A stochastic online algorithm for unloading boxes from a conveyor line

Reinhard Bürgy · Pierre Baptiste · Alain Hertz · Djamal Rebaine · André Linhares

Published in **Flexible Services and Manufacturing Journal** (online first, July 2017)
The final publication is available at Springer via http://dx.doi.org/10.1007/s10696-017-9291-9.

Abstract This article discusses the problem of unloading a sequence of boxes from a single conveyor line with a minimum number of moves. The problem under study is efficiently solvable with dynamic programming if the complete sequence of boxes is known in advance. In practice, however, the problem typically occurs in a real-time setting where the boxes are simultaneously placed on and picked from the conveyor line. Moreover, a large part of the sequence is often not visible. As a result, only a part of the sequence is known when deciding which boxes to move next.

We develop an online algorithm that evaluates the quality of each possible move with a scenario-based stochastic method. Two versions of the algorithm are analyzed: in one version, the quality of each scenario is measured with an exact method, while a heuristic technique is applied in the second version. We evaluate the performance of the proposed algorithms using extensive computational experiments and establish a simple policy for determining which version to choose for specific problems. Numerical results show that the proposed approach consistently provides high-quality results, and compares favorably with the best known deterministic online algorithms. Indeed, the new approach typically provides results with relative gaps of 1% to 5% to the optimum, which

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

R. Bürgy, P. Baptiste and A. Hertz (corresponding author)
GERAD & École Polytechnique de Montréal, Montréal (QC), Canada H3C 3A7
E-mail: reinhard.burgy@gerad.ca, alain.hertz@gerad.ca, pierre.baptiste@polymtl.ca

D. Rebaine
GERAD & Université du Québec à Chicoutimi, Saguenay (QC), Canada G7H 2B1
E-mail: djamal_rebaine@uqac.ca

A. Linhares
University of Waterloo, Waterloo (ON), Canada N2L 3G1
E-mail: alinhare@uwaterloo.ca

is about 20% to 80% lower than those obtained with the best deterministic approach.

Keywords sequencing · real-time scheduling · heuristics · conveyor line · stochastic online optimization

1 Introduction

The problem studied in this paper was introduced by Baptiste et al. (2012) and arose in a company that manufactures household appliances. It can be described as follows. Different versions of a good are produced in a manufacturing system, where each final product is packed in a standardized box. The boxes are placed on a conveyor line that automatically transports them to an unloading zone. In this zone, the boxes stay on the conveyor until a human operator picks and moves them with a forklift to their predefined destinations, which are, for example, trucks, semi-trailers or shipping containers. Within a move, the operator can only pick contiguous boxes with the same destination, and their number is limited by the fork size. When boxes are removed from the conveyor, the gap that is momentarily created is immediately filled as the boxes are automatically propelled by gravity or by powered rolls.

The conveyor line has three distinct areas: an accessible area where the operator can pick the boxes, a visible area where the operator can see but not pick the boxes, and an invisible area where the remaining boxes are hidden. While the sequence of the boxes in the invisible area is not known in advance, the production quantity is typically known. Consequently, for each destination, the total number of boxes in the invisible area can be deduced or, at least, accurately estimated.

An example, which is used in the sequel, is depicted in Figure 1. There are three destinations, the forklift has a capacity of three, and the conveyor has an accessible area with nine boxes and a visible area with three boxes. For simplicity, we assign to each destination a distinct color and depict each box by the color of its destination.

We consider the problem of determining a box unloading strategy. This is a typical online decision problem as information is revealed incrementally and decisions must be taken without full knowledge. The goal is to minimize the total time needed to remove all boxes. In practice, the travel times between the conveyor and the different destinations are typically similar, or at least, their difference is negligible when compared to the total time needed for one move, which is the sum of the picking, unloading and travel times. Hence, we use the objective of minimizing the total number of moves as a means of minimizing the total time needed to unload all boxes.

Two versions of the problem under study have been discussed so far in the literature. Baptiste et al. (2013) addressed a version of the problem where the complete sequence of the boxes is visible from the beginning. This "offline" version can be seen as a special case of our problem where the visible area

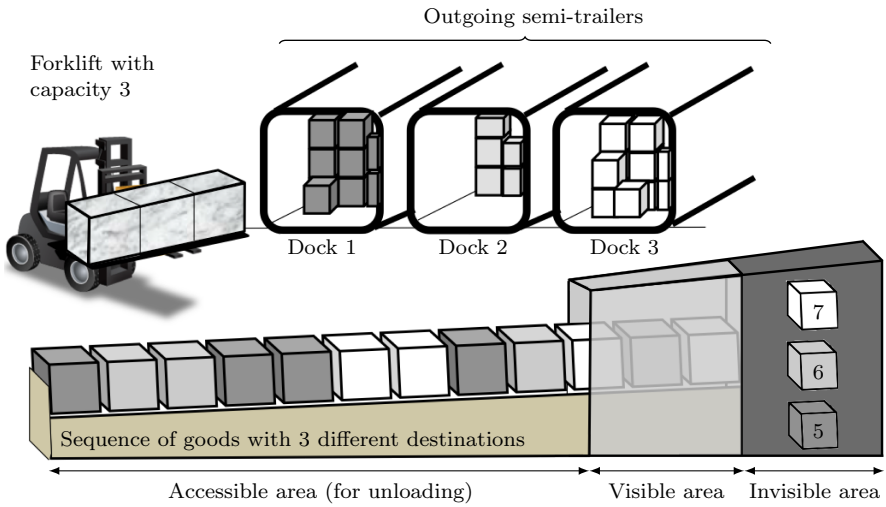

Fig. 1 An example with three different destinations (semi-trailers) indicated with colors white, light gray and dark gray. The conveyor has nine boxes in the accessible area and three boxes in the visible area. The invisible area contains seven white, six light gray, and five dark gray boxes. The Figure is adapted from Figure 1 in (Baptiste et al., 2013).

size is so large that no boxes are in the invisible area. The authors developed a dynamic-programming based approach that finds an optimal solution in $O(n^3 A \log F)$-time, where n denotes the total number of boxes, A is the size of the accessible area, and F is the fork size. This approach is usually not applicable in practice as boxes are simultaneously picked from and newly placed on the conveyor. Hence, the sequence of boxes is built on the fly. Due to various unknowns in the production process, the complete sequence of the boxes cannot be deduced in advance from the production plan.

Baptiste et al. (2012) studied a problem similar to the one addressed in this paper. The only difference is that they assumed that no information is available about the boxes in the invisible area. They proposed three simple heuristics based on intuitive rules for selecting the next boxes to be moved. They observe that the results are typically more than 20% above the optimal value, and conclude that rules of common sense may give an unsatisfactory performance. Based on this study, Baptiste et al. (2017) developed an algorithm that produces results with an average gap of about 6% to 8% to the optimum. Interestingly, the basic principles of their approach can easily be applied in practice by human operators without the need of a computing device.

As the total number of boxes for each destination is typically known in practice, we include this information and ask if and how this additional information can be exploited to improve the online approach discussed above. This research direction was also mentioned in (Baptiste et al., 2017). The main contribution of this article is a scenario-based stochastic move evaluation scheme, which uses the additional information about the boxes in the invisible area to

generate the scenarios. Embedded in an online algorithm, the new evaluation scheme typically finds results with relative gaps of 1% to 5% to the optimum.

