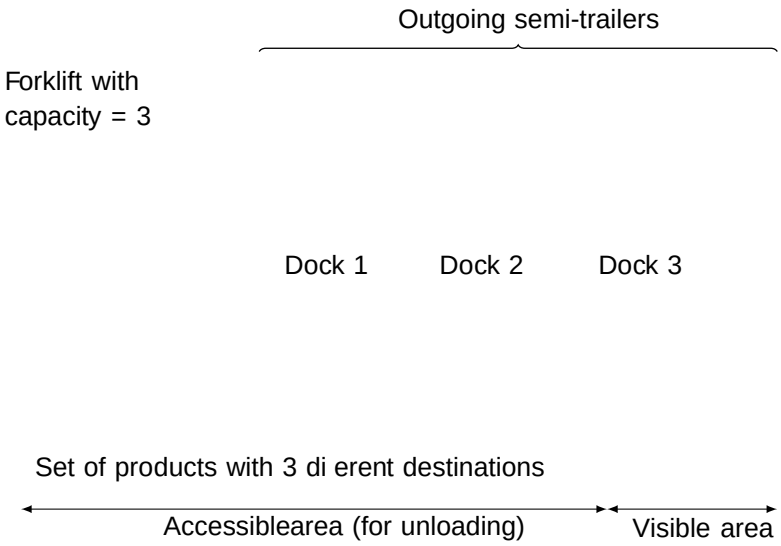

Figure 1. Unloading boxes o a gravity conveyor.

aim at developing strategies that unload all boxes in as few moves as possible. If the travel times between the gravity conveyor and all destinations are the same, then the total travel time is also minimized when minimizing the total number of moves. In practice, the travel times between the gravity conveyor and the di erent destinations are never exactly the same, but the di erences are mostly negligible when compared to the total time needed for one move, which is the sum of the loading, unloading, and travel times.

Clearly, the sequence in which the boxes are placed on the gravity conveyor has a signi cant in uence 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 o 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 speci c items from conveyors. For example, Bozma and Kalal oglu (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 o ine 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 e cient dynamic programming based approach that nds an optimal solution in $O(n^3 A \log F)$ time, where n is the total number of boxes, A the size of the accessible area, and F the forklift capacity. However, this approach cannot be applied in practice since only a part of the boxes is visible, the other boxes being hidden. More precisely, the boxes arrive at the gravity conveyor in a continuous fashion, and their order is not known in advance. This is due to various unknowns inherent to the quality inspection process. As a consequence, online approaches are sought in practice. Moreover, the forklifts and the gravity conveyor typically possess no computing

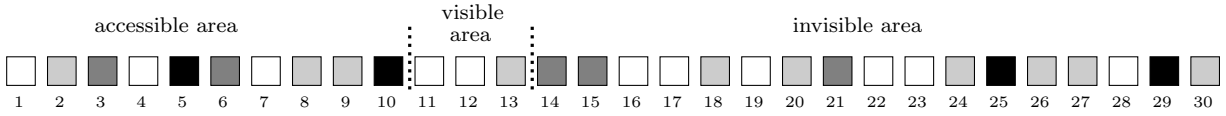

Figure 2. An example with 30 boxes and 4 different destinations indicated with the colors white, light grey, dark grey and black. The numbers below the boxes indicate the position of the boxes.

device. While this might change in the future, there is currently some reluctance in practice to install hardware and develop software to assist the decision-making process. As a by-product, the approach we propose can be applied by human operators without using any computing device.

To the best of our knowledge, the only contribution tackling the online version of our problem is (Baptiste, Rebaine, 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 the present study.

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.

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 sequence 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 $F \cdot \max\{|C|, A\}$. 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

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

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

Definition 2. *A move $(p, \ell) \in \Sigma(C, A, F)$ 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 $C' \sqsubseteq C$. The following property clearly holds.

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 $OPT(C', A, F) \leq OPT(C, A, F)$.*

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 $C \odot (p, \ell) \sqsubseteq C \odot (p^*, \ell^*)$, and it follows from Proposition 3 that $OPT(C \odot (p, \ell), A, F) \leq OPT(C \odot (p^*, \ell^*), A, F)$. 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 in (Baptiste, Rebaine, 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 move (p, ℓ) belongs to $\Sigma^{R}(C, A, F)$ 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 generic 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 $\mathcal{R}$. 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 the leftmost block b_r such that $|b_r| = \max_{i=1}^{n} |b_i|$.

largePlus Choose the leftmost 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 $\lambda_i = |b_{i-1}| + |b_{i+1}|$, else $\lambda_i = 0$. In words, λ_i is a bonus on 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 the leftmost 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 $C' = \{c_1, \ldots, c_m\}$. 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.

