Computers & Operations Research 157 (2023) 106280

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

One-dimensional stock cutting resilient against singular random defects

Claudio Arbib [a], Fabrizio Marinelli [b,*], Ulrich Pferschy [c], Fatemeh K. Ranjbar [a,1]

[a] Dipartimento di Ingegneria/Scienze dell'Informazione e Matematica, Università degli Studi dell'Aquila, L'Aquila, Italy
[b] Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche, Ancona, Italy
[c] Institut für Operations und Information Systems, Universität Graz, Graz, Austria

ARTICLE INFO

Keywords:
Cutting stock
Bin packing
Recoverable defects
Dynamic programming
Mixed integer programming

ABSTRACT

When industrial components are obtained by cutting bars of raw material (*stocks*), production volumes and values can be affected by random defects in the stocks. To deal with this inconvenience, we propose to design reconfigurable cutting patterns that can be adjusted so that defects fall, as far as possible, in the residual area that is normally discarded. In this situation, a trade-off arises between the amount of this scrap area and the probability that there exists a reconfiguration with no loss of items. We define mathematical models for the expected economic value produced with a single stock, or with all the stocks cut to obtain the required items. We then introduce the relevant optimization problems, discuss their complexity and devise various solution algorithms, comprising dynamic programming and Integer Linear Programming. The effectiveness of our algorithms is finally illustrated by computational tests on sample problems derived from the literature.

1. Introduction

In the One-dimensional Cutting Stock Problem (CSP) we are given a set $I := \{1, \dots, n\}$ of items of (generally non-distinct) integer lengths $w_i \in \mathbb{N}$, $i \in I$, and sufficiently many bars of raw material from which all the items can be cut.[2] Those bars are called the *stock items* (in brief, *stocks*) and are all of a standard integer length $w \in \mathbb{N}$. Obviously, we assume $w_i < w$ for all $i \in I$. A feasible solution of the basic CSP is a partition of I into m subsets $P_1, \dots, P_m$ such that

$$\sum_{i \in P_k} w_i \leq w$$

for $k = 1, \dots, m$. Every $P_k \subseteq I$ fulfilling the above length condition is called a *pattern*, and a collection of patterns that define a solution is called a *cutting plan*.

For every pattern P we define the *residual piece* as the part of the stock not used (lengthwise) by the items of P. Its length w_0^P is called the pattern *leftover*:

$$w_0^P = w - \sum_{i \in P} w_i \geq 0 \tag{1}$$

From an economical perspective, the total gain z of a solution using m stocks is given by profit minus cost: $z = \sum_{i \in I} e_i - m \cdot c$, where $e_i \in \mathbb{Q}_+$ is the *profit* (or *economic value*) of item $i \in I$, and $c \in \mathbb{Q}_+$ is the stock unitary cost. In a traditional CSP, the total value of all items in I does not depend at all on how patterns are formed, and hence maximizing the gain z is equivalent to minimizing the number m of stocks used. We will denote by m^* the optimal, i.e. minimum, number of patterns whose union gives I.

An interesting new aspect arises however if stocks are prone to defects. If the positions of defects are known before the cutting patterns are computed, an optimal CSP solution with no defective items can be obtained, at least in principle, by splitting the stocks so as to remove all the defects and then solving a multiple length CSP (Alves and de Carvalho, 2008). This approach is however not common in industrial settings for considerations related to internal logistics: stocks placed far away from the manufacturing department; costs of offline stock inspection, classification and pre-cut, etc. Quite often, stocks are therefore cut as if they were faultless, and defective items are just discarded afterwards. An intermediate approach, however, is to adopt a *robust* solution coupled with some online recourse action to limit the impact of defects. Such a solution can be defined – intuitively – as a cutting plan where the items of each pattern can be rearranged to avoid defects as far as possible. To fix ideas, imagine a practical CSP setting with the following timeline:

* Corresponding author.
E-mail addresses: claudio.arbib@univaq.it (C. Arbib), fabrizio.marinelli@staff.univpm.it (F. Marinelli), ulrich.pferschy@uni-graz.at (U. Pferschy), fatemeh.kafashranjbar@graduate.univaq.it (F.K. Ranjbar).

[1] Funded by the Italian Ministry of Education, PON 2014-2020, CCI 2014IT16M2OP005.

[2] Most of the cutting and packing literature assumes integer values of items. Clearly, rational data could be covered as well by multiplying all numbers by their lowest common denominator.

https://doi.org/10.1016/j.cor.2023.106280
Received 15 November 2022; Received in revised form 30 March 2023; Accepted 12 May 2023
Available online 15 May 2023

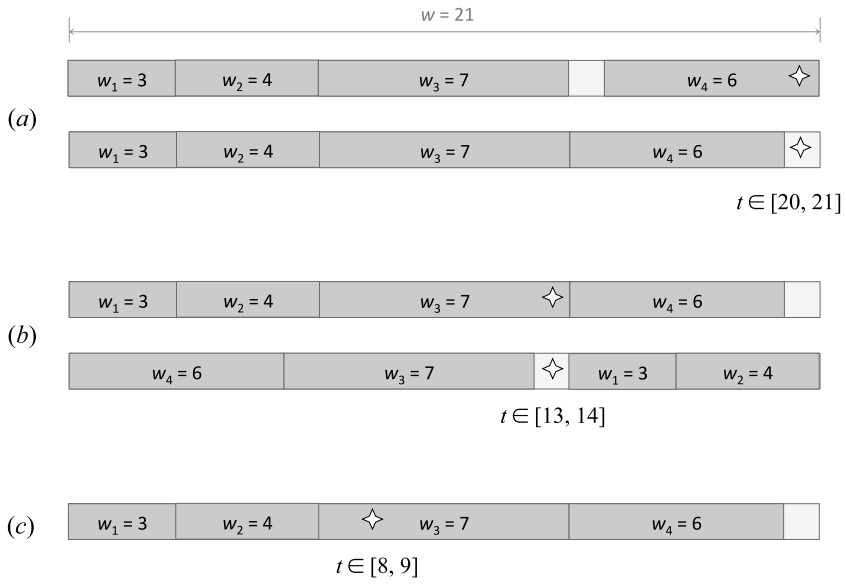

Fig. 1. $(a)-(b)$ t-reconfigurable and (c) not t-reconfigurable pattern for a defect in position t indicated by a star.

1. First, the data of all items in I are received and a basic CSP is solved, i.e., I is partitioned into a sequence of patterns $P_1, \dots, P_m$.
2. Then a sequence of stocks becomes available, and the jth stock of the sequence is cut according to pattern P_j. Both the defectiveness of each single stock and, if defective, the positions of faults within it are regarded as random events.
3. When the jth stock is received, it is checked and possible defect positions are detected. With this information, the items of P_j, or a subset thereof, are assigned to fixed positions on the horizontal segment $[0, w]$, so that no interval occupied by an item contains a defect.
4. Unassigned items – if any – are permanently discarded, that is, do not contribute to generate profit.

Notice that, in the above setting, defect positions are not known during phase 1, i.e., when patterns are formed: only when the stock is ready to be processed, it becomes evident whether it is defective or not and – if it is the case – where defects are positioned. Examples of production settings of this kind taken from literature will be given in Section 2.

If, for given defect positions, all the items of a pattern can be produced without faults, the pattern is said to be *reconfigurable* (see Fig. 1). This gives a probabilistic meaning to the notion of *solution robustness*: in fact, we define the robustness $\pi(P)$ of a pattern P as the probability that P is reconfigurable against defects occurring at random in the stock associated.

If pattern P turns out to be not reconfigurable for a certain arrangement of defects, then not all items can be produced (phase 4 above), i.e. cut from the faulty stock, which results in an *economic loss* equal to the sum of economic values, or profits, of all discarded items of P. In general, once defects are spotted in a stock, a local (deterministic) problem arises to minimize the economic loss the defects cause in the set of items assigned. The *Expected Economic Loss* (EEL) of a pattern P, denoted as $ee\ell(P)$, is then obtained by summing up the minimum loss over all possible defect positions, weighing each one by the probability that a defect arises exactly there.

Finally, considering that the processed stocks have a cost, one can also ask to compute the number m of patterns that maximizes the expected gain. More formally, one can define the *Expected Total Revenue* (ETR) as the total value of the items, minus the EEL and the total stock cost. Intuitively, the two opposing goals of material utilization and robustness should be balanced. In fact, a solution with a minimum number of stocks will generally be tightly packed; consequently, chances are higher that items must be discarded because of defects in the stock. On the other hand, spreading the items over a larger number of stocks will entail a larger slack; thence, one would have more chances to avoid economic loss, but at the cost of using more raw material (which, by the way, would also give chances to additional defects — see Section 6 for a detailed analysis).

Summarizing, the above discussion gives rise to four problems, two of them concerning the robustness of a single pattern:

Problem 1 (*Pattern Robustness*)**.** Given a pattern P, compute its robustness $\pi(P)$, i.e., the probability that P is reconfigurable.

Problem 2 (*Pattern Expected Economic Loss*)**.** Given a pattern P and item economic values e_i, $i \in P$, compute $ee\ell(P)$.

The other two problems refer to robust CSP solutions with favorable properties in terms of both residual pieces and sensitivity to defects:

Problem 3 (*Total Expected Economic Loss*)**.** Given a fixed number of stocks $m \geq m^*$ and the probability ρ that a stock is subject to one defect, find a cutting plan (with some patterns possibly empty) which minimizes the total EEL

$$ee\ell(P_1, \dots, P_m) \;=\; \rho \cdot \sum_{k=1}^{m} ee\ell(P_k) \tag{2}$$

Problem 4 (*Total Expected Revenue*)**.** Assuming unitary stock cost $c \in \mathbb{Q}_+$, find a cutting plan which maximizes the ETR:

$$etr(P_1, \dots, P_m) \;=\; \sum_{i \in I} e_i - ee\ell(P_1, \dots, P_m) - c \cdot m \tag{3}$$

or equivalently minimizes the total stock cost plus EEL.

In this paper we consider the one-dimensional setting, and hence refer to applications where stocks are iron bars, lumber rods, and similar. We also assume that defects are point-shaped spots that corrupt only a unit interval of the stock, and that the simultaneous occurrence of more than one such defect in a stock is a rare event, i.e., each stock contains at most one defect. These assumptions will be further discussed in Section 3.1.

Finally, as in the standard CSP, we assume $e_0 = 0$, i.e., that the leftovers have no economic value. In practice, the leftover of a stock might well be utilized in a later stage. However, keeping a stack of

partially used raw material of different sizes often constitutes an operational burden or is technically infeasible. The difficulties of producing "useful" leftovers under uncertainty was recently investigated by Cherri et al. (2023). In any case, it is difficult to estimate the monetary value of a residual piece that may (or may not) be used in some future production lot. That is why we stick to the standard model and represent the cost of a solution by the number of utilized stocks (= patterns) multiplied by the stock unitary cost c, thereby ignoring the potential values of residual pieces. Generally speaking, it may also happen that eliminating some items would increase the total gain. Take for example $w = c = 10$, $|I| = 4$, $w_i = 3$ and $e_i = 6$: an optimal solution that cuts all the four items uses two stocks and hence produces a total gain of $6 \cdot 4 - 10 \cdot 2 = 4$, but the gain increases to $6 \cdot 3 - 10 = 8$ if one gives up one item and so saves the cost of one stock. However, we again stick to the standard CSP assumption that requires the production of *all the items of* I (some of which might possibly be discarded later because of defects detected during the production process) because such an item elimination strategy has only a marginal impact concerning the use of the very last stock.

Our contribution comprises three main aspects: First of all, we introduce a new cutting stock problem based on a stochastic model of small defects and study theoretical properties of the resulting expected utilization of stock. Secondly, we state a subset sum based deterministic algorithm, which computes a-priori the best utilization of a given cutting pattern for every possible defect position. Algorithmic improvements are developed to reduce its running time complexity. Finally, we introduce ILP-based models for maximizing the total expected revenue for a given list of orders balancing the stock cost with the expected loss incurred by defects. Computational experiments analyze the practical behavior of these models.