While the specific unloading problem under study received little attention in the literature, there is a considerable amount of papers dealing with related problems. Dyer and Glover (1970) discussed the unloading of coal barges at piers. The coal is moved by conveyors from each of the piers, and dumped into a common receptacle. The goal is to find an optimized sequence in which barges are loaded onto the conveyor at the piers. The authors developed a heuristic that provides optimal or near-optimal results. Bozma and Kalalıoğlu (2012) considered a problem where multiple robots pick items from a moving conveyor band, and each robot has a fixed exclusive access area on the band. The goal is to define a strategy maximizing the total number of picked items. Their non-cooperative game-theory based approach provides good results and is easy to integrate in a real-time application. Bartholdi and Platzman (1986) discussed the picking of items from a carousel conveyor that is rotatable in either direction. The goal is to pick all items as fast as possible by optimizing the picking sequence of the orders and the picking sequence of the items within each order. Based on structural results, they developed heuristic solution strategies for different variants of their problem. Similar problems were also addressed in a series of publications, including Ghosh and Wells (1992); van den Berg (1996); Lee and Kuo (2008).

The remainder of this article is organized as follows. The following section provides a formal problem description. A generic online algorithm is defined in Section 3, while Section 4 discusses how to evaluate the possible moves using a scenario-based stochastic method. Two stochastic online algorithms are analyzed in Section 5, and their performance is compared to a deterministic online method. Concluding remarks are given in Section 6.

2 Problem formulation

The conveyor line has three areas, namely an accessible area containing A boxes, followed by a visible area containing V boxes, followed by an invisible area. The forklift has a fork of size F. Let $\mathcal{C}$ be the set of colors and let $c_i \in \mathcal{C}$ be the color of the box at position i. Denote by $C = (c_1, \ldots, c_{|C|})$ the complete sequence of colors on the conveyor line. We identify a box by its position in C, meaning that box i has color c_i. We say that the positions are increasing from *left to right*. An instance is then denoted by (C, A, V, F). Figure 2 illustrates the complete example of Figure 1, which is $(C, 9, 3, 3)$ with C = (3,2,2,3,3,1,1,3,2,1,2,1,2,1,1,1,3,2,3,2,2,3,1,1,3,1,2,3,2,1), where the colors white, light gray, and dark gray are associated with the numbers 1, 2, and 3, respectively.

For decision-making, the sequence of boxes in the invisible is not known. However, for each color $c \in \mathcal{C}$, the number of boxes n_c in this area is known. In the example, these numbers are $n_1 = 7$, $n_2 = 6$, and $n_3 = 5$. The removal

Fig. 2 The example of Figure 1 with the hidden sequence of boxes in the invisible area. The numbers below the boxes indicate their positions.

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) $\ell \leq F$.

The validity conditions state that a) all boxes of the move must be in the accessible area, b) they must have the same color, and c) the number of removed boxes must not exceed F. We denote by $C \odot (p, \ell)$ the sequence of colors resulting 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 discharges all boxes from the conveyor. The objective is to find a strategy with a minimum number of moves. If all boxes are visible from the beginning, the dynamic-programming based algorithm of Baptiste et al. (2013) finds an optimal strategy in polynomial time. We denote by $OPT(C, A, F)$ the optimal value for (C, A, V, F). This value is clearly independent of V.

3 A generic online algorithm

In this section, we first review some properties of the set of moves established in (Baptiste et al., 2017) and then describe a generic online algorithm.

3.1 Valid, dominant and optimal moves

Denote by $\Sigma(C, A, F)$ the set of valid moves for instance (C, A, V, F). Among all moves in $\Sigma(C, A, F)$, we search for a move (p^*, ℓ^*) that minimizes the total number of moves needed to remove all the remaining boxes. Formally, we are looking for a move (p^*, ℓ^*) in $\Sigma(C, A, F)$ such that

$$OPT(C \odot (p^*, \ell^*), A, F) \leq OPT(C \odot (p, \ell), A, F) \;\; \forall (p, \ell) \in \Sigma(C, A, F).$$

Clearly, $OPT(C, A, F) = 1 + OPT(C \odot (p^*, \ell^*), A, F)$. Hence, optimal moves are defined as follows.

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

Some valid moves can be discarded as they are dominated by others. For this purpose, partition the sequence of boxes in the accessible area into maximal subsequences of contiguous boxes with the same color. Denote by $\mathcal{B}(C, A)$ the resulting partition, and call any element $b \in \mathcal{B}(C, A)$ a *block*. In the example of Figure 2, this partition is $\mathcal{B}(C, A) = ((1), (2, 3), (4, 5), (6, 7), (8), (9))$.

For each block $b = (p, \ldots, p + |b| - 1) \in \mathcal{B}(C, A)$, consider the valid move (p, ℓ) with $\ell = \min\{|b|, F\}$, which consists of removing the first ℓ boxes of b. Denote by $\Sigma^{\mathrm{R}}(C, A, F)$ the collection of all such moves. In the example of Figure 2, we get $\Sigma^{\mathrm{R}}(C, A, F) = \{(1, 1), (2, 2), (4, 2), (6, 2), (8, 1), (9, 1)\}$.

Proposition 1 *(Proposition 2 in Baptiste et al., 2017)* $\Sigma^R(C, A, F)$ *contains an optimal move for* (C, A, V, F).

Considering moves in $\Sigma^{\mathrm{R}}(C, A, F)$ rather than in $\Sigma(C, A, F)$ speeds up the search as $\Sigma^{\mathrm{R}}(C, A, F)$ contains at most $\min\{|C|, A\}$ moves, while the size of $\Sigma(C, A, F)$ is bounded by $F(\min\{|C|, A\} - \frac{F-1}{2})$ (which is the number of valid moves if all boxes in the accessible area have the same color).

While it is typically not possible to determine if a move is optimal without knowing the order of the boxes in the invisible area, the two following propositions describe special cases where optimal moves can easily be detected. They state that valid moves that remove F boxes or all remaining boxes of a specific color are optimal.

Proposition 2 *(Proposition 3 in Baptiste et al., 2017) If* $b = (p, \ldots, p + |b| - 1)$ *is a block in* $\mathcal{B}(C, A)$ *with* $|b| \geq F$*, then move* (p, F) *belongs to* $\Sigma^R(C, A, F)$ *and is optimal.*

Proposition 3 *(Proposition 4 in Baptiste et al., 2017) If* (p, ℓ) *is a move in* $\Sigma(C, A, F)$ *such that* $c_i \neq c_p$ *for all* $i \in \{1, \ldots, p - 1\} \cup \{p + \ell, \ldots, |C|\}$*, then move* (p, ℓ) *belongs to* $\Sigma^R(C, A, F)$ *and is optimal.*

The online algorithm in (Baptiste et al., 2017) uses Proposition 3 only if $|C| \leq A + V$ since this is the only case where it can be said that there are no boxes of a given color in the invisible area. The knowledge of the number n_c of boxes of color c in the invisible area makes it possible to also apply Proposition 3 when $|C| > A + V$. Indeed, if there is a block $b = (p, \ldots, p + |b| - 1)$ in $\mathcal{B}(C, A)$ such that $n_{c_p} = 0$ and $c_i \neq c_p$ for all $i \in \{1, \ldots, p - 1\} \cup \{p + |b|, \ldots, A + V\}$, then move (p, ℓ) in $\Sigma^{\mathrm{R}}(C, A, F)$ is optimal. Indeed, Proposition 2 ensures optimality of move (p, ℓ) if $\ell = F$, while Proposition 3 does the same if $\ell = |b|$.