Note that the 'large', 'largePlus', and 'pop' rules were introduced in (Baptiste, Rebaine, and Brika 2012).

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. These characteristics were inspired by the instances encountered in practice, which had a forklift capacity of three, about four different destinations, and number of boxes and color distributions as described above.

For each combination of the above parameters, we 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 computed the total number of moves needed to remove all boxes. For each instance we also determined 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). For each instance and rule, we calculated the relative percent deviation (RPD) between the attained number of moves (res) and the optimal value (opt), i.e. $100 \cdot (res - opt)/opt$. We then determined the average RPD (ARPD), minimum, first quartile, median, third quartile, and maximum RPD for each rule. Figure 4 illustrates these values in box plots. 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 32.7% and 30.6%, respectively. The best rules are 'largePlus' and 'locOpt' which have a much lower ARPD of 14.7% and 15.7%, respectively. These findings are also supported by the other statistical measures. Indeed, the 'largePlus' and 'locOpt' rules have substantially lower medium RPDs and lower spreads between the first and third quartiles than the other rules. It can also be observed that the 'largePlus' rule has a significantly lower median RPD than the 'locOpt' rule while having a slightly larger spread. Note also that the results obtained with the 'large', 'largePlus', and 'pop' rules are similar to the values obtained in (Baptiste, Rebaine, and Brika