The paper is organized as follows. In Section 2 we briefly survey the existing literature on related topics. In Section 3 we introduce the reconfiguration issue, and the notion of pattern robustness against a single random defect in the stock; here we also define (Section 3.2) the expected economic loss of a pattern (or of a whole packing) as the minimum loss of item values obtainable by pattern reconfiguration. In Section 4 we discuss the complexity of computing pattern robustness and expected economic loss. Solution approaches are proposed in Section 5: in particular, Problem 4 is treated by repeatedly solving Problem 3 for increasing number of stocks, starting from $m = m^*$ and moving up to a convenient number computed by solving a CSP with all reconfigurable patterns. A computational experience based on benchmark instances taken from literature is described in Section 7, and conclusions are finally drawn in Section 8.

2. Related work

The recent scientific literature on discrete optimization includes a large body of theoretical and application-oriented papers on cutting (and packing) problems, see, e.g., the typology by Wäscher and Haußner (2007) and the survey on the deterministic setting without defective stocks by Delorme et al. (2016).

Industrial cutting processes use a large variety of raw material: glass in automotive and building components (Arbib et al., 2022b,a), lumber logs and boards in the furniture and wood industry (Ghodsi and Sassani, 2005; Wenshu et al., 2015), leather sheets in shoes and textile (Sarker, 1988; Özdamar, 2000), steel in metallurgy (Sierra-Paradinas et al., 2021), silicon plates in electronics (Hwang et al., 2011) and paper rolls in paper mills (Aboudi and Barcia, 1998), just to mention a few. Although raw material is almost always subject to possible imperfections (knotholes in wood, bubbles in glass, contaminated areas in steel, holes, stains or streaks in paper, etc.), relatively few papers in the cutting literature address problems arising in defective stock cutting, and in most of them defect size and location are known in advance.

One of the earliest contributions on cutting problems with known defects is due to Hahn (1968). Since then, variants considering one (Carnieri et al., 1993; Neidlein et al., 2009) or more (Afsharian et al., 2014; Wenshu et al., 2015) defects per stock were addressed, either in one or two dimensions. In those papers, defect geometry ranges from simple points to rectangular or generally convex areas, and damage can either involve immediate item scrap or just reduce item quality. Also defect handling is addressed in different ways: it can concern the optimal use of faultless areas or, more generally, the optimization of the whole cutting process including fault numbers and/or placement of items in the stock (Sarker, 1988). A recent paper by Durak and Tüzün (2017) considers an optimization setting where only a few stocks are available at any given time; however, despite the online nature of the process, the faults affecting that limited stock set are again treated in a deterministic way. The cutting problems with known defects addressed in the above references are mostly modeled as mixed integer linear programs, and solved by dynamic programming, Lagrangian relaxation and subgradient optimization (Durak and Tüzün, 2017; Rönnqvist and Åstrand, 1998; Rönnqvist, 1995).

In some cases, the problem calls for rearranging the items of a given pattern so as to avoid as many defects as possible. To the best of our knowledge, the earliest reference to this issue is the Min Defective Subset Sum problem considered in Aboudi and Barcia (1998), where a single defective interval occurs at a given position in the stock, and one wishes to find a layout of a given pattern that places items for a maximum total value. The authors observe that the problem is $\mathcal{NP}$-hard already in the one-dimensional case, reformulate it as Multiple Subset Sum and solve it by a branch-and-bound approach. Such a problem is actually a special case of the Multiple Knapsack Problem (Dell'Amico et al., 2019), with two knapsacks and profits equal to lengths.

Papers dealing with uncertainty in cutting problems especially address the stochastic behavior of such major item attributes as size, demand and economical value. Indeed, most cases consider either an online process where items arrive sequentially, or a production setting with uncertain demand. Just to mention some examples, robust optimization is used in Alem and Morabito (2012) to address a combined lot-sizing and cutting-stock problem, and in Ide et al. (2015) to address an application in the wood cutting industry. Uncertainty in customer demand is tackled by two-stage stochastic optimization in Alem et al. (2010) and Beraldi et al. (2009). Random customer demand and random cutting times are investigated in Krichagina et al. (1998) and heuristically solved by linear programming and dynamic control.

A parallel line of contributions in the stochastic setting concerns the One-dimensional Bin Packing problem (very similar to the CSP) with uncertain item sizes. Recent works by Perez-Salazar et al. (2022) and Schepler et al. (2022) provide a rich source of references. A closely related problem, indeed a subproblem of the CSP, is the classical Knapsack problem: uncertain variants are addressed in Monaci et al. (2013), where the item weights belong to a given interval, and in Gaivoronski et al. (2011), where different elements of the problem formulation, subject to a degree of uncertainty described by random variables, are reformulated by probability constraints and a solution approach based on semi-definite relaxation is proposed. However, these references do not take stock defectiveness into account.

Several articles, indeed, observe that feeding cutting machines with stocks is an online process and, as such, calls for a stochastic model of defects. An early paper by Sculli (1981) assumes a random defectiveness of one-dimensional stocks that however only affects the edges (and therefore the size) of rolls. More recent examples of industrial cutting problems, where defects are detected only immediately before the cutting process, can be found, e.g., in wood processing. Window frame production with on optical surface scanner integrated in the cutting machine was discussed in Petutschnigg et al. (2007). A more complicated analysis of logs scanned by a computer tomograph for internal errors in the production of high quality lumber was treated in Petutschnigg et al. (2005), Pernkopf and Riegler M. Gronalt (2019).

Another example is described in Arbib et al. (2022b) where an assortment-and-cut problem arising in the glass industry is considered, and a stochastic model for defect occurrence proposed. The stochastic realization of defects is modeled as a spatial Poisson point process, and a mixed integer program in the vein of robust optimization is presented. This paper presents an analogy with the present contribution in an idea of recourse strategy based on pattern reconfiguration. However, the context sensibly differs from ours because, although 2-dimensional, patterns are elementary. Moreover, the main focus of the problem is on the reduction of back-orders rather than economic loss.

3. Pattern reconfiguration

Suppose that, independently on other stocks, every stock is subject to a single defect with probability ρ. If a stock has a defect, assume its integer position $t \in [1, w]$ to be distributed along the stock length according to some discrete probability function $f(t)$ independent from and identical for all stock. Given the integrality of the item sizes, it is easy to see that any fractional defect position t can be replaced by $\lceil t \rceil$.

As mentioned in the introduction, we here consider small defects which render only one length unit of the stock unusable. We can then formally associate a defect with a certain integer position $t \in [1, w]$, which decomposes the stock into two intervals, $[0, t-1]$ and $[t, w]$, that can still be used to produce items. The part of the stock between positions $t-1$ and t contains the defect and cannot be used. Two cases may then occur for a given pattern P: either the defect can be isolated within the residual piece of the pattern (Figs. 1-*a*, *b*), or this operation is impossible and therefore an item must be discarded (Fig. 1-*c*). In the former case, no economic loss is incurred because the leftover has no economic value; in the latter case, we assume the loss for not producing an item to be equivalent to the item value.

For a defect at an integer position t we then define:

Definition 1. Let $P \subseteq I$ be a pattern and let $t \in [1, w] \cap \mathbb{N}$. We say that P is t-reconfigurable if there exists $L \subseteq P$ such that

$$w(L) = \sum_{i \in L} w_i \;\leq\; t-1 \qquad\qquad w(P \setminus L) = \sum_{i \in P \setminus L} w_i \;\leq\; w-t \tag{4}$$

Moreover, we say that P is *reconfigurable* if it is t-reconfigurable for all integer $t \in [1, w]$.

Given a finite set P of numerical weights $w_i \in \mathbb{N}$ and a capacity $q \in \mathbb{N}$, the classical Subset Sum Problem (SSP), cf. Kellerer et al. (2004), asks for finding a subset $S \subseteq P$ with maximum total weight such that $\sum_{i \in S} w_i \leq q$. From an algorithmic point of view, one can then search for a subset L fulfilling (4) by solving a subset sum problem with item set P and capacity $t-1$. If the optimal solution set S has total value $w(S) \geq w(P) - w + t$, then P is t-reconfigurable.

Recalling the Min Defective Subset Sum problem pointed out in Section 2, a pattern P is t-reconfigurable if and only if the instance of Min Defective Subset Sum defined by P and a defective interval $[t-1, t]$, has an optimum with no discarded items.

Note that if P is t-reconfigurable for some t, then it is also $(w-t+1)$-reconfigurable by symmetry. Although this aspect can be useful to speed-up computations, we will not consider it explicitly in this paper for simplicity of exposition.

Definition 1 can be generalized to r defect positions in a stock. However, in this paper we suppose that the simultaneous occurrence of more than one defect in a stock is a rare event, and hence we focus only on the case where each stock contains no more than one defect.

3.1. Pattern robustness

All the integers $t \in [1, w]$ for which P is t-reconfigurable define a set $T_P \subseteq [1, w]$ built from a finite number of *reconfiguration intervals* as shown in Fig. 2. The reconfiguration intervals are at most 2^p, with $p = |P|$ (Fig. 2-*a*), and each of them is generated from a subset $L \subseteq P$ by moving all the items of L to the left side of the stock and the remaining ones to the right: Fig. 2-*c* shows that no more sets are necessary to build T_P, as any other way of t-reconfiguring a pattern is implied by a subset of reconfiguration intervals. Clearly, $T_Q \supseteq T_P$ for any subset Q of P.

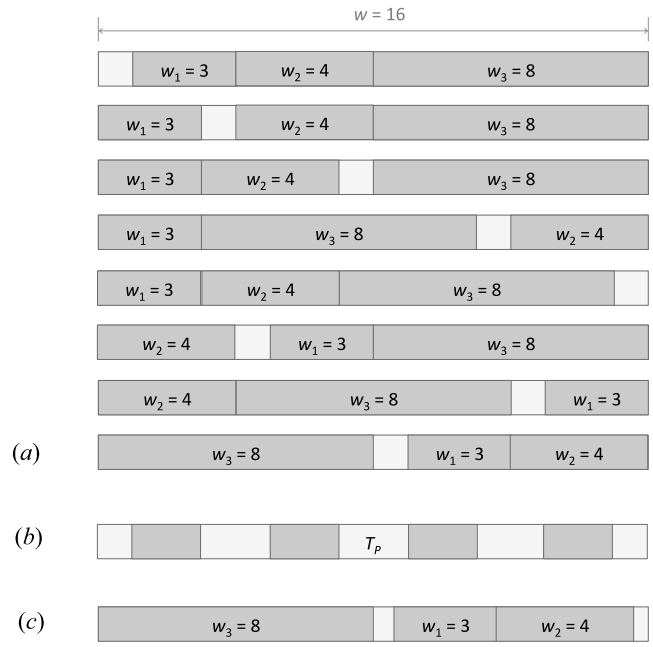

Fig. 2. (*a*) Reconfiguration intervals, (*b*) reconfiguration set and (*c*) a generic reconfiguration implied by T_P.

Given a discrete probability distribution $f : [1, w] \cap \mathbb{N} \to \mathbb{R}$, where $f(t)$ gives the probability of a defect at position t, we define the measure $\pi(P)$ under f as the probability that P is t-reconfigurable (i.e. that all the items of P can be produced) for a single defect at position t distributed according to f along the length of the stock:

$$\pi(P) = \sum_{t=1}^{w} \chi(P, t) f(t), \tag{5}$$

where $\chi(P, t)$ is a characteristic function with $\chi(P, t) = 1$ if P is t-reconfigurable and 0 otherwise. The probability $\pi(P)$ represents the pattern robustness as stated in Problem 1: that is, the larger the $\pi(P)$, the more pattern P is *robust* against defects. If, in particular, $\pi(P) = 1$, then P is reconfigurable.

3.2. Expected economic loss of a pattern

If a pattern turns out to be not t-reconfigurable for a defect position t, then some items must be removed from the pattern as they cannot be produced; equivalently, if produced, we assume that these items contribute no revenue, just like the pattern residual piece. Considering a certain defect position t, it is easy to see that one will never have to remove more than one item from any pattern P: given in fact a certain ordering of the items in P, where each item is associated to an interval with integer extremes, it suffices to remove the unique interval containing t.