3.2 The generic online algorithm

The proposed generic online algorithm can now be described. If there are no boxes in the invisible area, we apply the optimal algorithm proposed in (Baptiste et al., 2013). Otherwise, we build the set $\mathcal{B}(C, A)$ of blocks and determine the subset $\Sigma^{\mathrm{R}}(C, A, F)$ of valid moves. If one of the two special situations described in Propositions 2 and 3 occurs, we perform the corresponding optimal

move. Otherwise, we compute a score for every move in $\Sigma^{\mathrm{R}}(C, A, F)$, and execute one with the minimum score. The next section describes two ways for measuring the score of a move. The pseudo-code of the proposed generic online algorithm is given in Figure 3, where $f(p, \ell)$ denotes the score of move (p, ℓ).

```
Input: instance (C, A, V, F);
while |C| > A + V do
  Determine B(C, A) and Σ^R(C, A, F);
  if there is (p, ℓ) ∈ Σ^R(C, A, F) with ℓ = F then set (p*, ℓ*) to (p, ℓ);
  else
    if there is b = (p, ..., p + |b| − 1) ∈ B(C, A) such that n_{c_p} = 0 and c_i ≠ c_p
      for all i ∈ {1, ..., p − 1} ∪ {p + |b|, ..., A + V} then set (p*, ℓ*) to (p, ℓ);
    else
      determine the value f(p, ℓ) of every move (p, ℓ) ∈ Σ^R(C, A, F);
      choose (p*, ℓ*) so that f(p*, ℓ*) ≤ f(p, ℓ) for all (p, ℓ) ∈ Σ^R(C, A, F);
    end_if
  end_if
  Perform move (p*, ℓ*) and replace C with C ⊙ (p*, ℓ*);
end_while
Determine the remaining moves with the optimal (offline) algorithm
of Baptiste et al. (2013) and perform them accordingly;
```

Fig. 3 A pseudo-code of the proposed generic online algorithm.

4 Two stochastic evaluation schemes

In order to compute $f(p, \ell)$ for every move (p, ℓ) in $\Sigma^{\mathrm{R}}(C, A, F)$, we consider several sequences of boxes in the invisible area and measure the quality of move (p, ℓ) for each of these sequences, using two different methods. This is explained in more detail in the next subsections.

4.1 Generic evaluation scheme

Let λ be a positive integer parameter whose value will be discussed in Section 5. A *scenario* is a randomly generated sequence of r boxes in the invisible area, with $r = \min\{\lambda, \sum_{c \in \mathcal{C}} n_c\}$. Such a sequence is generated as follows: we first construct a sequence containing n_c boxes of each color $c \in \mathcal{C}$; we then randomly shuffle the $\sum_{c \in \mathcal{C}} n_c$ elements of the sequence, and finally consider the first r elements of the generated sequence. For an instance (C, A, V, F) and a scenario $s_i = (c_1^i, \ldots, c_r^i)$, we consider the sequence $C_i = (c_1, \ldots, c_{A+V}, c_1^i, \ldots, c_r^i)$ of $A+V+r$ boxes, and evaluate the quality of move (p, ℓ) by applying one of the algorithms proposed in Baptiste et al. (2017) on instance $(C_i', A, V+r, F)$, with $C_i' = C_i \odot (p, \ell)$. In other words, we assume that there are exactly r boxes in the invisible area, ordered as in scenario s_i, we consider (p, ℓ) as first move, and we determine how many moves are required to remove the $A+V+r-\ell$ remaining boxes, assuming that all of them are visible, but only the first A boxes are accessible. We consider two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ (detailed in Section 4.2) for solving $(C_i', A, V+r, F)$, and we denote by $m_i^j(p, \ell)$ $(j = 1, 2)$ the number of moves required by $\mathcal{A}_j$ to remove all boxes. This process is applied until a

time limit T is reached. Let σ_j denote the number of scenarios generated when using algorithm $\mathcal{A}_j$. The value $f(p,\ell)$ of move (p,ℓ), associated with algorithm $\mathcal{A}_j$ $(j=1,2)$, is finally set to $\sum_{i=1}^{\sigma_j} m_i^j(p,\ell)$. A pseudo-code of this evaluation scheme is given in Figure 4.

```
1  Input: instance (C,A,V,F), set Σ^R(C,A,F) of moves, parameter λ,
2         algorithm A_j (j=1 or 2);
3  set i to 0 and r to min{λ, Σ_{c∈C} n_c};
4  while time limit T is not reached do
5    set i to i+1;
6    generate scenario s_i = (c_1^i,...,c_r^i) and set C_i = (c_1,...,c_{A+V},c_1^i,...,c_r^i);
7    for_each move (p,ℓ) in Σ^R(C,A,F) do
8      if time limit T is reached then
9        set σ_j to i-1, exit the while loop and goto line 16;
10     else
11       set C'_i = C_i ⊙ (p,ℓ); and determine the number m_i^j(p,ℓ) of moves
12       required by A_j to solve instance (C'_i, A, V+r, F);
13     end_if
14   end_for
15 end_while
16 for_each move (p,ℓ) in Σ^R(C,A,F) do
17   set f(p,ℓ) = Σ_{i=1}^{σ_j} m_i^j(p,ℓ);
18 end_for
```

Fig. 4 Pseudo-code of the stochastic move evaluation scheme.

4.2 Algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$

Baptiste et al. (2017) proposed several rules to unload boxes, assuming that no information is available about the boxes in the invisible area. We briefly describe the two rules that perform substantially better than the others. As shown in Baptiste et al. (2017), these rules produce solutions with an average gap of about 6% to the optimum. Consider an instance (C,A,V,F):

- The locOpt rule considers the subsequence $C'=(c_1,\ldots,c_{A+V})$ of C obtained by removing all boxes in the invisible area. An optimal solution *opt* to instance (C',A,V,F) is determined by applying the exact algorithm of Baptiste et al. (2013), and the move $(p,\ell)\in\Sigma^{\mathrm{R}}(C,A,F)$ chosen by locOpt is then the first move in *opt*.
- The near rule defines a distance between every two blocks in $\mathcal{B}(C,A+V)$ having the same color. It then chooses $(p,\ell)\in\Sigma^{\mathrm{R}}(C,A,F)$ so that box p belongs to a large block that lies between two blocks of the same color close to one another. For more details, the reader is referred to Baptiste et al. (2017).

As mentioned in Section 4.1, the proposed stochastic move evaluation scheme has to solve instances for which all boxes are visible, but only A of them are accessible. Algorithm $\mathcal{A}_1$ solves these instances using the locOpt rule, while $\mathcal{A}_2$ uses the near rule. In what follows, we call StocOpt the online algorithm based on $\mathcal{A}_1$, while StocNear is the one based on $\mathcal{A}_2$. Note that

$m_i^1(p,\ell) = OPT(C_i', A, F)$ since $\mathcal{A}_1$ uses the locOpt rule which is optimal when there is no invisible area.

Instead of choosing a move (p,ℓ) that minimizes $f(p,\ell)$ one could prefer a move (p,ℓ) that maximizes one of the two functions $f'(p,\ell)$ and $f''(p,\ell)$ defined as follows:

$$f'(p,\ell) = \sum_{i=1}^{\sigma_j} (OPT(C_i, A, F) - m_i^j(p,\ell)),$$

$$f''(p,\ell) = \sum_{i=1}^{\sigma_j} (OPT(C_i, A, F) - m_i^j(p,\ell))^2.$$

But this would not make any difference for StocOpt. Indeed, if (p,ℓ) is an optimal move for $(C_i, A, V+r, F)$, then, as observed in Section 3, $OPT(C_i, A, F)) = 1 + OPT(C_i', A, F) = 1 + m_i^1(p,\ell)$, while if (p,ℓ) is not an optimal move for $(C_i, A, V+r, F)$, then $OPT(C_i, A, F)) = OPT(C_i', A, F) = m_i^1(p,\ell)$. Hence, $OPT(C_i, A, F) - m_i^1(p,\ell)$ is 1 or 0, which means that, with StocOpt, $f'(p,\ell) = f''(p,\ell) = \sum_{i=1}^{\sigma_1} OPT(C_i, A, F)) - f(p,\ell)$ is the number of scenarios for which (p,ℓ) is an optimal move, and minimizing $f(p,\ell)$ is then equivalent to maximizing $f'(p,\ell)$ or $f''(p,\ell)$. The situation is different with StocNear since $m_i^2(p,\ell)$ is possibly strictly larger than $OPT(C_i, A, F)$. Since preliminary numerical tests have revealed that the different functions produce similar results with StocNear, we will only consider $f(p,\ell)$ in the sequel.