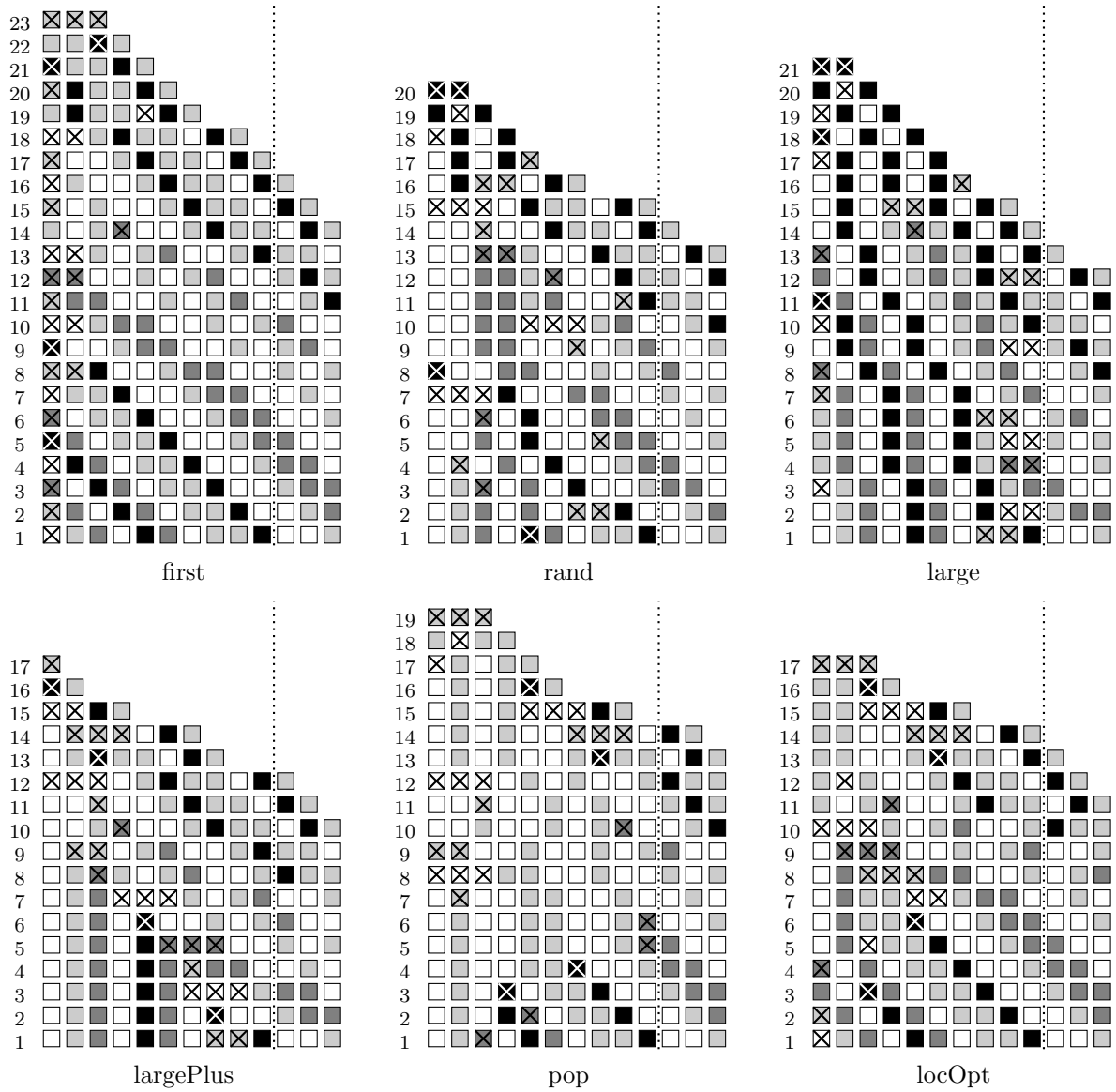

Figure 3. Illustration of the six rules using the example in Figure 2. A run of the online algorithm is depicted as follows. The numbers on the left are counting the moves. Boxes on the left of the dotted vertical line are in the accessible area while boxes on the right are in the visible area. The boxes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

2012), which are the best current benchmarks for our problem.

The attained 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), in which case obviously no profitable merge can happen, or the last one (if the boxes that just entered the accessible area have created a block with more than one box). This behavior can easily be observed in Figure 3 where the 'large' rule removes only five times another block than the first or last one (i.e., in moves 1, 6, 14, 15, and 20), and two of these moves are performed because they remove the last boxes of a color.

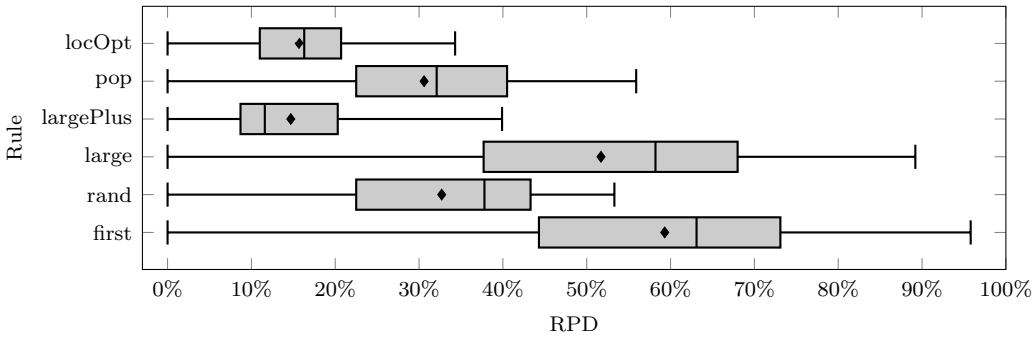

Figure 4. For each rule, a box plot illustrates the minimum, first quartile, median, third quartile, and maximum attained RPD. The ARPDs are represented by diamonds.