To minimize the economic loss $\ell(P, t)$ that may occur if pattern P is hit by a defect in position t, one then has to identify the least valuable item whose removal makes P a t-reconfigurable pattern: clearly, the removed item (if any) may be different for different defect positions. To compute the expected economic loss (EEL) of a pattern P, denoted as $ee\ell(P)$, we have then to determine the corresponding reaction, i.e. the removal of the least valuable item, for every position t for which P is not t-reconfigurable.

Example 1. Let $w = 10$, $P = \{1, 2\}$, $w_1 = 1$, $w_2 = 9$ and $e_i = e\, w_i$ for some positive constant e. Moreover, suppose that a defect occurs at an integer position uniformly distributed in $[1, 10]$.

Because the leftover is $w_0^P = 0$, P is not t-reconfigurable for any $t \in [1, 10]$, hence $\pi(P) = 0$. However, if $t \in \{1, 10\}$ we can reorder items and save the most valuable item, i.e., 2: thus we lose ew_1 with probability 0.2; in all the other cases, we are forced to lose ew_2. Therefore the EEL of pattern P is $ee\ell(P) = (0.2w_1 + 0.8w_2)e = 7.4e$. □

Generally speaking, to express $ee\ell(P)$ we need to know the probability π_i that i is the least expensive item whose removal allows to reconfigure the pattern. The EEL will then be computed as

$$ee\ell(P) = \sum_{i \in P} e_i \pi_i \,. \tag{6}$$

For economic values depending on the item sizes, the computation of the least valuable item for removal from a given pattern P can be formalized as follows. Considering the removal of a certain item k from P, we denote the set of all defect positions t for which $P \setminus \{k\}$ is t-reconfigurable as $T^k := T_{P\setminus\{k\}}$ and T^0 for T_P. We already observed $T_Q \supseteq T_P$ for $Q \subseteq P$: thus $T^0 \subseteq T^k$ for all k. In addition

Proposition 1. *Assume that the items in P are sorted by non-decreasing economic values $e_1 \le \ldots \le e_p$. If, for all $i \in P$, $e_i = e(w_i)$ for a non-decreasing function $e : \mathbb{Q}_+ \to \mathbb{Q}_+$, then $T^i \subseteq T^k$ for any $i < k$.*

Proof. By our assumption, $i < k \Rightarrow e_i \le e_k$, therefore $w_i \le w_k$ by definition of $e(.)$. Let Q be a reconfiguration interval of T^k corresponding to a partition L, R. Suppose w.l.o.g. $i \in L$ and set $L' = L \setminus \{i\} \cup \{k\}$, i.e., replace k for i: then $w(L') \ge w(L)$. Let Q' be the reconfiguration interval of T^i obtained by aligning L' to the left of the stock and the remaining items to the right: since $w(R)$ remains unchanged, we have $Q' \subseteq Q$. A similar argument can be repeated if $i \in R$. As the argument holds for all the reconfiguration intervals of T^i, T^k, we then conclude $T^i \subseteq T^k$. □

Now let us formalize how to select the item to be discarded for a defect position t under the assumption of Proposition 1.

- If $t \in T^0$, no item has to be removed.
- If $t \in T^1$, we can select the least valuable item $i = 1$. So we discard item 1 for $t \in T^1 \setminus T^0$.
- If $t \in T^2 \setminus (T^0 \cup T^1) = T^2 \setminus T^1$, we will discard $i = 2$. And so on.

Summarizing, for a non-decreasing value function $e(w_i)$ the probability π_i with which the ith lowest valued item of P will be discarded is

$$\pi_i = \sum_{t \in T^i \setminus T^{i-1}} f(t). \tag{7}$$

Note that the number n_i of integer points in $T^i \setminus T^{i-1}$ gives how many of the w defect positions incur an economic loss equal to e_i. Hence, in the simple case of an integer defect position distributed uniformly at random in $[1, w]$, one has $\pi_i = \frac{n_i}{w}$.

4. Complexity and solution approaches

This section is devoted to state fundamental properties of the four problems defined in Section 1. Besides complexity results, we here describe basic algorithmic approaches that will be refined in Section 5 by pseudo-polynomial dynamic programming algorithms.

To compute, for a given pattern and an arbitrary discrete defect distribution, the robustness $\pi(P)$ and the expected economic loss $eel(P)$ (Problems 1 and 2), at first a fixed defect position has to be considered. This gives rise to the following subproblem.

Pattern reconfiguration. Given a pattern P with p items, $w(P) \le w$, and an integer defect position $t \in [1, w]$, is P t-reconfigurable?

Proposition 2. Pattern reconfiguration *is weakly $\mathcal{NP}$-complete.*

Proof. The problem can easily be reduced from Partition (Garey and Johnson, 1979):

Partition. *Given a set S of p numbers $a_i \in \mathbb{N}$, decide whether S contains or not a subset S_1 such that*

$$w(S_1) = \sum_{i \in S_1} w_i = \sum_{i \notin S_1} w_i = w(S \setminus S_1).$$

Define an instance of Pattern Reconfiguration with $w_i = a_i$ for $i = 1, \ldots, p$, and $w = w(S) + 1$. We can assume w odd (otherwise Partition must be a *no* instance). For $t = \frac{w+1}{2}$, Pattern Reconfiguration has a *yes* answer if and only if we have a *yes* instance of Partition. □

From an algorithmic viewpoint, the robustness $\pi(P)$ can be computed by solving Pattern Reconfiguration for all integers $t \in [1, w]$ and summing up the probabilities $f(t)$ for all t values with a "yes" answer. Such an answer can be found by searching for a subset $L \subseteq P$ fulfilling conditions (4). This task resembles a subset sum problem and can be solved, e.g., by an elementary integer linear programming formulation or by dynamic programming (see Section 5).

From a theoretical perspective we can establish the following upper bound on $\pi(P)$ for a uniform defect probability distribution.

Proposition 3. *If $f(t) = 1/w$, the robustness of a pattern can be bounded by*

$$\pi(P) \le 2^p \, \frac{w_0^P}{w} \,. \tag{8}$$

Proof. Every $L \subseteq P$ gives rise to a t-reconfigurable interval, with extremes $w(L)+1$, $w(L)+w_0^P$, that contains w_0^P integer defect positions. In general, these intervals may overlap, but in the best case there can be 2^p subsets L defining disjoint intervals of w_0^P positions. Therefore, uniformly distributed defects can cover a total number of at most $2^p\, w_0^P$ positions out of the w positions of the stock length. This implies (8). □

The bound given in Proposition 3 is strict, if the reconfiguration set T_P is the union of 2^p disjoint intervals. As an example for this case one can consider $P = \{5, 9, 11\}$ with $w = 27$ and thus $w_0^P = 2$.

The complexity of computing $\pi(P)$ (that is, of Problem 1) can be established by considering the following variant of Partition:

Subpartition. *Given a set S of numbers $a_i \in \mathbb{N}$, decide whether S contains or not two disjoint subset S_1, S_2 so that*

$$w(S_1) = \sum_{i \in S_1} w_i = \sum_{i \in S_2} w_i = w(S_2)$$

In case of affirmative answer, we say that S has a subpartition. The Subpartition problem is $\mathcal{NP}$-complete (Woeginger and Yu, 1992).

Proposition 4. *There are patterns P for Problem 1, where the question "Is $\pi(P) < 2^p/w$?" is $\mathcal{NP}$-complete to decide.*

Proof. Given an instance $\mathcal{I}$ of Subpartition, construct an instance of Problem 1 with $P = \{w_i \mid w_i = 2a_i, a_i \in S\}$, $w = w(P) + 1$, i.e. $w_0^P = 1$, and uniform defect distribution. Each subset $L \subseteq P$ yields a t-reconfigurable interval consisting only of position $w(L)+1$. Considering Proposition 3, $\pi(P) = 2^p/w$ if and only if all subsets L have distinct length $w(L)$. Thus, the probability of robustness $\pi(P) < 2^p/w$ if and only if $\mathcal{I}$ is a *yes* instance. □

In addition, note that Problem 1 may require a certificate exponentially long in p. Indeed, given an answer to the problem (say, "$\pi(P) = \frac{8}{15}$"), a certificate for that answer is the set T_P that generates the result. Take for instance $w = 2^{p+1} - 1$, $w_i = 2^i$ for $i = 1, \ldots, p = |P|$: the certificate T_P is constructed on p items but is the union of 2^p disjoint intervals, for a total measure $\pi_P = \frac{2^p}{2^{p+1}-1}$.

Moving from pattern robustness to pattern expected economic loss $eel(P)$, as per Problem 2, again a specific integer defect position $t \in [1, w]$ has to be considered. The resulting economic loss can be trivially computed by repeatedly eliminating one item $i \in P$ at a time, and checking whether $P \setminus \{i\}$ is t-reconfigurable. As above, the expected economic loss is derived as the weighted sum of economic losses over all defect positions. In fact, finding the minimum economic loss for a fixed defect position is equivalent to solving the Min Defective Subset Sum problem defined in Aboudi and Barcia (1998), where the problem is recognized as $\mathcal{NP}$-hard and an ILP model is proposed to partition P into subsets placed left and right of t, maximizing the total profit. Dynamic programming approaches for this problem as well as for the more general Problem 2 will be presented in Section 5.

Note that any algorithm for Problem 2 can also be used to answer Problem 1 by setting $e_i = 1$ for all $i \in P$ and reporting $\pi(P) = 1 - eel(P)$. Hence we conclude from Proposition 4.

Proposition 5. *Problem 1, and therefore Problem 2, is $\mathcal{NP}$-hard.*

5. Dynamic programming algorithms

In this section we discuss algorithmic alternatives for solving Problem 2, i.e., for computing the expected economic loss of a single pattern P. We had already discussed elementary solution approaches for this weakly $\mathcal{NP}$-hard problem in Section 4. Here we consider pseudo-polynomial dynamic programming algorithms. As usual, dynamic programs require a certain effort for computation and storage. However, the solution structure produced for a given pattern P can be used to return the minimum economic loss for arbitrary integer defect positions t in constant time. We will present two general approaches in Sections 5.1 and 5.2, then refine the latter in Section 5.3.

5.1. Decomposition approach

As a first option for solving Problem 2 we can generate all decompositions of the total pattern length into two segments w', w'' such that $w' + w'' \leq \sum_{i \in P} w_i$, which can be obtained from two disjoint subsets $P' \cup P'' \subseteq P$ with $\sum_{i \in P'} w_i = w'$ and $\sum_{i \in P''} w_i = w''$. Any such decomposition allows a pattern reconfiguration if $w' \leq t - 1$ and $w'' \leq w - t$. The computation can be done by keeping all such decompositions (w', w'') as states of a dynamic program. The profit of a state is trivially given by $\sum_{i \in P' \cup P''} e_i$ for the corresponding subsets P' and P''. During the computation, items $j \in P$ are added iteratively to every state (w', w'') giving rise to possible new states $(w' + w_j, w'')$ and $(w', w'' + w_j)$. If such a state already exists, we keep the one with the higher profit value. Without going into details, it is easy to see that such an approach would require a running time of $O(w^2 p)$, since the number of states is trivially bounded by w^2. To minimize the economic loss $\ell(P, t)$ for a certain defect position t, one has to find a state (w', w'') with maximum profit, and therefore minimum loss, such that $w' \leq t - 1$ and $w'' \leq w - t$.

In order to successively compute the minimum economic losses $\ell^*(P, t)$ for various defect positions, we determine an auxiliary array $\texttt{profit}[\omega', \omega'']$ where every entry contains the maximum profit over all states (w', w'') with $w' \leq \omega'$ and $w'' \leq \omega''$ (and zero if no such states exist). Once this array is filled in $O(w^2)$ time,[3] $\texttt{profit}[t-1, w-t]$ corresponds to $\ell^*(P, t)$ for any given integer $t \in [1, w]$.

The practical performance of such an approach will suffer from the quadratic influence of the capacity w. Thus, we will not explore this approach in our computational experiments.

[3] This can be done e.g. by going through the array row-wise in an outer loop. For each ω'' we keep an increasing sequence of profits over all ω'. Incrementing ω'', it suffices to compare the states for ω'' with the best profit found for $\omega'' - 1$.