4.3 An example

The stochastic move evaluation scheme is illustrated with the example of Figure 2. Assume $\lambda = 10$, and let $s_1 = (1,1,1,2,1,2,2,3,1,3)$ be the first scenario. The corresponding sequence C_1 is shown in Figure 5. We have to compute $f(p,\ell)$ for every (p,ℓ) in $\Sigma^{\mathrm{R}}(C, A, F) = \{(1,1),(2,2),(4,2),(6,2),(8,1),(9,1)\}$. Figure 6 illustrates a run of $\mathcal{A}_1$ and $\mathcal{A}_2$ for $(p,\ell) = (2,2)$. Both algorithms solve instance $(C_1', 9, 11, 3)$, where $C_1' = C_1 \odot (2,2)$ has 20 boxes, and is obtained from C_1 by removing the boxes at positions 2 and 3. We see that $m_1(2,2) = 9$ while $m_2(2,2) = 10$. Within a time limit T of one second, StocOpt is able to evaluate 313 scenarios, and the resulting value $f(2,2)$, shown in Table 1, is equal to 2903. For comparison, StocNear is able to evaluate 10636 scenarios and gets $f(2,2) = 103177$. Each move in $\Sigma^{\mathrm{R}}(C, A, F)$ is evaluated in the same way, and we see in Table 1 that both StocOpt and StocNear consider $(6,2)$ as the best move. Note that the large scores of StocNear, when compared to those produced by StocOpt, are due to a larger number of evaluated scenarios.

Finally, the complete sequences of moves chosen by StocOpt and StocNear for the example of Figure 2 are shown in Figure 7. Both versions find an optimal strategy with 14 moves, the optimal number of moves being determined with the exact offline algorithm of Baptiste et al. (2013).

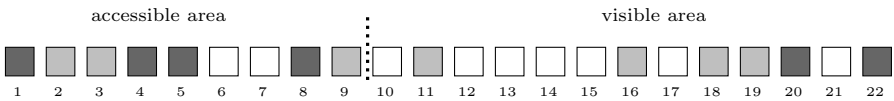

Fig. 5 A first scenario for the example of Figure 2.

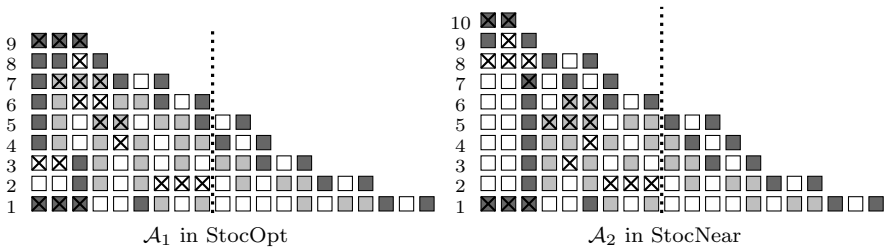

Fig. 6 Evaluation of move $(2, 2)$ for the scenario of Figure 5. The numbers on the left are counting the moves. Boxes on the left of the dotted vertical line are in the accessible area, while those on the right are the visible but not accessible ones. A black cross indicates the boxes picked in each move.

Table 1 Scores of all moves in the first scenario (above) and overall scores (below).

	move (p, ℓ)	$(1, 1)$	$(2, 2)$	$(4, 2)$	$(6, 2)$	$(8, 1)$	$(9, 1)$
$m_1(p, \ell)$	with StocOpt	9	9	9	9	10	10
$m_1(p, \ell)$	with StocNear	10	10	11	9	10	10
$f(p, \ell)$	with StocOpt	2966	2903	2987	2842	2969	2980
$f(p, \ell)$	with StocNear	103952	103177	107686	98712	107932	107986

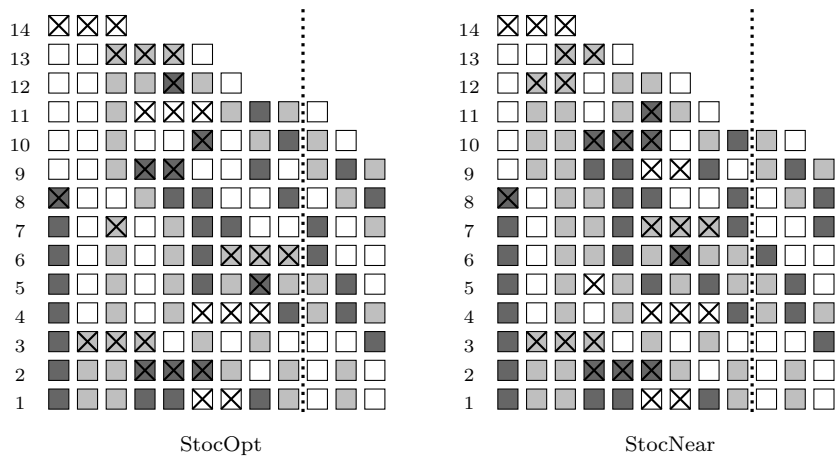

Fig. 7 A run of StocOpt and StocNear for the example of Figure 2. The numbers on the left are counting the moves. Boxes on the left of the dotted vertical line are in the accessible area, while those on the right are the visible but not accessible ones. The invisible area is not shown. A black cross indicates the boxes picked in each move.

5 Computational results

In this section, we assess the quality of StocOpt and StocNear by conducting extensive computational tests. Both versions of the stochastic online algorithm were implemented in Java and run on a PC with 3.1 GHz Intel Core i5-2400 processor and 4 GB memory. The performance of the algorithms is evaluated on a set of instances introduced by Baptiste et al. (2013). The instance characteristics, which are inspired by instances encountered in practice, are as follows. Each instance consists of 400 boxes of five different colors, with 80 boxes of each color. The size A of the accessible area belongs to $\{10, 20, 30, 40, 50\}$, the size V of the visible area belongs to $\{0, 20\}$, and the size F of the fork belongs to $\{2, 3, 4, 5\}$. For each combination of parameters A, V, and F, five sequences of colors were randomly generated, giving a total of $5 \times 2 \times 4 \times 5 = 200$ instances.

Parameters λ and T have a big impact on the performance of the algorithms. Clearly, small values for λ give fast scenarios to evaluate, while large values for λ give a less myopic approach. Also, large values for T allow to evaluate a large number of scenarios, which makes the algorithms more robust. We report results for a time limit T (for each move evaluation) in $\{1, 10, 60\}$ seconds, and a scenario length λ in $\{10, 20, 40, 80\}$, which gives a total of $200 \times 3 \times 4 = 2400$ runs for each version of the stochastic online algorithm.

In order to assess the quality of the results, for each run we compute the relative percent deviation (RPD) between the attained number of moves (res) and the optimal value (opt) obtained by applying the exact algorithm of Baptiste et al. (2013), assuming that the 400 boxes are visible, but only A of them are accessible. Hence, the RPD is $100\frac{res-opt}{opt}$.

5.1 StocOpt versus StocNear

Figure 8 shows the results obtained with StocOpt and a visible area of size $V = 0$. For each combination of A and T, a diagram depicts the statistical distribution of the RPDs for the four tested values of λ. Each box-and-whisker plot shows the average RPD (ARPD) represented with a diamond, as well as the minimum, first quartile, median, third quartile and maximum RPD, using standard conventions for drawing such plots. The color of a box-and-whisker plot indicates the average number of computed scenarios for each move evaluation: if, on average, less than three scenarios are computed, we show no box-and-whisker plot, and we use very dark gray, dark gray, light gray, and white for an average number of scenarios in $[3, 10)$, $[10, 40)$, $[40, 90)$, and $[90, \infty)$, respectively. The box plots are cropped at 10%, higher values not being displayed. Figure 9 depicts the box-and-whisker plots for StocOpt with a visible area of size $V = 20$.