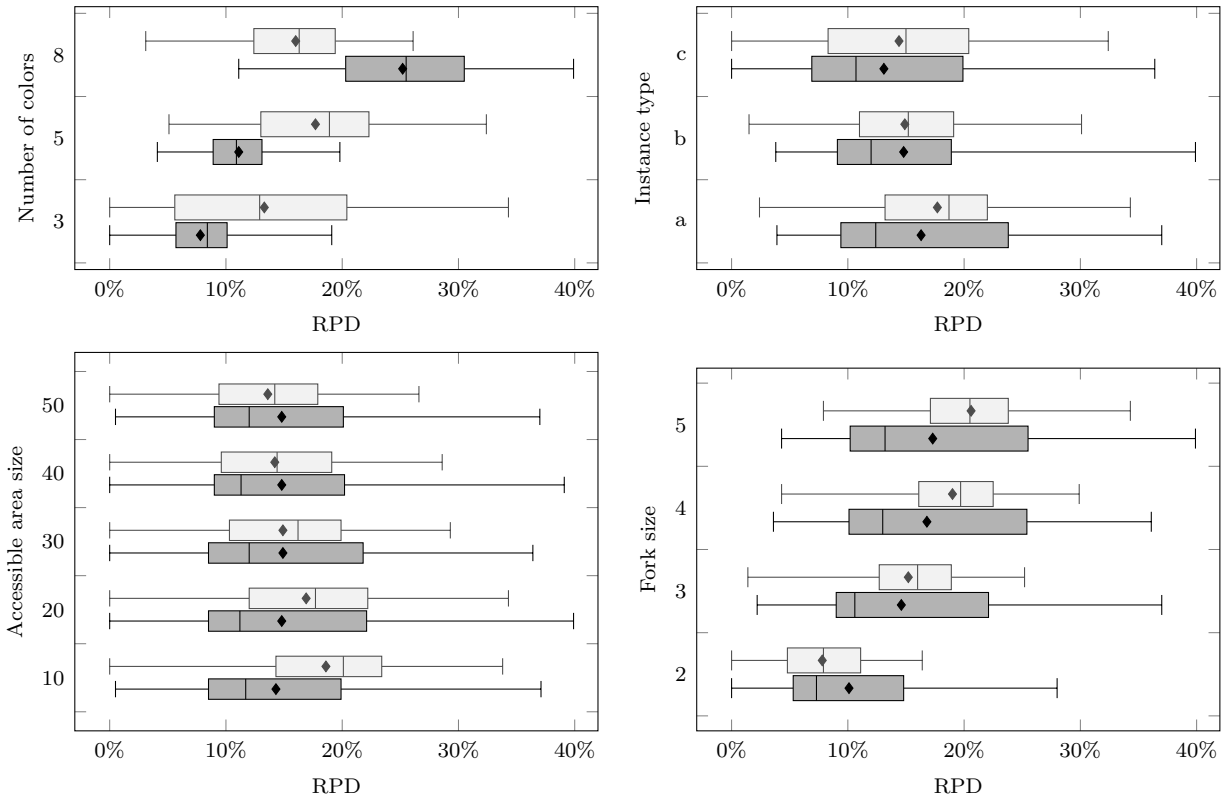

Figure 5. Comparison of the attained RPDs by the 'largePlus' rule (dark grey) and the 'locOpt' rule (light grey) on different subsets of instances.

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 5 compares the two best rules 'largePlus' and 'locOpt' on 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 of merging two blocks with the 'largePlus' rule is generally lower than in instances with fewer colors. In contrast, the performance of the 'locOpt' rule is quite good with a larger number of colors. Observing the box plots of the different instance types, we conclude

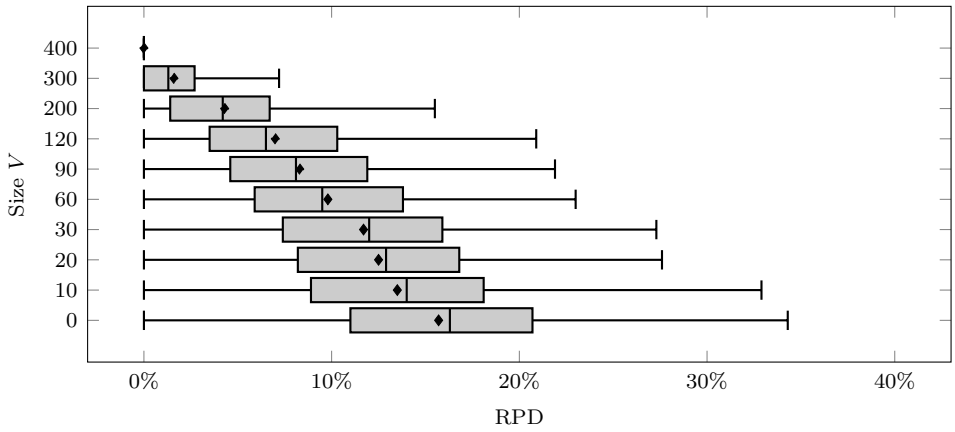

Figure 6. The box plots illustrate the minimum, first quartile, median, third quartile, and maximum RPD attained with the 'locOpt' rule and different sizes of V. The ARPDs are represented by diamonds.

that instances with a uniform distribution of colors (instance type a) are slightly more difficult than instances of the other types for both rules. Furthermore, 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 is quite independent of the size of the accessible area. Finally, the instances become more difficult as the fork size increases for both rules.