5.2. An approach based on Subset Sum

Algorithm 1 Subset Sum-based dynamic programming for computing $\pi(P)$ and $ee\ell(P)$

```
loss[r'] := max_{j∈P} e_j, ∀r' ∈ [1, w]
π(P) := 0
for all j ∈ P do
    R^j := Subset Sum(P \ {j})
    R^j := UpdateLoss(R^j, w_0^P + w_j, e_j)
end for
if w_0^P ≥ 1 then
    R^0 := Subset Sum(P)
    R^0 := UpdateLoss(R^0, w_0^P, 0)
    π(P) := Σ_{r∈R^0} f(r)
end if
eeℓ(P) := (1/w) Σ_{r'∈[1,w]} loss[r']
output (π(P), eeℓ(P))

function UpdateLoss(R, w^0, e)
    R' := ∅
    for all r ∈ R and r' ∈ [r + 1, r + w^0] do
        R' := R' ∪ {r'}
    end for
    for all r' ∈ R' do
        loss[r'] := min{loss[r'], e}
    end for
    return R'
end function
```

As a second possibility we can exploit the fact that at most one item will be discarded from P for any defect position t (recall Section 3.2). Therefore, one can simply go through all p candidates and solve the Subset Sum problem (4) for the remaining $p - 1$ items, which takes p executions of an $O(pw)$ time algorithm, i.e. $O(p^2 w)$ time.[4] The iteration discarding the item with the smallest economic value e_j and permitting a solution of (4) returns $\ell^*(P, t)$ for the given position t (in order to solve Problems 1 and 2, an additional iteration with all items must eventually be performed).

However, noting that the usual dynamic programming algorithm for Subset Sum determines *all* reachable length values $r \in \{0, 1, \ldots, w\}$, i.e. values r such that there exists $S \subseteq P$ with $\sum_{i \in S} w_i = r$, we can do better and use the dynamic program to store information so that $\ell^*(P, t)$ can be computed in the same running time for arbitrary defect positions as described in detail in Algorithm 1. Observe that, assuming the residual piece as a dummy item, the set of reachable lengths corresponds to the union of the starting coordinates of the items of all the *normal patterns* of P as described in Christofides and Whitlock (1977).

Let us define by R^j the set of all length values reachable after discarding some item j, i.e. for the item set $P \setminus \{j\}$, and R^0 the set of reachable values for P. These are computed by the standard dynamic programming procedure Subset Sum$(P \setminus \{j\})$. If a position $r \in R^j$ can be reached, then we can conclude that $P \setminus \{j\}$ is t-reconfigurable for every defect position $r + 1 \leq t \leq r + w_0^P + w_j$. Any such solution yields a loss of e_j. This argument could be easily extended to the case of a defect covering more than one unit of stock.

Throughout the dynamic programming algorithm we will use an array $\texttt{loss}[\,]$ of length w such that $\texttt{loss}[r']$, $r' = 1, \ldots, w$ contains the current minimum loss achievable for a defect at position r'. Every reachable position $r \in R^j$ implies a series of potential updates since $\texttt{loss}[r'] \leq e_j$ for $r' = r + 1, \ldots, r + w_0^P + w_j$. These updates are performed

[4] This assumes the standard dynamic programming algorithm as described in Kellerer et al. 2004, ch. 4.1. Improved algorithms will be discussed in Section 5.3.

by function UpdateLoss($R^j, w_0^P + w_j, e_j$). For R^0 the updated length positions are returned only for the computation of the probability of reconfigurability in line 10.

Starting with an upper bound $\texttt{loss}[r'] = \max_{j\in P}\{e_j\}$ for all r', we consider iteratively the elimination of each item j. Therefore, we compute all positions $r \in R^j$ reachable for $P \setminus \{j\}$ and update the resulting array entries $\texttt{loss}[r'] := \min\{\texttt{loss}[r'], e_j\}$ thus keeping possible improvements gained by discarding item j. Note that the same index r' may be affected for different reachable positions. This update can be done easily by one pass through the dynamic programming array computed for Subset Sum.

Eventually, also R^0 must be computed to nullify the loss of the positions r' for which the pattern is t-reconfigurable without removing any item. Note that this is the case only if $w_0^P \geq 1$ (line 7).

Each iteration for one item j requires $O(pw)$ time which yields an overall running time of $O(p^2w)$, as before. After this procedure, we can compute $\ell^*(P,t)$ for any defect position t in constant time by simply reporting the entry $\texttt{loss}[t]$. Problems 1 and 2 can also be straightforwardly solved by reading $\texttt{loss}[\,]$.

If $\ell^*(P,t)$ shall be computed only for one particular defect position t, it will be useful in practice to sort the items in increasing order of economic values e_j. Then, the computation can be stopped as soon as the removal of an item permits a t-reconfigurable pattern, that is, $\texttt{loss}[t]$ is updated for the first time. Clearly, this happens for the item causing the smallest possible loss. However, this shortcut does not improve the worst-case complexity.

5.3. An improved approach based on Subset Sum

Instead of the standard dynamic programming algorithm, one can employ the recent algorithm by Koiliaris and Xu (2019) which solves the subset sum problem in $\tilde{O}(\sqrt{p}w)$ time, instead of $O(pw)$. Here, $\tilde{O}$ "hides" polylogarithmic factors. Since the algorithm in Koiliaris and Xu (2019) also determines all reachable length values, it can be used to improve the running time of Section 5.2 from $O(p^2w)$ to $\tilde{O}(p^{3/2}w)$.

Sticking to the classical dynamic programming algorithm, we can improve the running time of the strictly serial iteration over all items $j \in P$ performed in Algorithm 1 as follows. For a parameter k (to be chosen later) we group the p iterations into k subsets as follows.

1. Partition the set of items in P into k subsets of (roughly) equal size p/k, then run k iterations, each of them considering the case that the discarded item lies in the corresponding subset.
2. For the remaining $p - p/k$ items, compute all reachable length positions by dynamic programming (as above) in $(p - p/k)\cdot w \approx pw$ time. Store the resulting array of reachable positions.
3. Now consider each of the p/k items j in the current subset as a candidate for discarding: For each such candidate item j, add the remaining $p/k - 1$ items to the stored dynamic programming array from Step 2 to determine in $(p/k - 1)\cdot w \approx p/k \cdot w$ time the positions reachable without item j. Compute the resulting positions R^j and run UpdateLoss($R^j, w_0^P + w_j, e_j$) to update array $\texttt{loss}[\,]$ with e_j in $O(w)$ time, as before.
   For the next candidate j' from the same subset, go back to the dynamic programming array stored in Step 2 containing the reachable positions of the other $p - p/k$ items and do the same procedure as for j.
4. Iterate this process over all k subsets.

The running time of this algorithm consists of k iterations in each of which a dynamic programming array is computed in $O(pw)$ time and there are p/k candidate items to be considered each of them requiring $O(p/k\, w)$ time. This yields a total running time of $O(kpw + p^2/k\, w)$. Plugging in the best choice, $k = \sqrt{p}$, into this expression eventually yields $O(p^{3/2}w)$ time. Note that a similar partitioning approach was used in Malaguti et al. (2019). This construction avoids the polylogarithmic factors above. One could also solve Step 2 by the algorithm in Koiliaris and Xu (2019), but this method does not permit the iterative insertion of items in Step 3. Without going into details, such a combined approach would give a running time of $\tilde{O}(k(\sqrt{p}w + p/k(p/k\, w)))$. This yields $\tilde{O}(p^{5/4}w)$ running time for a parameter $k = p^{3/4}$.

One can push this idea of evaluating and storing partial solutions further, and as a limit obtains the following tree-like structure of the computation. For simplicity of description assume $p = 2^k$. Arrange the set of items of P in a complete binary tree with $\log p$ levels where every leaf corresponds to one item. Each inner node v at level ℓ is the root of a subtree containing in its leaves a subset $S(v)$ of $2^{\log p - \ell}$ items. The main computation consists of the construction of the following auxiliary information:

For each inner node v we compute an array $R(v)$ of length w containing all length positions reachable with items $P \setminus S(v)$, i.e. with all items *except* those in the subtree rooted at v. Starting with the trivial solution for the overall root node *root* (no positions are reachable, $R(root) = \emptyset$), we can go top-down in the tree and compute $R(v)$ for an arbitrary inner node, assuming that $R(pred(v))$ is known. To compute $R(v)$ it suffices to consider $pred(v)$ and all items $S(w)$, where w is the sibling node of v (i.e. $pred(w) = pred(v)$). Add each of these $2^{\log p - \ell}$ items to the set of reachable values in $R(pred(v))$. The total effort to compute these arrays for all 2^ℓ nodes at some level ℓ is $2^\ell \cdot 2^{\log p - \ell}\cdot w = 2^{\log p} w = p \cdot w$. Summing up over all $\log p$ levels gives a running time of $O(p \log(p) w)$ for this preprocessing step.

In the main computation step we keep a loss array $\texttt{loss}[\,]$ as before. Then every item j (=leaf of the tree) is considered as a candidate for removal. For each such j, we still have to consider the possibility of adding the sibling item of j to all reachable length positions in $R(pred(j))$. For ever resulting reachable position r we determine again all implied entries r' and (possibly) update $\texttt{loss}[r']$ with e_j, as in Step 3 above.

All this can be done in $O(w)$ time for one candidate j, which yields an overall running time of $O(pw)$ for the main step. As before, $\texttt{loss}[r']$ equals $\ell^*(P, r')$ for any defect position r', after a total running time of $O(p \log(p) w)$.

It remains to discuss the storage requirements. While the basic dynamic programming algorithm described above as well as the improved version with subset partitioning can be easily implemented using $O(w)$ space for the auxiliary arrays, the tree-based approach would require $O(p)$ auxiliary arrays of length w in a naive implementation. However, we can generate the required arrays $R(v)$ on-demand as follows. When we consider the first item (= leaf) as a candidate for removal, it suffices to generate all $\log p$ arrays on the path to *root*. This approach can be continued during the computation for all items j, so that at any time we do not store more than $O(\log p)$ arrays of length w, yielding a total space complexity of $O(\log(p) w)$.

These descriptions of dynamic programming algorithms to compute $\ell^*(P,t)$ can be summarized in the following result:

Theorem 1. *Given a pattern P, one can compute in $O(\min\{\sqrt{p}, w\}pw)$ time and $O(w)$ space an array of length w containing the minimum economic loss $\ell(P,t)$ for every defect position $t \in \{1, 2, \ldots, w\}$. Disregarding polylogarithmic factors, a running time of $\tilde{O}(p^{5/4}w)$ can be reached. Alternatively, the same result can be obtained in $O(\log(p)pw)$ time and $O(\log(p)w)$ space.*

Algorithm 2 implements the above tree-based dynamic programming procedure by using an iterative construction instead of the more natural recursive function, since the latter is computationally less efficient. The algorithm assumes the nodes of the tree as labeled with the integers $0, \ldots, 2^p - 1$ going top-down over all levels and then left–right in each level. The array $\texttt{tree}[\,]$ describes the depth-first visit: $\texttt{tree}[i]$ is the label of the node visited at the ith step.

Notice that the procedure visits one more leaf corresponding to a zero length item in order to compute the set R^0 of length positions reachable by the whole set P, and $ee\ell(P)$ accordingly (as in line 12 of Algorithm 1).