These two figures clearly indicate that the number of scenarios has a big impact on the quality of the results. The stochastic aspect of the approach is not exploited enough if the number of scenarios is too low. For example, the box-and-whisker plots for $A = 40$ and $T = 10$ show that the ARPD is 2.7

with $\lambda = 10$, 2.8 with $\lambda = 20$, 4.2 with $\lambda = 40$, and 13.7 with $\lambda = 80$. The corresponding average numbers of scenarios are 31, 18, 9, and 4. A similar drop of the quality can be observed in other plots. It appears that at least 10 scenarios are needed to get accurate results, all very dark grey plots having a very high ARPD, compared to the other plots with a lighter color.

Also, increasing λ when enough scenarios can be evaluated seems to have a minor effect on the quality of the solutions. This is especially evident when $T = 60$ seconds. Indeed, good results are found with $\lambda = 10$, and larger values do not significantly improve the results. Also, an increase in λ causes a decrease of the number of scenarios, which may outweigh the advantage of having larger scenarios.

The figures suggest that λ can be increased, as long as the total number of scenarios stays above 90 (white box-and-whisker plots). We therefore suggest the following simple policy for choosing λ: if less than 90 scenarios can be evaluated with $\lambda = 10$, then set λ to 10; otherwise set λ to the largest value between 10 and 80 so that at least 90 scenarios can be calculated on average. The chosen values are indicated in Figures 8 and 9 with an asterisk (*) for each V, T, and A. They correspond to the highest white box-and-whisker plot, if any, or to $\lambda = 10$ if no plot is white. In practice, an optimized scenario length λ can be estimated by simulating typical instances.

Figures 10 and 11 provide a similar analysis for StocNear with $V = 0$ and $V = 20$, respectively. We observe that StocNear is very fast since almost all plots are white, even with $T = 1$ second, which means that more than 90 scenarios are evaluated for each move. The scenario length seems to have a minor effect on the quality. Indeed, the results with $\lambda = 10$ are comparable to those obtained with $\lambda = 80$. We suggest to set λ to 40 for all sizes A and V, and for all time limits T.

Figures 12 and 13 and Table 2 compare StocOpt with StocNear, with the aforementioned values for parameter λ. We observe that with no visible area (i.e., $V = 0$), both algorithms provide results with a similar quality for the instances with $A = 10$. While the results are still similar for $A = 20$ and $T \leq 10$, StocOpt produces slightly better results with $T = 60$. Also, StocOpt is substantially better than StocNear for instances with $A \geq 30$ if StocOpt is able to compute at least 10 scenarios.

In the instances with $V = 20$ (see Figure 13 and Table 2), StocOpt provides substantially better results for all sizes A of the accessible area, provided that it computes at least 10 scenarios, on average. This means that StocOpt better utilizes the information about the sequence of boxes in the visible area, when compared to StocNear. This can be explained by the fact that an increase of the size of the visible area allows StocOpt to better estimate the remaining number of moves, while StocNear cannot profit in the same manner from this additional information.

In summary, StocOpt should be preferred to StocNear if at least ten scenarios can be calculated for each move evaluation. The chosen version is indicated with an asterisk (*) for each A, V, and T in Figures 12 and 13.

5.2 Stochastic versus deterministic

In this section, we compare the deterministic online approach described in (Baptiste et al., 2017) with the proposed stochastic algorithms. The main difference between the two approaches is that the deterministic one does not take into account the information available about the boxes in the invisible area. In other words, we evaluate in this section to what extent StocOpt and StocNear are able to exploit the information about the numbers n_c of hidden boxes of color $c \in \mathcal{C}$. As shown in (Baptiste et al., 2017), the near rule dominates the locOpt rule for $V = 0$ and $V = 20$. It is therefore compared to the proposed stochastic approach.

The comparison appears in Figure 14 and Table 2. For each value of V and A, we show the box-and-whisker plots of the RPDs for the deterministic approach (det), and for the stochastic approach with a time limit T of 1, 10, and 60 seconds. Letter "O" indicates that StocOpt is preferred to StocNear (as explained in the previous section), while letter "N" indicates the opposite.

It clearly appears that the stochastic approach provides substantially better results than the deterministic one in all considered configurations. For the instances with $V = 0$, the performance difference increases as the size A increases. Indeed, with $A = 10$, the ARPD decreases from 7.28% (with det) to 5.74% (with $T = 1$), which is an absolute difference of 1.54% and a relative decrease of 21.2%. With $A = 50$ the ARPD decreases from 6.45% to 3.83%, which is an absolute difference of 2.62% and a relative decrease of 40.6%.

A larger time limit is especially useful for instances with a larger accessible area size. Indeed, while the results obtained with a time limit of 1 and 60 seconds are similar in instances with $A = 10$, the difference is substantial in instances with $A = 50$, where the ARPD decreases from 3.83 (with $T = 1$) to 1.88 (with $T = 60$), which is an absolute difference of 1.95% and a relative decrease of 50.9%. This finding intuitively makes sense. Indeed, the set of possible moves increases with A, and there is therefore more room for improvements with larger values of A. Also, each scenario evaluation takes more time with a large A value, hence only few scenarios can be computed with a short time limit.

For instances with $V = 20$ and $A = 10$, the ARPD decreases from 6.38% (with det) to 1.29% (with $T = 1$). This is an absolute difference of 5.09% and a relative decrease of 79.8%. A larger time limit does not lead to further improvements in this case. Similar improvement performances are obtained for instances with larger A, but the time needed to achieve these improvements is higher. For example, with $A = 50$, the stochastic approach needs 60 seconds to get an ARPD of 1.73%, which is to be compared with the ARPD of 6.32% obtained with the deterministic approach. These results suggest that the stochastic approach provides near-optimal results for all instances if enough computation time is available.

Altogether, it can be observed that the stochastic approach performs well across all tested instances and substantially improves the results provided by the deterministic approach of Baptiste et al. (2017).

6 Concluding remarks

We addressed the problem of unloading boxes from a conveyor line, where, in contrast to the version studied in Baptiste et al. (2017), we included information about the number of boxes of each color in the invisible area. Two versions of a stochastic online approach were proposed: in one version (StocOpt), an optimal algorithm is applied to evaluate the quality of each valid move, while a heuristic is applied in the second version (StocNear). Extensive experimental tests have clearly shown that StocOpt provides better results if enough scenarios can be computed, while StocNear should be used in the other case. Altogether, the stochastic approach provides substantially better results than those obtained with the deterministic online algorithm of Baptiste et al. (2017). While improvements are obtained for all studied configurations of the problem, the stochastic approach is particularly efficient if the accessible area size is large, or if there is a visible area.

The following extension of the problem may be studied in future research. In practice, multiple conveyor lines, human operators, and forklifts can be available for moving the boxes to their destinations. In addition to the problem of sequencing the unloading of boxes, one then typically has to choose a conveyor line for each box, which is an additional dimension in the optimization problem. Also, the overall optimization criterion might still be minimizing the total time needed to unload all boxes. However, this objective may not be attained by minimizing the number of moves. Balancing the operator workload over time while keeping the total number of moves low might be a good means to minimize the total time needed for all moves.

The basic principle of the stochastic online approach may also be useful for solving other picking, loading or unloading problems. An interesting problem is, for example, the order picking strategy in an automated carousel. In an online setting, the orders arrive over time and are not known in advance. While the online problem better reflects realistic scenarios, it received little attention in the literature compared to the offline version. An application of our stochastic online approach using already established exact or heuristic offline algorithms to evaluate the possible next picking step may lead to good strategies.