4.1.2 With a visible area

The only rule that uses information about 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 ran the 'locOpt' rule and with a visible area of size $V \in \{0, 10, 20, 30, 60, 90, 120, 200, 300, 400\}$. Figure 6 illustrates the obtained results in box plots. The following can be observed.

With $V = 0$, the ARPD is 15.7%, the median RPD is 16.3% and the spread between first and third quartile is 9.7%. These numbers decrease as V increases until they are all 0 at $V = 400$. Clearly, if all the boxes are in the accessible or in the visible area, then the 'locOpt rule always finds an optimal solution. When compared to the 'largePlus' rule (which does not take into account the visible area), the 'locOpt' rule generates lower values if the visible area has 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 developed another rule called 'near', which is also applied within the generic online algorithm. This rule 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.

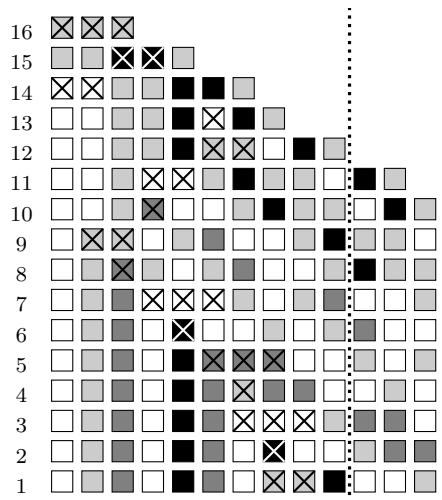

Figure 7. A run of the example with the 'near' rule. The obtained solution with 16 moves is optimal.

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 7, 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 the leftmost block b_r such that $|b_r| = \max_{i \in I^{\min}} |b_i|$. For the example of Figure 7, we have $d^{\min} = 2$, $I^{\min} = \{2, 3, 4, 5, 6, 8, 9\}$, and b_8 is selected.

We observe in Figure 7 that, when applied to the example of Figure 2, the 'near' rule produces an optimal 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 visually, and the distances d_i are quite simple to calculate exactly, or at least to estimate. Roughly speaking, the operator must choose a large block between two blocks of the same color that are as close as possible to each other.

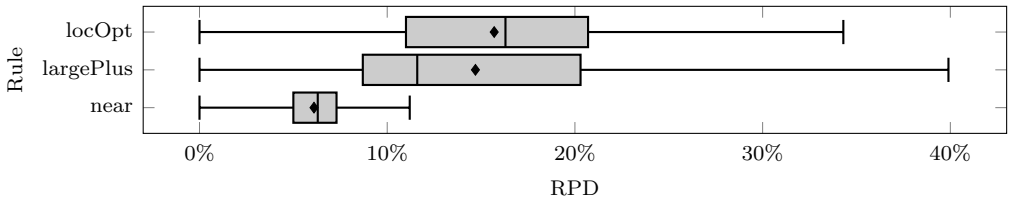

Figure 8. The box plot illustrates the minimum, first quartile, median, third quartile, and maximum RPD attained with the 'near' rule (without a visible area). The ARPD is represented by a diamond. For the purpose of comparison, we reproduce the results obtained with the 'largePlus' and 'locOpt' rules.

5.1 *Computational results*

In order to evaluate the performance of the 'near' rule, we ran the online algorithm with the 'near' rule on the 900 instances of Section 4.1, first setting the visible area to $V = 0$. Figure 8 illustrates the obtained results in a box plot. The following can be observed.

The 'near' rule has an ARPD of 6.1%, a median RPD of 6.3%, and a spread between first and third quartile of only 2.3%. These numbers are considerably lower than those obtained by the previous six rules. More details are given in Figure 9 where the 'near' rule is compared to the 'largePlus' and 'locOpt' rules on different subsets of instances. It can be observed that the 'near' rule clearly outperforms the two other rules. Furthermore, the results obtained with the 'near' rule are robust with respect to different instance characteristics.

Figure 10 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. In addition, 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.

We next report the results obtained with the 'near' rule in Figure 11, when the visible area is of size $V \in \{0,10,20,30,60,120,200,300,400\}$.