Algorithm 2 dynamic programming for computing $\pi(P)$ and $ee\ell(P)$

$\ell := \lceil \log_2 p \rceil$ // tree depth
$p_U := 2^\ell$ // tree # of leaves
$\tau := 2p_U$ // tree # of nodes
$\texttt{loss}[r'] := \max_{j\in P} e_j, \quad \forall r' \in [1, w]$
$w_j := 0 \quad \forall j \in [p+1, p_U]$
$\pi(P) := 0$
$R[0] := \emptyset$
for $i := 1$ **to** τ **and** $\texttt{tree}[i] < p_U + p$ **do**
  $\ell' := \log_2(\texttt{tree}[i]) + 1$ // tree level of the current node v
  $s := \texttt{tree}[i] + 1 - 2^{\ell'}$ // offset of v on level ℓ'
  $R[\ell'] := R[\ell' - 1]$
  $(j_L, j_U) := \textsc{SiblingItems}(\ell - \ell', s)$
  for $j := j_L$ **to** j_U **and** $w_j > 0$ **do**
    for all $r \in R[\ell']$ **and** $r \le w - w_j$ **do**
      $R[\ell'] := R[\ell'] \cup \{r + w_j\}$
    end for
    $R[\ell'] := R[\ell'] \cup \{w_j\}$
  end for
  if $\texttt{tree}[i] \ge p_U - 1$ **then**
    if $w_0^P + w_j = 0$ **then**
      $R[\ell'] := \emptyset$
    else
      $R[\ell'] := \textsc{UpdateLoss}(R[\ell'], w_0^P + w_j, e_j)$
    end if
    if $w_j = 0$ **then**
      $\pi(P) := \sum_{r\in R[\ell']} f(r)$
    end if
  end if
end for
$ee\ell(P) := \frac{1}{w} \sum_{r'\in[1,w]} \texttt{loss}[r']$
output $(\pi(P), ee\ell(P))$

function $\textsc{SiblingItems}(l, s)$
  //compute indices of items in $S(sibling(v))$
  if s is even **then**
    return $(2^l(s+1), 2^l(s+2) - 1)$
  else
    return $(2^l(s-1), 2^l s - 1)$
  end if
end function

6. Robust CSP solution

Being the computation of EEL and ETR (Problems 3 and 4) particularly complex, we address them heuristically starting from the trivial observation that pattern robustness normally increases with leftover. Indeed, as a general rule and in rough terms, the robustness of a pattern P against point-shaped faults increases with

- the pattern leftover w_0^P, and
- the pattern assortment, i.e. the number of different item lengths in P.

Let $w_0 = mw - \sum_{i\in I} w_i$ be the total leftover obtained after cutting I from m stocks. Concerning the pattern leftover, it makes sense to distribute w_0 as evenly as possible among the patterns of the cutting plan. See the following example:

Example 2. $I = \{1,2,3,4\}$, $w = 20$, $w_1 = w_2 = 10$, $w_3 = w_4 = 5$.

There are two minimum cutting plans that allow to cut all the items of I: $\mathcal{A}$, consisting of $P_1 = \{1,2\}$ and $P_2 = \{3,4\}$, and $\mathcal{B}$, consisting of $P_3 = \{1,3\}$ and $P_4 = \{2,4\}$ (or equivalent combinations like $\{1,4\}$ etc.). P_1 has 0 leftover, but is not robust at all as $\pi(P_1) = 0$. P_2, P_3, P_4 are instead totally robust, as $\pi(P_2) = \pi(P_3) = \pi(P_4) = 1$. Therefore

$$ee\ell(\mathcal{A}) > 0 \qquad ee\ell(\mathcal{B}) = 0.$$

So we have two solutions that are equivalent from the viewpoint of total leftover but not from that of potential value loss caused by a single random fault: value loss is minimized by the solution that distributes leftover among the stocks as fairly as possible. □

Observe however that equally distributing w_0 as far as possible among stocks may not be sufficient to maximize robustness. The following simple example illustrates the issue of pattern assortment, thus introducing a combinatorial view of the notion of robust pattern:

Example 3. $|I| = 12$, $w = 11$, $w_1 = w_2 = 5$, $w_3 = w_4 = 4, w_5 = w_6 = w_7 = w_8 = 2, w_9 = w_{10} = w_{11} = w_{12} = 1$.

It is easy to see that $\sum_{i\in I} w_i = 30$ and $m^* = 3$. Among the cutting plans that minimize the trim loss, at least two have the total leftover of 3 perfectly balanced among the three stocks:

$\mathcal{A}$: $P_1 = \{1,2\}$, $P_2 = \{3,4,5\}$, $P_3 = \{6,7,8,9,10,11,12\}$
$\mathcal{B}$: $P_4 = \{1,5,6,9\}$, $P_5 = \{2,7,8,10\}$, $P_6 = \{3,4,11,12\}$

The mean robustness of $\mathcal{A}$ is $\bar\pi(\mathcal{A}) = \frac{20}{33}$, smaller than $\bar\pi(\mathcal{B}) = \frac{31}{33}$, and indeed, the patterns of $\mathcal{B}$ have more variation in their lengths than those of $\mathcal{A}$. □

Indeed, we observe that stock leftovers are in some way subject to a trade-off: they must be sufficiently small to reduce the requirement of stocks, but also sufficiently large to maximize the chances of pattern recovery. In the following, we derive a sufficient condition of pattern reconfigurability that basically depends on the assortment of its items.

6.1. Assignment with bounded leftover

To derive an applicable criterion for the appropriate size of the leftover for one stock, we will state the following sufficient condition. Consider all the possible subsets L (including $\emptyset$) of a set P of items assigned to a single stock, and rank them according to non-decreasing $w(L)$. Let $0 = t_0 < t_1 < t_2 < \cdots < t_s$ be the distinct lengths $w(L)$ obtained in this way. To recover a defect falling within t_k and t_{k+1} one can then choose the $L \subseteq P$ such that $w(L) = t_k$, provided that the leftover $w_0^P = w - w(P)$ is at least $t_{k+1} - t_k$. In that case, the items $P \setminus L$ can be placed in the interval $[t_{k+1}, w]$. Hence, we can state the following sufficient condition for P to be reconfigurable.

Proposition 6. *Let P be the pattern assigned to a given stock. If the leftover fulfills*

$$w_0^P \ge \max_{0\le k<s} \{t_{k+1} - t_k\} =: \bar w_0^P \tag{9}$$

then P is reconfigurable.

A difficulty of formula (9) is that the number of t_k is exponential in p. However, assuming $w_1 \le w_2 \le \cdots \le w_p$, this drawback is partially overcome by the following simple bound:

Proposition 7. *Let P be a pattern assigned to a given stock with $\bar w_0^P$ given by (9). Then*

$$\bar w_0^P \le \max\{w_1, w_i - w_{i-1} : i \in P, i > 1\} =: \tilde w_0^P \tag{10}$$

Proof. Let $w(L) = t_k$, and $i \in L$. If $i = 1$, let $S = L - \{1\}$, otherwise $S = L - \{i\} \cup \{i-1\}$. In both cases, $w(S) \le w(L)$, and since $w(L) = t_k$, then $w(S) \le t_{k-1}$. Thus in the former case $t_k - t_{k-1} \le w(L) - w(S) = w_1$; in the latter, $\le w_i - w_{i-1}$ and hence the thesis. □

Combining Propositions 6 and 7, it is evident that a pattern P is reconfigurable if its leftover fulfills $w_0^P \geq \tilde{w}_0^P$. Imposing this condition for every stock would guarantee a reconfigurable packing of I. However, it may well lead to an exaggerated number of stocks. Thus, we tone down this requirement multiplying each bound $\tilde{w}_0^P$ by a parameter $\delta \in [0,1]$.

One can then formulate an assignment-type ILP-model for the problem of cutting all the items of I from a minimum number of stocks, with the condition that a minimum leftover $\delta \cdot \tilde{w}_0^P$ appears in each pattern P of a stock, so as to achieve a certain degree of pattern robustness. Of course, only $\delta = 1$ would guarantee robustness. However, we experimentally observed that $\delta = 0.5$ already guarantees solutions with a mean pattern robustness greater than 0.95, while the mean pattern robustness of the CSP solutions is under 0.2. A further discussion on the influence of δ is given in Section 7.

Recall $w_1 \leq \cdots \leq w_n$ and assume that a sufficiently large set M of stocks is available. We will use the following variables:

x^k = 1 if stock $k \in M$ is used, 0 else;
x_i^k = 1 if item $i \in I$ is cut from stock $k \in M$, 0 else;
y_i^k = 1 if $i \in I$ is the shortest item of stock $k \in M$, 0 else;
y_{ij} = 1 if items i and j, $i < j$ are cut from the same stock and no item h with $i < h < j$ is cut from that stock, 0 else;
w_0^k be a non-negative real variable indicating the leftover assigned to stock $k \in M$.

$$[\mathrm{RCS}]_\delta: \quad \min \sum_{k\in M} x^k \tag{11}$$

$$\sum_{k\in M} x_i^k = 1 \qquad i \in I \tag{12}$$

$$\sum_{i\in I} w_i x_i^k + w_0^k = w x^k \qquad k \in M \tag{13}$$

$$w_0^k + \delta(w_i - w_j)(y_{ij} + x_i^k - 1) \geq 0 \qquad i,j \in I: i<j; k \in M \tag{14}$$

$$-y_{ij} + x_i^k + x_j^k - \sum_{i<h<j} x_h^k \leq 1 \qquad i,j \in I: i<j; k \in M \tag{15}$$

$$w_0^k - \delta w_i (y_i^k + x_i^k - 1) \geq 0 \qquad i \in I; k \in M \tag{16}$$

$$-y_i^k + x_i^k - \sum_{h<i} x_h^k \leq 0 \qquad i \in I; k \in M \tag{17}$$

$$x^k, x_i^k, y_{ij}, y_i^k \in \{0,1\}, \quad w_0^k \in \mathbb{Q}_+ \qquad i,j \in I; k \in M$$

Constraints (12) ensure that every item of I is cut from exactly one stock. Constraints (13) state (*i*) that a stock is used whenever is assigned at least one item, and (*ii*) that every stock used has a non-negative leftover w_0^k.

Constraints (14) quantify the convenient leftover via the bound $\tilde{w}_0^P$ given by Proposition 7 scaled with $\delta \in [0,1]$. Notice that $[\mathrm{RCS}]_0$ is the classical assignment formulation of the CSP.

The definition of y_{ij} is implemented by constraints (15) that enforce $y_{ij} = 1$ if items i and j, but no item in between, are cut from the same stock. Constraints (16) and (17) implement the same conditions of (14) and (15), to cope with the case in which w_0^k is determined by the smallest item cut from stock k. Notice that item sizes $w_i > w/(1+\delta)$ make (16) infeasible. However, items with such sizes are always cut individually from distinct stocks in any solution compliant with Proposition 7, therefore they can be preprocessed and removed from I.

A solution to model $[\mathrm{RCS}]_1$ consists of guaranteed robust patterns only. This level of robustness is found at the expense of additional stocks compared to the minimum m^* of an optimal cutting plan. Notice that Proposition 7 states a condition of optimal robustness which is only *local* to the pattern, and only sufficient (and presumably too much restrictive). Local robustness in general does not guarantee a cutting plan to be *globally* robust, that is to minimize, e.g., the total expected economic loss. In fact, large stock leftovers increase on one hand the robustness of individual patterns, but on the other hand leftover minimization helps reduce the expected amount of discarded items, since the largest the number of stocks in a plan (and hence the largest the total leftover), the higher the chances that some stock will be faulty.

6.2. Equal distribution of leftovers

A different approach striving for robustness aims at an equal distribution of leftovers over all stocks. That means: given a cutting plan using a fixed set M of $m \geq m^*$ stocks, we would like to distribute the items "as equally as possible" among the m stocks. Such a solution, which improves the discussed trade-off between global and local robustness, can be obtained heuristically by solving a relative of a Multiprocessor Scheduling problem, namely minimizing the total deviation from the leftover averaged over the number $m = |M|$ of stocks.

$$[\mathrm{MS}]_m: \quad \min \sum_{k\in M} \left| w_0^k - \frac{m\,w - \sum_{i\in I} w_i}{m} \right| \tag{18}$$

$$\sum_{k\in M} x_i^k = 1 \qquad i \in I \tag{19}$$

$$\sum_{i\in I} w_i x_i^k + w_0^k = w \qquad k \in M \tag{20}$$

$$x_i^k \in \{0,1\}, \quad w_0^k \in \mathbb{Q}_+ \qquad i \in I, k \in M$$

Absolute values in the objective function can be linearized in a standard way. However, as previously observed, the equalization of w_0 among stocks may not be sufficient to maximize robustness, see Example 3.

In fact, the objective function (18) only surrogates the maximization of plan robustness which can be evaluated only *ex post* by computing the EEL and the mean pattern robustness of the solutions provided by $[\mathrm{MS}]_{|M|}$. Other loss equalization strategies, such as

$$\max \min_{k\in M} w_0^k \tag{21}$$

and

$$\min(\max_{k\in M} w_0^k - \min_{k\in M} w_0^k), \tag{22}$$

can be also considered; a preliminary computational experience showed that (18), (21) and (22) do not dominate each other in terms of both solution quality and computational costs, but (18) turned out the best trade-off.