References

Baptiste P, Rebaine D, Brika Z (2012) Chargement de camions d'une ligne de production: comparaison d'heuristiques. In: Conférence ROADEF 2012

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

Baptiste P, Bürgy R, Hertz A, Rebaine D (2017) Online heuristics for unloading boxes off a gravity conveyor. International Journal of Production Research 55(11):3046–3057

Bartholdi JJ, Platzman LK (1986) Retrieval strategies for a carousel conveyor. IIE Transactions 18(2):166–173

van den Berg JP (1996) Multiple order pick sequencing in a carousel system: a solvable case of the rural postman problem. Journal of the Operational Research Society 47(12):1504–1515

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

Dyer JS, Glover F (1970) A barge sequencing heuristic. Transportation Science 4(3):281–292

Ghosh JB, Wells CE (1992) Optimal retrieval strategies for carousel conveyors. Mathematical and Computer Modelling 16(10):59–70

Lee SD, Kuo YC (2008) Exact and inexact solution procedures for the order picking in an automated carousal conveyor. International Journal of Production Research 46(16):4619–4636

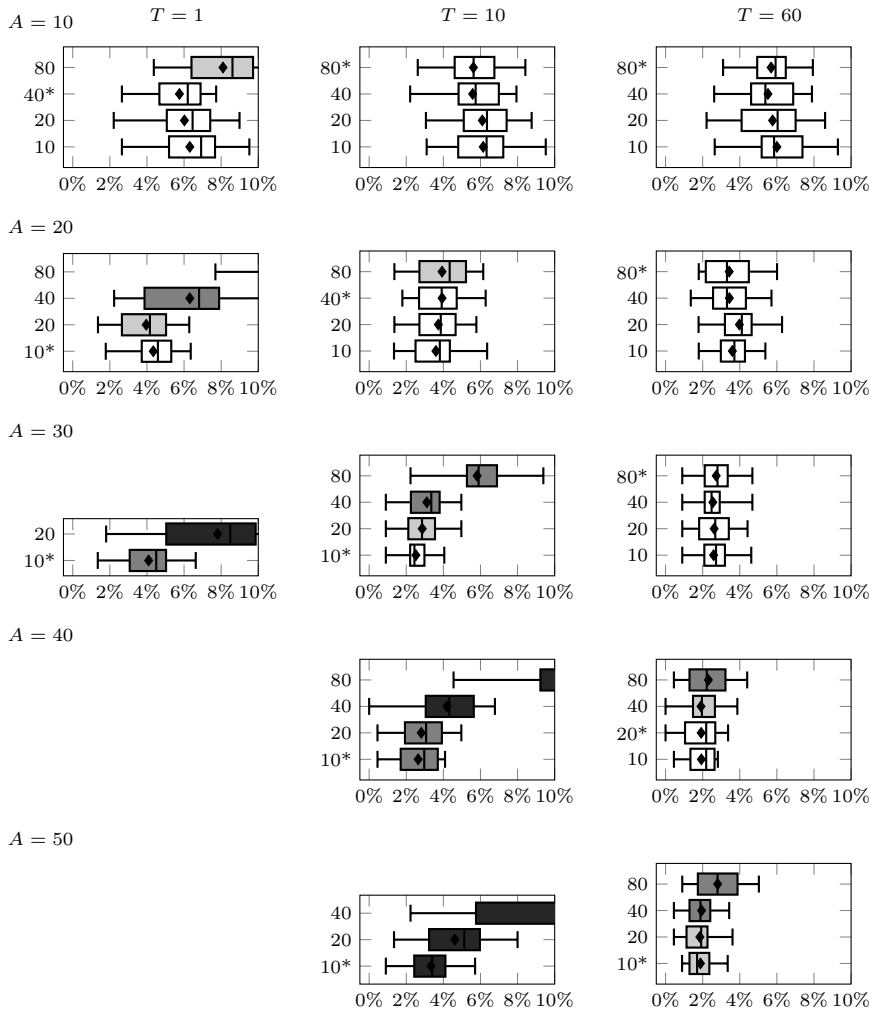

Fig. 8 Analysis of StocOpt with a visibility area of size $V = 0$. The suggested scenario length λ is shown with an asterisk.

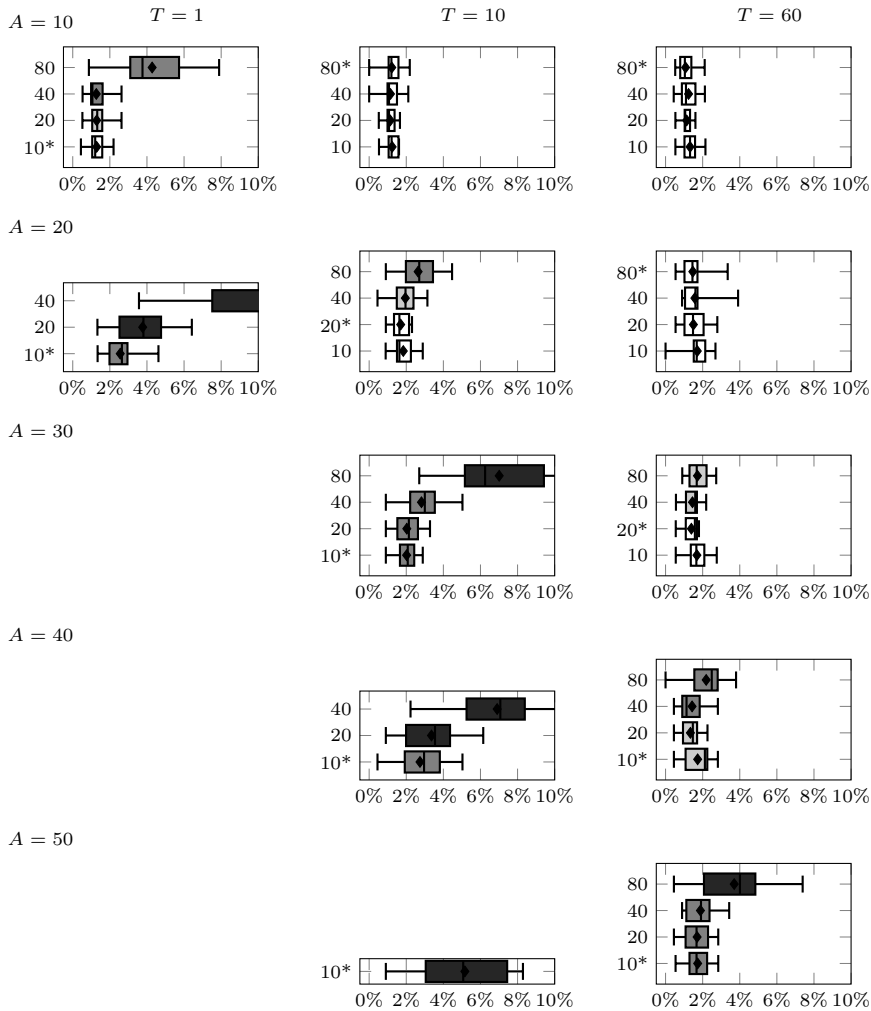

Fig. 9 Analysis of StocOpt with a visibility area of size $V = 20$. The suggested scenario length λ is shown with an asterisk.