When increasing the visible area size from 0 to 10, the ARPD slightly decreases from 6.1% to 5.9%, the median RPD goes from 6.3% to 5.9%, and the spread between first and third quartile decreases from 2.3% to 2.0%. Further increasing the visible area has no effect on the results. These numbers indicate that the performance of the 'near' rule slightly increases if there is a visible area. The main reason is 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 results between small and large visible areas are similar.

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. Observing the results in Figure 6 and 11, we conclude that the 'near' rule outperforms the 'locOpt' rule for $V \leq 120$, whereas 'locOpt' is better if $V > 200$. These findings are supported by Figure 12, which displays the percentage of instances for which the 'near' rule gives worse, equal, or better results than the 'locOpt' rule. In conclusion, the 'locOpt' rule needs a large visible area to meet the performance of the 'near' rule. In practice, however, the size of the visible area is typically quite small.

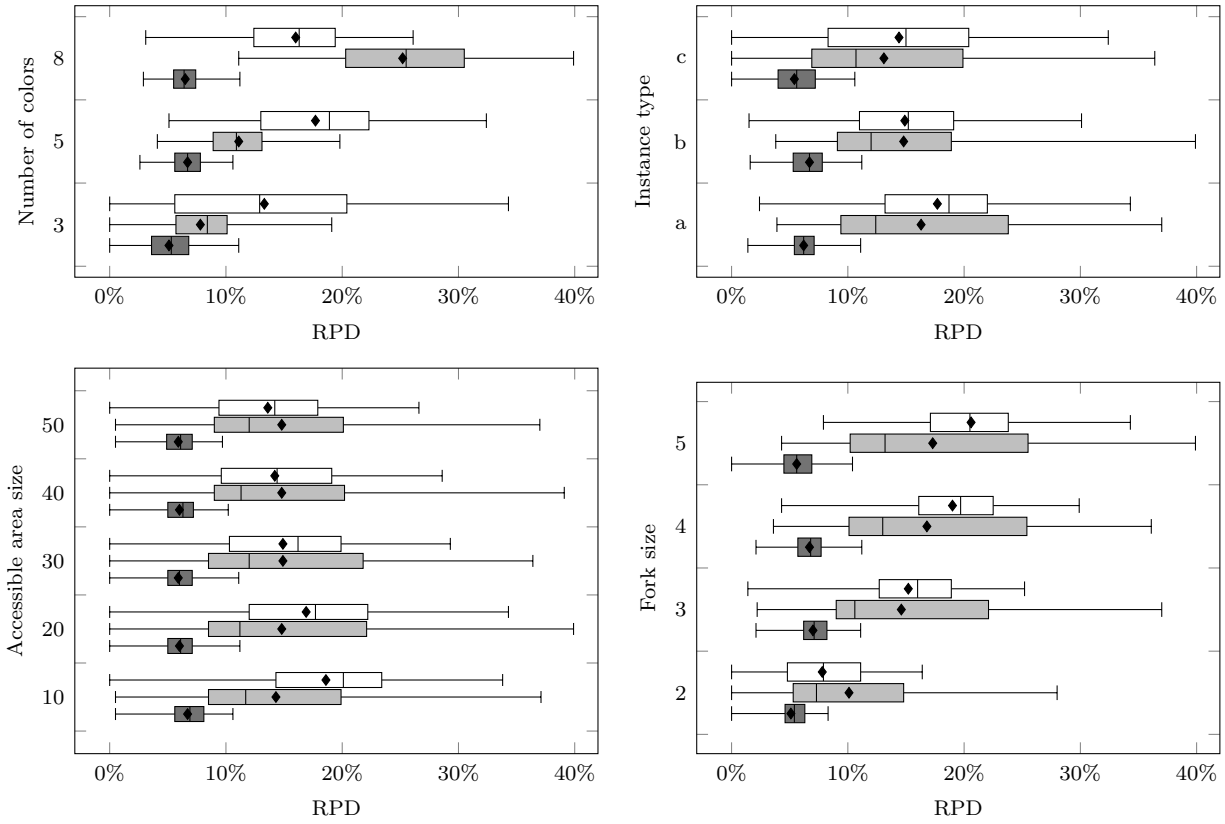

Figure 9. Comparison of the attained RPDs by the 'near' rule (dark grey), the 'largePlus' rule (light grey) and the 'locOpt' rule (white) on different subsets of instances.