Numerical experiments in Section 7 show the gain in robustness one can get with the above procedure.

7. A computational experience

Numerical tests were carried out to evaluate the performance of models $[\mathrm{RCS}]_\delta$ (see Section 6.1) and $[\mathrm{MS}]_{|M|}$ (see Section 6.2) for Problems 3 and 4 in terms of both total EEL reduction and improvement of ETR and mean pattern robustness with respect to a CSP solution that just minimizes the number of stocks used.

We assumed stocks prone to a defect with certainty, i.e., with identical probabilities $\rho = 1$, and defect position t uniformly distributed in the stock length, i.e., $f(t) = \frac{1}{w}$.

Pattern robustness $\pi(P)$ is computed by formula (5) where $\chi(P,t)$ is determined by the dynamic program implemented with Algorithm 2. The expected economic loss of a single pattern is computed by formula (6), where we assume item economic values to be equal to item lengths, i.e. $e = 1$. Summarizing, with the above assumptions formula (6) is rendered

$$ee\ell(P) = \frac{1}{w} \sum_{\bar{t}=1}^{w} \ell^*(P,\bar{t})$$

where we recall that $\ell^*(P,\bar{t})$ is the minimum economic loss if a defect occupies position $\bar{t}$ in pattern P. Expected economic loss $ee\ell(\cdot)$ and total revenue $etr(\cdot)$ of a solution are computed according to (2) and (3) with $c = w$. Since economic value and stock cost are identical to length, $etr(\cdot)$ will always be negative.

All integer programs derived from the models $[\mathrm{RCS}]_\delta$ and $[\mathrm{MS}]_{|M|}$ of Section 6 were solved by GuRoBi 9.1.2 with default setting and running time limited to 1800 seconds. Algorithm 2 was implemented in C++ and

Table 1
Features of instance sets from Scholl et al. (1997).

Group	Set	n	w	w_i
Group 1	N1C1W1	50	100	$\in [1, w]$
	N2C1W1	100	100	$\in [1, w]$
Group 2	N1W4B1	50	1000	$\in [0.8\frac{w}{9}, 1.2\frac{w}{9}]$
	N1W4B2	50	1000	$\in [0.5\frac{w}{9}, 1.5\frac{w}{9}]$
	N1W4B3	50	1000	$\in [0.1\frac{w}{9}, 1.9\frac{w}{9}]$
	N2W4B1	100	1000	$\in [0.8\frac{w}{9}, 1.2\frac{w}{9}]$
	N2W4B2	100	1000	$\in [0.5\frac{w}{9}, 1.5\frac{w}{9}]$
	N2W4B3	100	1000	$\in [0.1\frac{w}{9}, 1.9\frac{w}{9}]$

compiled using Microsoft *cl* ver. 19.14 with option */O2*. Tests were run on a virtual machine QUEMU CPU 1.5.3 (8 cores) at 2.30 MHz with 8 GB of RAM.

We tested 100 instances taken from Scholl et al. (1997) and organized in two groups with different features: Group 1 (Group 2) consists of two sets of twenty (of six sets of ten) instances each. Details are given in Table 1: the columns report, for each set, the number n of items, the stock length w and the interval from which the integer item sizes were randomly generated by Scholl et al. (1997). Sets are constructed so that in Group 1 (Group 2) the expected average number of items per stock is not larger than 3 (than 9).

The general approach of our solution method works as follows: Determine lower and an upper bounds m_L and m_U for the number of stocks which maximizes the ETR. For each value $m = m_L, \dots, m_U$, solve $[\text{MS}]_m$ and compute EEL and ETR of the packing obtained. Finally, select the solution with the largest ETR among the $m_U - m_L + 1$ candidates.

To implement this approach the experiments followed the baseline described below:

1. For each instance, we compute an upper bound m_U by solving model $[\text{RCS}]_1$. The lower bound m_L is straightforwardly given by the number m^* of patterns of an optimal CSP solution. Such a solution can be computed by $[\text{RCS}]_0$ or, more efficiently, by our adaptation of the Sequential Value Correction (SVC) algorithm described in Arbib et al. (2021) (that in fact happens to certify optimality for all the instances of the data set). In general, the SVC scheme was successfully employed in one-dimensional cutting stock problems (Belov and Scheithauer, 2007) and strip packing (Belov et al., 2008). The basic strategy is quite simple: given a *pseudo-price* λ_i for each item $i \in I$, stocks are cut sequentially, each one for obtaining the items not yet produced that maximize the sum of pseudo-prices. After a solution has been computed, pseudo-prices are conveniently updated and the process iterated until some halting criterion is fulfilled.
2. We solve model $[\text{MS}]_m$ for all $m \in \{m_L, \dots, m_U - 1\}$. For each solution $(P_1, \dots, P_m)$ so obtained we use Algorithm 2 to solve Problems 1 and 2 pattern by pattern. In this way we compute the mean pattern robustness $\bar{\pi}(P_1, \dots, P_m) = \frac{1}{m}\sum_{k\in M} \pi(P_k)$, the expected economic loss $eel(P_1, \dots, P_m)$ and the expected total revenue $etr(P_1, \dots, P_m)$.
   Indeed, instead of solving $[\text{MS}]_{m_U}$, we just take the solution obtained from $[\text{RCS}]_1$ in Step 1 which is reconfigurable by construction.

In the following, we report in Table 2 detailed results for computing the lower bound m_L and solving the corresponding problem $[\text{MS}]_{m_L}$ (rows report mean values taken in each class of instances). Table 2 shows:

- The number m_L of stocks used, together with the initial mean pattern robustness $\bar{\pi}_{CS}$ and expected total revenue etr_{CS} of the optimal CSP solutions computed by SVC (columns 2 to 4);
- The final mean pattern robustness $\bar{\pi}_{MS}$ and expected total revenue etr_{MS} of the solutions obtained by $[\text{MS}]_{m_L}$ (columns 6 and 7);
- The running time taken by the SVC algorithm to solve CSPs (column 5), and by GuRoBi to solve $[\text{MS}]_{m_L}$ (column 8). For the dynamic program to compute the required values of π and etr we report the total running time (in milliseconds) taken by the m_L executions of Algorithm 2 (column 11).
- Columns 9 and 10 report the improvement in terms of pattern robustness and expected total revenue of $[\text{MS}]_{m_L}$ versus the standard stock cutting solutions, computed as:

$$G_{\bar{\pi}} = \frac{\bar{\pi}_{MS}}{\bar{\pi}_{CS}} \qquad G_{etr} = 100 \cdot \frac{etr_{CS} - etr_{MS}}{etr_{CS}}$$

Table 3 provides analogous information for the other extreme point, namely the solution for m_U derived by model $[\text{RCS}]_1$. Solving this model turned out to be much harder than $[\text{MS}]_{|M|}$, with a computation time exceeding the time limit of 1800 s for the majority of instances. Therefore, instead of the CPU time, we report the mean optimality gap G_{opt} of the solutions taken over all instances (column 5) and the number *#limit* of instances (out of 20 resp. 10) where GuRoBi reached the time limit (column 6). Relative improvements in columns 7 and 8 refer to the values of $[\text{RCS}]_1$ compared to the CSP.

Between these two extreme points we noted that the relevant parameters behave just as they can be expected to do. The mean pattern robustness $\bar{\pi}_{MS}$ increases, while the expected economic loss $eel(\cdot)$ decreases with increasing stock number. The behavior of the ETR is less predictable and will be discussed later. As an intermediate point of reference we considered $[\text{RCS}]_{0.5}$ and report in Table 4 values analogous to Table 3. For every instance $[\text{MS}]_m$ was run for the number of stocks m returned by $[\text{RCS}]_{0.5}$. It can be seen that even if the sufficient condition for reconfigurability (14) is reduced considerably by a factor $\delta = 0.5$, the mean pattern robustness $\bar{\pi}_{RCS}$ already exceeds 0.95 for Group 2 instances while it ranged in $[0.1, 0.2]$ for $\delta = 0$. For Group 1 instances, the increase of $\bar{\pi}_{RCS}$ is less impressive, but still quite near to the best values reached for $\delta = 1$ (note that $\pi = 1$ cannot be reached for instances with very large items). It is interesting to note that $[\text{MS}]_m$ further improves the values of $\bar{\pi}_{RCS}$ and etr_{RCS} although $\bar{\pi}_{RCS}$ should be maximal by construction. This can be explained by the effect of $\delta = 0.5$.

Let us now comment in more detail the results obtained. First of all, we point out that the ETR is always negative because, not referring to a specific process – features can in fact vary a lot depending on application –, we consider the case of a process with no value-added (that is, item values correspond to item lengths) and one defect per stock ($\rho = 1$). This scenario does not correspond directly to any practical case but can be used as benchmark as we will see later. What is here relevant, instead, is to measure the ETR improvement one can obtain with the proposed approach. Instances of Group 1 are quite different from those of Group 2: unlike the latter, they consist of highly heterogeneous items, often longer than $w/2$, see Table 1. Such a difference has, in the first place, an impact on the optimal solutions of the CSP, both in the number of stocks used (on average, 39.6 for Group 1 vs. 8.86 for Group 2) and in the ETR (on average, −22.63 for Group 1 vs. −1.15 for Group 2). Similar gaps are also observed in the solutions of $[\text{MS}]_{m_L}$ and $[\text{RCS}]_1$, see Columns etr_{MS} resp. etr_{RCS} in Table 2 resp. 3. The ranges $[m_L, m_U]$ are very different as well: quite large in Group 1 (11.93 stocks on average), they reduce to very few stocks (1.12 on average) in Group 2.

Let us now focus on Table 2. Columns $G_{\bar{\pi}}$ and G_{etr} show the effectiveness of the $[\text{MS}]_{m_L}$ model in providing better solutions in terms of both robustness and ETR using the same number of stocks. Again, however, the difference of performance between Group 1 and 2 is remarkable. In fact, though CSP solutions have roughly similar values of $\bar{\pi}$ (on average, 0.12 in Group 1 and 0.14 in Group 2), the mean pattern robustness of $[\text{MS}]_{m_L}$ solutions increases by 1.6 times in Group

Table 2
Results for $\delta = 0.0$.

Set	SVC				$[MS]_{m_L}$					DP
	m_L	$\bar{\pi}_{CS}$	etr_{CS}	T (sec)	$\bar{\pi}_{MS}$	etr_{MS}	T (sec)	$G_{\bar{\pi}}$	G_{etr} (%)	T (msec)
N1C1W1	27.25	0.14	−15.73	0.052	0.22	−15.36	5.21	1.54	2.33	1.24
N2C1W1	51.95	0.09	−29.52	0.119	0.14	−29.12	5.44	1.66	1.30	2.50
N1W4B1	6.00	0.20	−0.98	0.013	0.98	−0.52	98.20	5.01	47.01	4.36
N1W4B2	6.00	0.18	−0.84	0.009	0.92	−0.50	280.52	5.31	43.29	5.82
N1W4B3	6.73	0.16	−1.03	0.007	0.98	−0.59	6.53	6.26	43.54	6.36
N2W4B1	11.60	0.13	−1.51	0.030	0.79	−0.77	86.99	6.33	48.78	7.52
N2W4B2	11.50	0.10	−1.32	0.006	0.90	−0.56	62.16	9.12	59.39	10.28
N2W4B3	11.33	0.09	−1.20	0.011	0.84	−0.44	48.10	9.62	64.81	12.91

Table 3
Results for $\delta = 1.0$.

Set	$[RCS]_1$							DP
	m_U	$\bar{\pi}_{RCS}$	etr_{RCS}	G_{opt} (%)	#limit	$G_{\bar{\pi}}$	G_{etr} (%)	T (msec.)
N1C1W1	34.25	0.62	−19.84	4.40	15	5.63	−27.45	0.74
N2C1W1	68.80	0.65	−39.40	9.18	20	9.64	−34.78	1.38
N1W4B1	6.80	1.00	−1.31	11.43	8	5.13	−35.73	2.37
N1W4B2	6.50	1.00	−0.97	4.29	3	5.84	−35.14	3.23
N1W4B3	7.55	1.00	−1.39	8.33	6	6.41	−50.03	4.79
N2W4B1	12.90	1.00	−1.84	7.76	10	8.69	−31.79	5.44
N2W4B2	12.70	1.00	−1.68	9.38	10	10.18	−33.27	6.97
N2W4B3	13.44	1.00	−2.43	14.72	10	11.49	−116.88	8.98

Table 4
Results for $\delta = 0.5$.