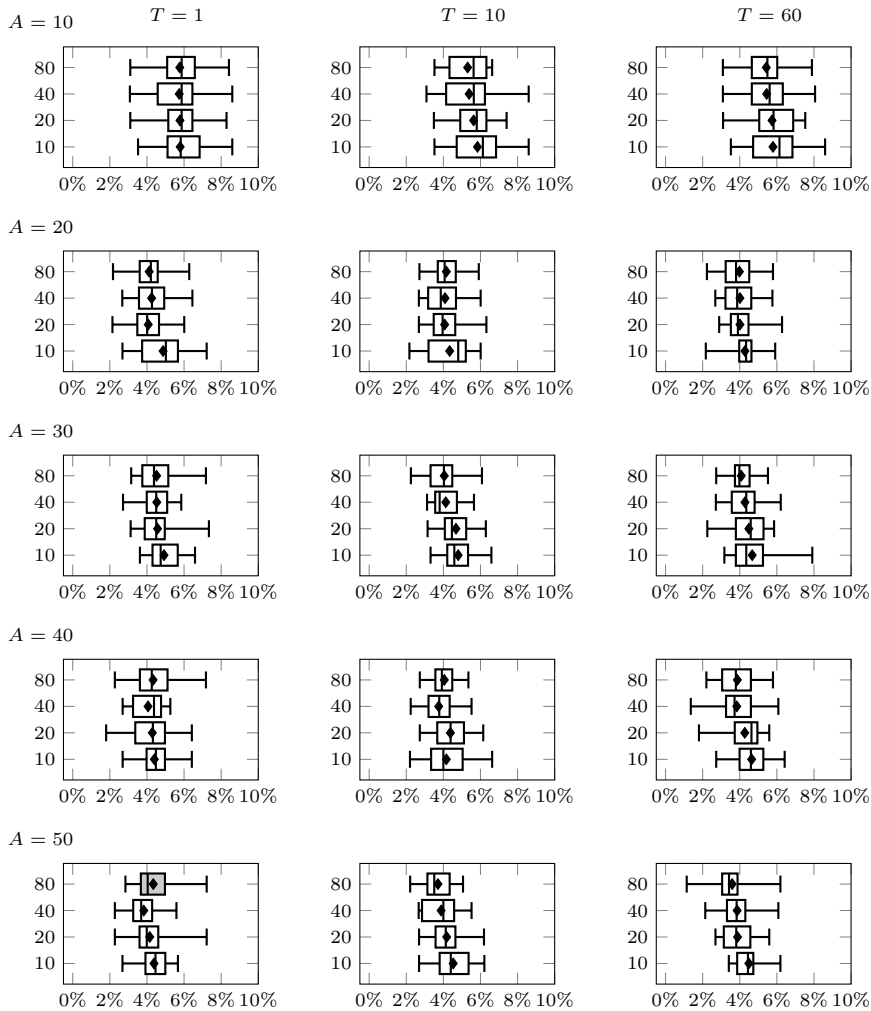

Fig. 10 Analysis of StocNear with a visibility area of size $V = 0$.

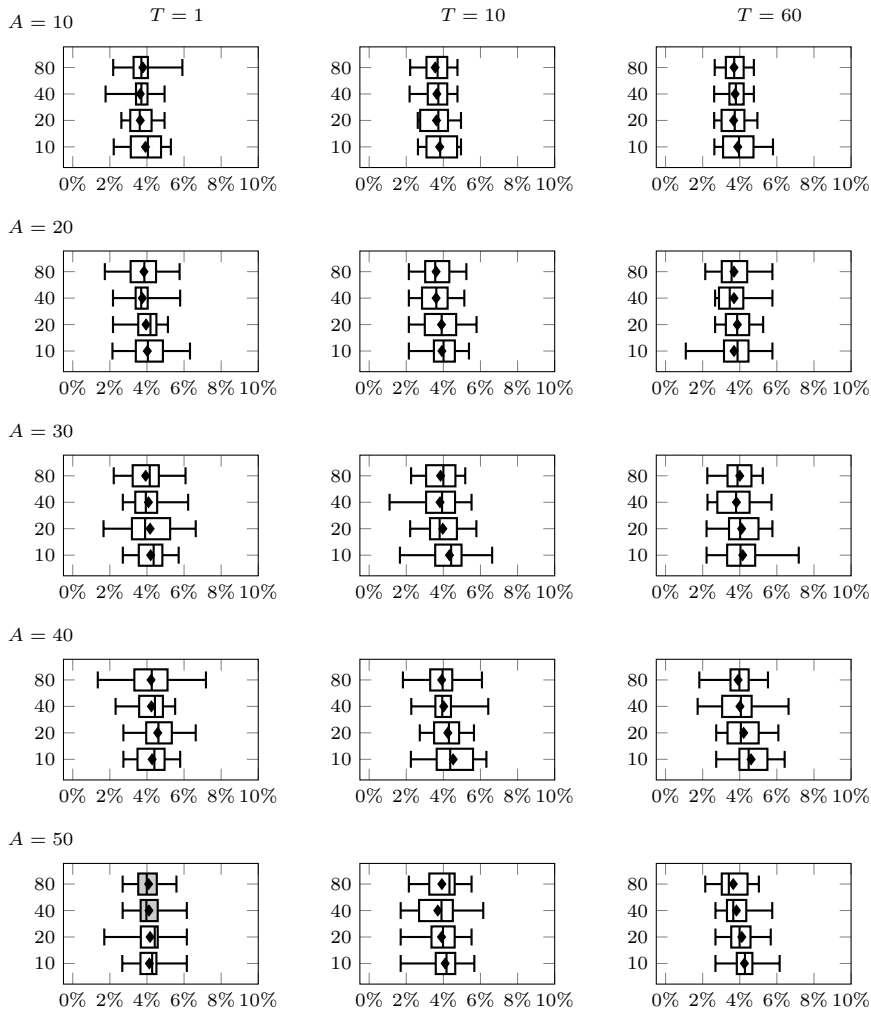

Fig. 11 Analysis of StocNear with a visibility area of size $V = 20$.

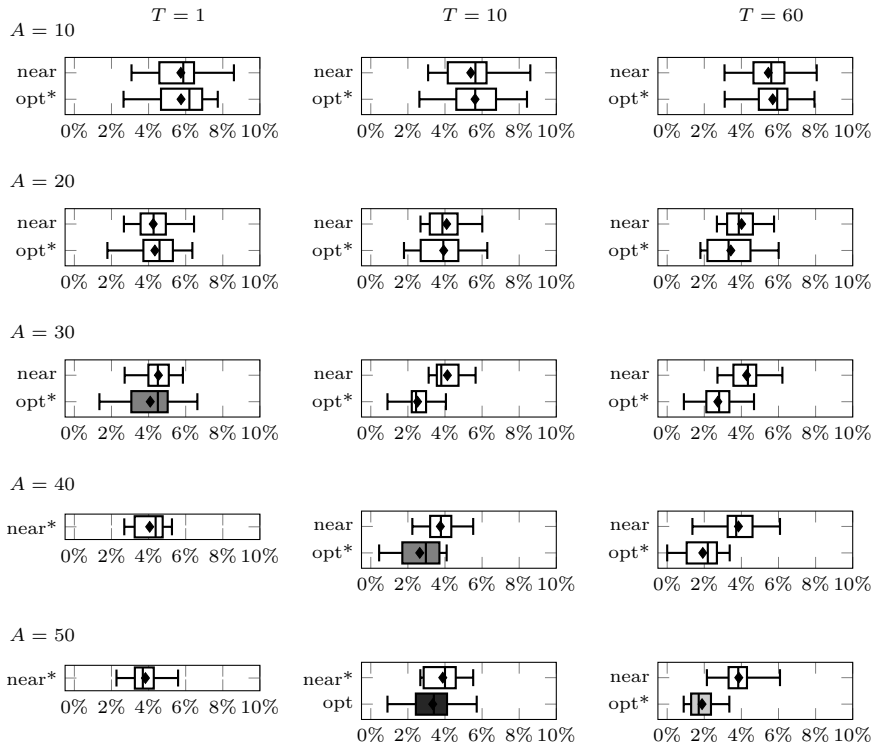

Fig. 12 Comparison of StocOpt with StocNear for a visibility area of size $V = 0$. The suggested algorithm is shown with an asterisk.