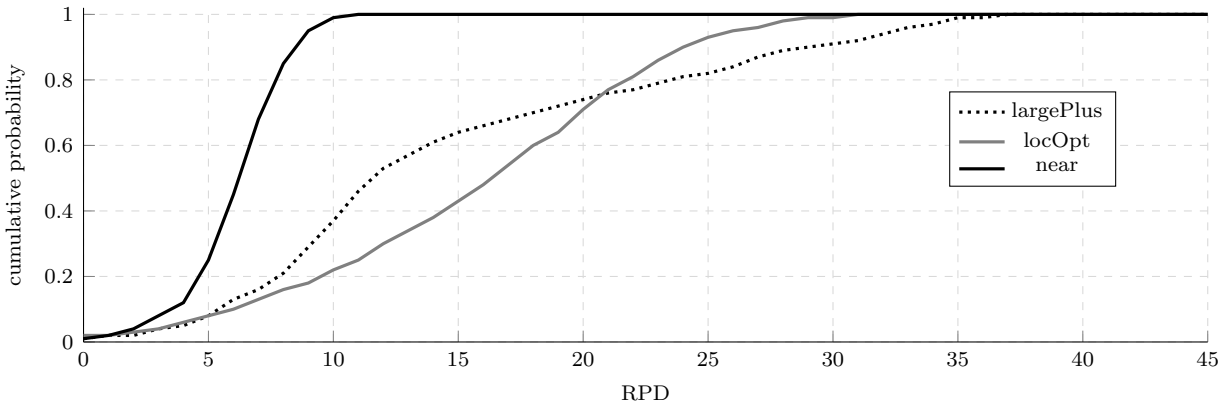

Figure 10. Empirical cumulative distribution of the relative percent deviations for the 'largePlus', 'locOpt', and 'near' rules.

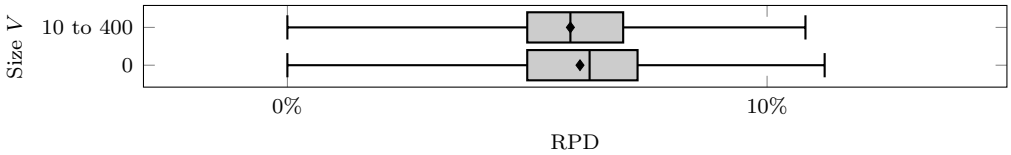

Figure 11. The box plots illustrate the minimum, first quartile, median, third quartile, and maximum RPD attained with the 'near' rule and different sizes of V. The ARPDs are represented by diamonds.

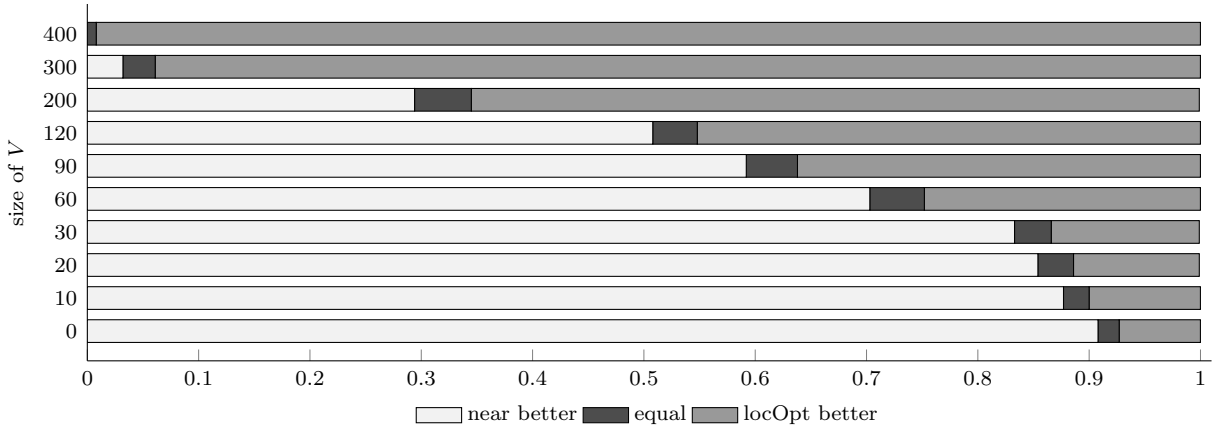

Figure 12. Comparison of the 'near' and 'locOpt' rules as the size of the visible area V increases.