Set	$[RCS]_{0.5}$					$[MS]_m$				
	m	$\bar{\pi}_{RCS}$	etr_{RCS}	G_{opt} (%)	#limit	$\bar{\pi}_{MS}$	etr_{MS}	T (sec.)	$G_{\bar{\pi}}$	G_{etr} (%)
N1C1W1	31.60	0.54	−17.66	3.06	17	0.56	−17.53	4.27	4.85	−12.81
N2C1W1	64.90	0.59	−36.28	14.36	20	0.61	−35.92	51.69	8.84	−24.39
N1W4B1	6.00	0.97	−0.52	0.00	0	0.97	−0.52	3.80	4.97	46.62
N1W4B2	6.20	0.97	−0.69	0.00	0	0.99	−0.67	7.51	5.62	4.55
N1W4B3	6.91	0.95	−0.79	2.60	2	0.99	−0.77	9.91	6.10	9.67
N2W4B1	12.00	0.95	−1.00	0.83	3	0.95	−1.00	58.03	8.30	26.19
N2W4B2	12.00	0.96	−1.03	4.10	6	0.99	−0.99	81.65	9.78	14.13
N2W4B3	12.11	0.97	−1.15	4.49	8	1.00	−1.10	84.33	11.09	−6.05

1 ($\bar{\pi}_{MS} = 0.18$ on average) and by 6.94 times in Group 2 ($\bar{\pi}_{MS} = 0.90$ on average), meaning that almost all patterns are reconfigurable.

The low robustness of Group 1 instances mainly depends on the presence of very large sizes (those with $w_i > w/2$) which make the pattern certainly not t-reconfigurable for some positions t (preventing, *de facto*, the realization of reconfigurable patterns). The improvement of $\bar{\pi}$ obtained by model [MS] implies a corresponding improvement of the ETR, by 1.82% in Group 1 (increasing from −22.63 of the stock cutting solutions to −22.54 of the $[MS]_{m_L}$ solutions) and by 51.13% in Group 2 (increasing from −1.15 to −0.56). To conclude with Table 2, the relative homogeneity of item sizes in Group 2 has also an impact on the CPU time required to solve $[MS]_{m_L}$, that is on average some 18 times that spent for solving the problems of Group 1 (see column 8).

Let us now comment on Table 3. The maximum $\bar{\pi}$ obtained by $[RCS]_1$ is, on average, 0.63 in Group 1 and 1.00 in Group 2 (see col 3). The instances of Group 1 do not reach full pattern reconfiguration due the presence of sizes $> w/2$. Furthermore, the impact of the restricted number m_L of stocks is much more relevant in Group 1 than in Group 2: indeed, the $\bar{\pi}$ of the former is only 0.286 the maximum computed via $[RCS]_1$, whereas the $\bar{\pi}$ of the latter is already 0.90 the maximum, being almost all patterns of Group 2 solutions already reconfigurable using $[MS]_{m_L}$.

Although the solutions of $[RCS]_1$ exhibit the largest pattern-by-pattern robustness, they are uneconomical under the item values e_i and defect probability ρ adopted in our experiments. In particular, compared to that obtained with $|M| = m_L$ stocks, the etr worsens on average by 31.12% in Group 1 and by 50.47% in Group 2. In fact, the composition of the total loss (see Fig. 3) shows that most pattern robustness is achieved by increasing the leftover: on average, the proportion of leftover and EEL is 28.86% vs. 71.14% in solutions with $\delta = 0$ and 79.84% vs. 20.16% in solutions with $\delta = 1$ (maximally robust patterns). Again, the frequency of very large sizes in Group 1 causes the proportion to be 8.99% vs. 91.01% for $\delta = 0$ and 47.13% vs. 52.87% for $\delta = 1$, whereas the EEL of Group 2 is much reduced with $\delta = 0$ (the proportion is 41.93% vs. 58.07%), and equals zero for $\delta = 1$.

Though not convenient in the benchmark scenario analyzed (item values $e_i = w_i$, stock defectiveness probability $\rho = 1$), the opportunity of improving the ETR by using more stocks than m_L is worth being considered in a general setting. For example, Fig. 4 reports the ETR obtained for $e_i = 2.5w_i$ and $\rho = 0.8$ from the 40 Group 1 instances, plotted as a function of the number of stocks used. One can observe the expected concave behavior of this function:

- If m is small, all the stocks are almost completely used, hence the probability that the associated patterns can be reconfigured is small: although raw material cost is minimized, revenue is sensibly reduced.
- If m is large, an increased slack allows to arrange items so that reconfiguration is more likely: expected economic loss is so reduced, but this comes at the expense of more raw material.

Depending on the ratio between economic value of items and cost of stocks, the optimal number of stocks can be determined as a trade-off between these two extreme cases.

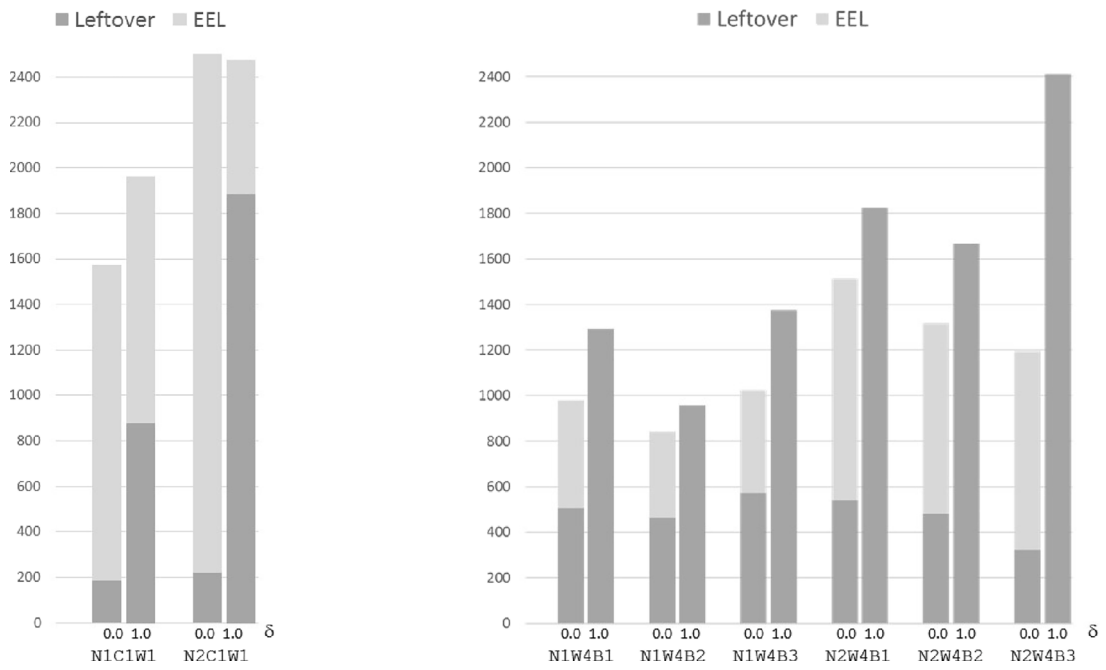

Fig. 3. Total loss for $\delta = 0.0$ and $\delta = 1.0$.

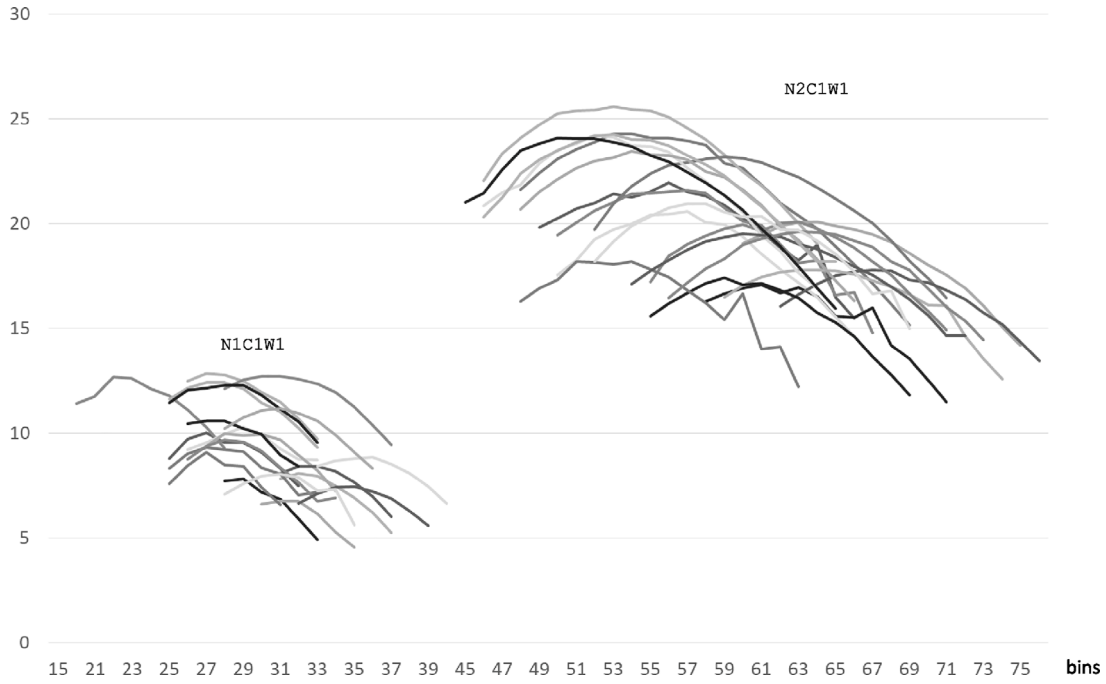

Fig. 4. *etr* for different numbers of stocks (Group 1 instances with item values $e_i = 2.5w_i$ and $\rho = 0.8$).

A final remark on the evaluation of $\pi(P)$ and EEL. Column 11 of Table 2 and column 9 of Table 3 report the total CPU time (in milliseconds) for computing the EEL and the robustness $\pi(P)$ of all the patterns of a solution (averaged on the number of instances per group). These computations were done by the dynamic programming Algorithm 2. As we elaborated in detail in Section 5.3, this tree-based algorithm should improve upon the standard version described in Algorithm 1.

Indeed, for the case of $\delta = 0.0$, the overall sum of computation times spent for Algorithm 1 exceeded the times for Algorithm 2 by a factor of 1.94. In this case, a total of 2111 cutting patterns were evaluated twice (for SVC and [MS]). For the case of $\delta = 1.0$, this factor goes down to 1.56 for a total of 2654 pattern computations. It seems that, in the latter case, a smaller number of items per pattern reduces the advantage of the more sophisticated Algorithm 2. As a whole, dynamic programming was used to evaluate 6876 patterns in a total CPU time of 897 ms (Algorithm 2) and in 1602 ms (Algorithm 1), i.e. an increase by

a factor of ≈ 1.78. However, it has to be said that most patterns contain only a fairly small number of items, and thus the computation times are very short and therefore sensitive to programming style.

8. Conclusions and future research

We addressed the problem of computing robust cutting patterns in the setting of one-dimensional stock cutting where stock pieces may be affected by a defect that makes unusable a unit length of material. The robustness of a pattern, (i.e., a set of items assigned to a single stock) is defined as the probability that all its items can be cut so as to avoid the defective unit, which is distributed arbitrarily or uniformly at random along the length of the stock.

For a given pattern, we considered the occurrence of a defect at some integer position t and determined algorithms for deciding if the pattern is t-reconfigurable, i.e. all items can be obtained avoiding position t, or, if this is not the case, for finding the most advantageous subset of items that can still be cut from the stock. Iterating over all possible defect positions, these algorithms can be used to determine the expected economic loss of the pattern. The latter computation can be done in one run of a dynamic programming approach for which we derived several algorithmic improvements.