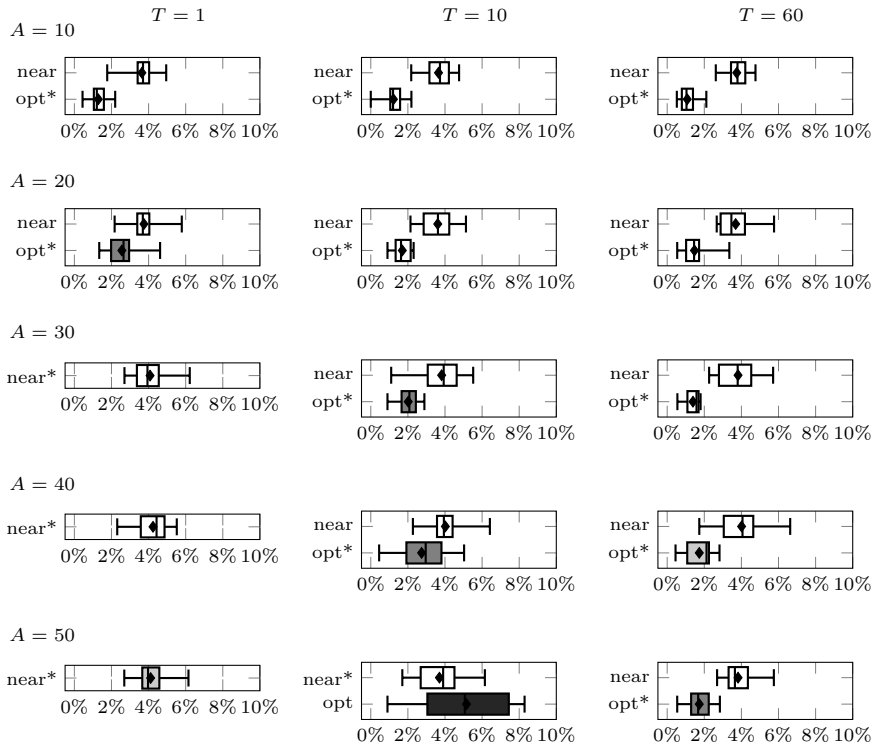

Fig. 13 Comparison of StocOpt with StocNear for a visibility area of size $V = 20$. The suggested algorithm is shown with an asterisk.

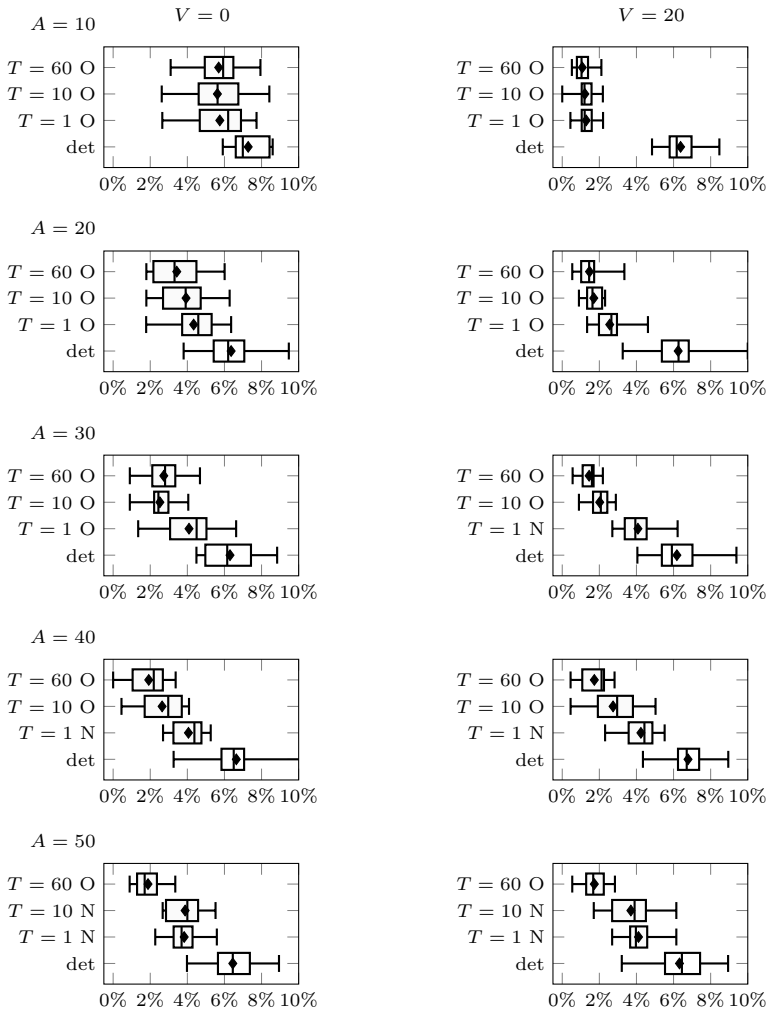

Fig. 14 Comparison of the stochastic algorithms with the deterministic one.

Table 2 Comparison of the average results obtained with the deterministic version (det), StocOpt, and StocNear for the different values of T, V, and A. The upper, middle and lower part shows the results for the time limit $T = 1$, 10, and 60 seconds, respectively. The first column gives the size of the accessible area A. Columns 2 to 4 and 5 to 7 provide the results for the visible area size $V = 0$ and 20, respectively. For each combination of T, V, and A, we give the ARPD of det, StocOpt, and StocNear, and in brackets, we display the relative improvement of the ARPDs of StocOpt and StocNear when compared to det.

$T = 1$						
$V = 0$				$V = 20$		
A	det	StocOpt	StocNear	det	StocOpt	StocNear
10	7.28	5.75 (21.0%)	5.74 (21.2%)	6.38	1.29 (79.8%)	3.64 (42.9%)
20	6.36	4.34 (31.8%)	4.25 (33.2%)	6.25	2.56 (59.0%)	3.75 (40.0%)
30	6.3	4.09 (35.1%)	4.53 (28.1%)	6.18	-	4.08 (34.0%)
40	6.64	-	4.06 (38.9%)	6.78	-	4.24 (37.5%)
50	6.45	-	3.83 (40.6%)	6.32	-	4.11 (35.0%)
$T = 10$						
$V = 0$				$V = 20$		
A	det	StocOpt	StocNear	det	StocOpt	StocNear
10	7.28	5.62 (22.8%)	5.39 (26.0%)	6.38	1.22 (80.9%)	3.65 (42.8%)
20	6.36	3.93 (38.2%)	4.09 (35.7%)	6.25	1.7 (72.8%)	3.61 (42.2%)
30	6.3	2.52 (60.0%)	4.13 (34.4%)	6.18	2.02 (67.3%)	3.82 (38.2%)
40	6.64	2.64 (60.2%)	3.75 (43.5%)	6.78	2.74 (59.6%)	4.02 (40.7%)
50	6.45	3.34 (48.2%)	3.88 (39.8%)	6.32	5.16 (18.4%)	3.70 (41.5%)
$T = 60$						
$V = 0$				$V = 20$		
A	det	StocOpt	StocNear	det	StocOpt	StocNear
10	7.28	5.69 (21.8%)	5.45 (25.1%)	6.38	1.08 (83.1%)	3.75 (41.2%)
20	6.36	3.43 (46.1%)	4.01 (36.9%)	6.25	1.47 (76.5%)	3.69 (41.0%)
30	6.3	2.73 (56.7%)	4.28 (32.1%)	6.18	1.39 (77.5%)	3.82 (38.2%)
40	6.64	1.92 (71.1%)	3.84 (42.2%)	6.78	1.73 (74.5%)	4.01 (40.9%)
50	6.45	1.88 (70.9%)	3.86 (40.2%)	6.32	1.73 (72.6%)	3.82 (39.6%)