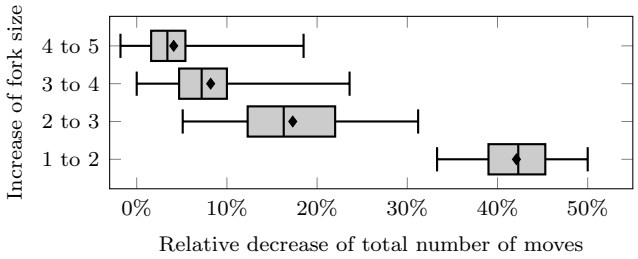

Figure 13. Relative decrease of the number of moves as the fork size increases.

5.2 *On the impact of increasing the fork size*

In a final set of experiments, we estimate the impact of increasing the fork size, which might be of interest when analyzing investments in new forklifts. For this purpose, we considered all instances with fork size 2, and ran the online algorithm with the near rule and increased fork size $F = 3$, 4, and 5. For each instance and fork size, we calculated the relative decrease of the number of moves, i.e. $(r_F - r_{F-1})/r_F$, where r_F is the total number of moves obtained with fork size F. Note that 400 moves are needed for all instances if $F = 1$. Figure 13 displays the obtained values in box plots. The following can be observed.

Increasing the fork size from 1 to 2 has a huge impact. Indeed, the total number of moves decreases by about 42% on average. The effect of increasing the fork size then diminishes. The increase from 2 to 3, 3 to 4, and 4 to 5 generates a relative decrease of the number of moves of about 17%, 8%, and 4%, respectively. Hence, a fork size of 3 or 4 might be adequate in practice.

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 conducted 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. Its performance is also robust with respect to different instance characteristics. In addition, its basic principles can be easily applied in practice by human

operators.

While the order of the hidden boxes is typically not known in advance, the total number of boxes of each color can often be estimated. This additional information could be exploited to improve the results, in particular when evaluating the possible moves with the proposed online algorithm. Indeed, it is often the case that multiple moves have the same quality evaluation, and it might be interesting to simulate several sequences of colors in the invisible area to break ties. We would then choose the move that produces, on average, the best result.

The results of this work may also be used in or stimulate research on similar problems. They arise not only in settings where a conveyor belt transports different types of products to an unloading zone, but also in games such as Clickomania and SameGame (Biedl et al. 2001).

Acknowledgement

We gratefully acknowledge the constructive remarks of three anonymous referees.

Funding

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

References

Baptiste, Pierre, Alain Hertz, André Linhares, and Djamal Rebaine. 2013. "A polynomial time algorithm for unloading boxes off a gravity conveyor." *Discrete Optimization* 10 (4): 251–262.

Baptiste, Pierre, Djamal Rebaine, and Zayneb Brika. 2011. "Chargement de véhicules à l'aide d'un convoyeur (Loading vehicles using a conveyor)." In *9e Congrès International de Génie Industriel,* Saint-Saveur, Canada.

Baptiste, Pierre, Djamal Rebaine, and Zayneb Brika. 2012. "Chargement de camions d'une ligne de production: comparaison d'heuristiques (Truck loading from a production line: comparison of heuristics)." In *13 Congrès Annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision,* .

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

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

Biedl, Therese C., Erik D. Demaine, Martin L. Demaine, Rudolf Fleischer, Lars Jacobsen, and J. Ian Munro. 2001. "The Complexity of Clickomania." In *More Games of No Chance,* edited by Richard Nowakowski, 1st ed., 389–404. Cambridge University Press.

Bozma, H. Işıl, 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, Jie, Betsy S. Greenberg, and Hirofumi Matsuo. 1998. "Repetitive testing strategies when the testing process is imperfect." *Management Science* 44 (10): 1367–1378.

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

Lodi, Andrea, Silvano Martello, and Daniele Vigo. 2002. "Recent advances on two-dimensional bin packing problems." *Discrete Applied Mathematics* 123 (1-3): 379–396.

Morabito, Reinaldo, Silvia Regina Morales, and João Alexandre Widmer. 2000. "Loading optimization of palletized products on trucks." *Transportation Research Part E: Logistics and Transportation Review* 36 (4): 285–296.

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

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