Moving from one pattern to the general cutting stock problem (CSP), the trade-off between number of stocks (raw material) used and the expected loss incurred gives rise to a variant of CSP, where the total expected revenue is maximized. Using more stocks with lower utilization makes the patterns of each stock more robust and reduces expected economic loss, but incurs higher material cost. We presented two different ILP-models for resolving this trade-off. Finally, we set up numerical tests to check the validity of the proposed methods and compare the efficacy of different solution approaches.

The computational effort required for the problems here considered leaves room for improvements of the solution methodologies in the future. Since our ILP-models are built upon rather straightforward models for the standard stock cutting problem, we could proceed with a column generation approach, which is know to be highly effective for stock cutting. Moreover, investigating possible approximation algorithms would be an interesting perspective.

From a more general perspective, several research directions stem from the present work. First, we just regarded as rare the event of more than one fault in a stock, so a natural extension would be one where such an assumption is relaxed. Secondly, our model assumes point-shaped faults, whereas this might not be the case in relevant industrial applications: Some of our results easily extend to faults of larger size, as long as it is known; but treating the fault size as a random variable poses new challenges. Third, we just considered one-dimensional cutting processes, thus leaving important application settings in two dimensions uncovered. In fourth place, we limited our analysis to a recourse strategy which does not reassign items to stocks but just modifies the item sequence in each particular faulty stock.

In fact, phase 3 of the timeline described in the introduction assumes that the pre-computed sequence of patterns cannot be changed online. Indeed, for several reasons, such a setting is quite common in real applications. On one hand, if items must be collected in orders, and orders have due dates, efficient approaches combine the minimization of material usage with the minimization of some scheduling performance indicators, e.g., lateness, Arbib et al. (2021), Arbib and Marinelli (2017, 2014), Marinelli and Pizzuti (2018), whose values depend on the sequence of cutting patterns. Therefore any *ex post* adjustment of an optimal sequence could affect the quality of the solutions. On the other hand, specific features of cutting machines (e.g., a limited availability of out-buffers) could restrict feasible sequences of cutting patterns (Arbib et al., 2016, 2012) and therefore changing the initial sequence could lead to infeasible solutions. However, different and possibly more elaborated strategies can be considered, also depending on the application. Generally speaking, it would also be interesting to consider in future research a scenario where the patterns of a CSP solution are indeed computed in advance, but are only sequenced after all stocks are inspected and defects detected. In that scenario, patterns could be matched to (defective) stocks in the best possible way, still rearranging the items assigned to each defective stock in order to minimize economic loss.

CRediT authorship contribution statement

Claudio Arbib: Conceptualization, Methodology, Software, Validation, Writing – review & editing. **Fabrizio Marinelli:** Conceptualization, Methodology, Software, Validation, Writing – review & editing. **Ulrich Pferschy:** Conceptualization, Methodology, Software, Validation, Writing – review & editing. **Fatemeh K. Ranjbar:** Software, Validation.

Data availability

Data will be made available on request

Acknowledgments

We gratefully acknowledge Stefan Lendl for useful discussions on this topic. Ulrich Pferschy acknowledges support by the field of excellence "COLIBRI" of the University of Graz. Fatemeh K. Ranjbar was funded by Italian Ministry of Education, PON 2014–2020, CCI 2014IT16M2OP005. We are also very grateful to the anonymous referees for insightful comments that helped us to significantly improve the presentation of this work.

References

Aboudi, R., Barcia, P., 1998. Determining cutting stock patterns when defects are present. Ann. Oper. Res. 82, 343–354.

Afsharian, M., Niknejad, A., Wäscher, G., 2014. A heuristic, dynamic programming-based approach for a two-dimensional cutting problem with defects. OR Spectrum 36 (4), 971–999.

Alem, D.J., Morabito, R., 2012. Production planning in furniture settings via robust optimization. Comput. Oper. Res. 39, 139–150.

Alem, D.J., Munari, P.A., Arenales, M.N., Ferreira, P.A.V., 2010. On the cutting stock problem under stochastic demand. Ann. Oper. Res. 179, 169–186.

Alves, C., de Carvalho, J.M.V., 2008. A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. Comput. Oper. Res. 35 (4), 1315–1328.

Arbib, C., Marinelli, F., 2014. On cutting stock with due dates. Omega 46, 11–20.

Arbib, C., Marinelli, F., 2017. Maximum lateness minimization in one-dimensional bin packing. Omega 68, 76–84.

Arbib, C., Marinelli, F., Pezzella, F., 2012. An LP-based tabu search for batch scheduling in a cutting process with finite buffers. Int. J. Prod. Econ. 2 (136), 287–296.

Arbib, C., Marinelli, F., Pınar, M.Ç., Pizzuti, A., 2022a. Assortment and cut of defective stocks by bilevel programming. In: Proc. of the 11th Int. Conf. on Operations Research and Enterprise Systems. pp. 294–301.

Arbib, C., Marinelli, F., Pınar, M.Ç., Pizzuti, A., 2022b. Robust stock assortment and cutting under defects in automotive glass production. Prod. Oper. Manage. http://dx.doi.org/10.1111/poms.13812.

Arbib, C., Marinelli, F., Pizzuti, A., 2021. Number of bins and maximum lateness minimization in two-dimensional bin packing. European J. Oper. Res. 291 (1), 101–113.

Arbib, C., Marinelli, F., Ventura, P., 2016. One-dimensional cutting stock with a limited number of open stacks: bounds and solutions from a new integer linear programming model. Int. Trans. Oper. Res. 1–2 (23), 47–63.

Belov, G., Scheithauer, G., 2007. Setup and open-stack minimization in one-dimensional stock cutting. INFORMS J. Comput. 19 (1), 27–35.

Belov, G., Scheithauer, G., Mukhacheva, E.A., 2008. One-dimensional heuristics adapted for two-dimensional rectangular strip packing. J. Oper. Res. Soc. 59 (6), 823–832.

Beraldi, P., Bruni, M.E., Conforti, D., 2009. The stochastic trim-loss problem. European J. Oper. Res. 197 (1), 42–49.

Carnieri, C., Mendoza, G.A., Luppold, W.G., 1993. Optimal cutting of dimension parts from lumber with a defect: A heuristic solution procedure. Forest Prod. J. 43 (9), 66–72.

Cherri, A.C., Cherri, L.H., Oliveria, B.B., Oliveria, J.F., Carravilla, M.A., 2023. A stochastic programming approach to the cutting stock problem with usable leftovers. European J. Oper. Res. 308 (1), 38–53.

Christofides, N., Whitlock, C., 1977. An algorithm for two-dimensional cutting problems. Oper. Res. 25 (1), 30–44.
Dell'Amico, M., Delorme, M., Iori, M., Martello, S., 2019. Mathematical models and decomposition methods for the multiple knapsack problem. European J. Oper. Res. 274 (3), 886–899.
Delorme, M., Iori, M., Martello, S., 2016. Bin packing and cutting stock problems: Mathematical models and exact algorithms. European J. Oper. Res. 255 (1), 1–20.
Durak, B., Tüzün, D.A., 2017. Dynamic programming and mixed integer programming based algorithms for the online glass cutting problem with defects and production targets. Int. J. Prod. Res. 55 (24), 7398–7411.
Gaivoronski, A.A., Lisser, A., Lopez, R., Xu, H., 2011. Knapsack problem with probability constraints. J. Global Optim. 49, 397–413.
Garey, M., Johnson, D., 1979. Computers and Intractability; A Guide to the Theory of NP-Completeness. W.H. Freeman.
Ghodsi, R., Sassani, F., 2005. Real-time optimum sequencing of wood cutting process. Int. J. Prod. Res. 43 (6), 1127–1141.
Hahn, S.G., 1968. On the optimal cutting of defective sheets. Oper. Res. 16 (6), 1100–1114.
Hwang, J.Y., Kuo, W., Ha, C., 2011. Modeling of integrated circuit yield using a spatial nonhomogeneous Poisson process. IEEE Trans. Semicond. Manuf. 24 (3), 377–384.
Ide, J., Tiedemann, M., Westphal, S., Haiduk, F., 2015. An application of deterministic and robust optimization in the wood cutting industry. 4OR 13, 35–57.
Kellerer, H., Pferschy, U., Pisinger, D., 2004. Knapsack Problems. Springer.
Koiliaris, K., Xu, C., 2019. Faster pseudopolynomial time algorithms for subset sum. ACM Trans. Algorithms 15 (3), 1–20, Article No. 40.
Krichagina, E.V., Rubio, R., Taksar, M.I., Wein, L.M., 1998. A dynamic stochastic stock-cutting problem. Oper. Res. 46 (5), 690–701.
Malaguti, E., Monaci, M., Paronuzzi, P., Pferschy, U., 2019. Integer optimization with penalized fractional values: The knapsack case. European J. Oper. Res. 273, 874–888.
Marinelli, F., Pizzuti, A., 2018. A sequential value correction heuristic for a bi-objective two-dimensional bin-packing. Electron. Notes Discrete Math. 64, 25–34.
Monaci, M., Pferschy, U., Serafini, P., 2013. Exact solution of the robust knapsack problem. Comput. Oper. Res. 40, 2625–2631.
Neidlein, V., Vianna, A.C.G., Arenales, M.N., Wäscher, G., 2009. Two-dimensional guillotineable-layout cutting problems with a single defect – an AND/OR-graph approach. In: Operations Research Proceedings 2008, Part 3. Springer, pp. 85–90.
Özdamar, L., 2000. The cutting-wrapping problem in the textile industry: optimal overlap of fabric lengths and defects for maximizing return based on quality. Int. J. Prod. Res. 38, 1287–1309.
Perez-Salazar, S., Singh, M., Toriello, A., 2022. Adaptive bin packing with overflow. Math. Oper. Res. http://dx.doi.org/10.1287/moor.2021.1239, Published online in Articles in Advance 25 Feb 2022.
Pernkopf, M., Riegler M. Gronalt, M., 2019. Profitability gain expectations for computed tomography of sawn logs. Eur. J. Wood Wood Prod. 77, 619–631.
Petutschnigg, A., Pferschy, U., Sattler, L., 2007. Influence of production costs on cutting optimization in window frame production - a graph-theoretical model. Comput. Electron. Agric. 58, 133–143.
Petutschnigg, A., Schwarzbauer, P., Pferschy, U., 2005. Material flow simulation to support site planning of a sawmill with an installed computer tomograph - a case study. Paper and Timber (Paperi Ja Puu) 87, 47–52.
Rönnqvist, M., 1995. A methods for the cutting stock problem with different qualities. European J. Oper. Res. 83, 57–68.
Rönnqvist, M., Åstrand, E., 1998. Integrated defect detection and optimization for cross cutting of wooden boards. European J. Oper. Res. 108, 490–508.
Sarker, B.R., 1988. An optimum solution for one-dimensional slitting problems: a dynamic programming approach. J. Oper. Res. Soc. 39, 749–755.
Schepler, X., Rossi, A., Gurevsky, E., Dolgui, A., 2022. Solving robust bin-packing problems with a branch-and-price approach. European J. Oper. Res. 297 (3), 831–843.
Scholl, A., Klein, R., Juergensen, C., 1997. BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. Comput. Oper. Res. 24, 627–645.
Sculli, D., 1981. A stochastic cutting stock procedure: Cutting rolls of insulating tape. Manage. Sci. 27 (8), 946–952.
Sierra-Paradinas, M., Soto-Sánchez, O., Alonso-Ayuso, A., Martín-Campo, J., Gallego, M., 2021. An exact model for a slitting problem in the steel industry. European J. Oper. Res. 295 (1), 336–347.
Wäscher, G., Haußner, H., 2007. An improved typology of cutting and packing problems. European J. Oper. Res. 183 (3), 1109–1130.
Wenshu, L., Dan, M., Jinzhuo, W., 2015. Study on cutting stock optimization for decayed wood board based on genetic algorithm. Open Autom. Control Syst. J. 7, 284–289.
Woeginger, G.J., Yu, Z., 1992. On the equal-subset-sum problem. Inform. Process. Lett. 42, 299–302.