EURO Journal on Computational Optimization (2020) 8:263–288
https://doi.org/10.1007/s13675-020-00127-8

ORIGINAL PAPER

A hierarchical approach for solving an integrated packing and sequence-optimization problem in production of glued laminated timber

Heiner Ackermann[1] · **Erik Diessel**[1]

Received: 4 November 2019 / Accepted: 22 June 2020 / Published online: 12 July 2020

Abstract
Integrated packing and sequence-optimization problems appear in many industrial applications. As an example of this type of problem, we consider the production of glued laminated timber (glulam) in sawmills: Wood beams must be packed into a sequence of pressing steps subject to packing constraints of the press and subject to sequencing constraints. In this paper, we present a three-stage approach for solving this hard optimization problem: Firstly, we identify alternative packings for small parts of an instance. Secondly, we choose an optimal subset of these packings by solving a set cover problem. Finally, we apply a sequencing algorithm in order to find an optimal order of the selected subsequences. For every level of the hierarchy, we present tailored algorithms, analyze their performance and illustrate the efficiency of the overall approach by a comprehensive numerical study.

Keywords Production planning · Packing problems · Scheduling · Heuristics

Mathematics Subject Classification 90B30 · 90B90 · 90C59 · 90B35

1 Introduction

Cutting and packing problems arise in many industrial applications, especially in production and logistics. Such problems can be characterized in various ways, for example by their dimension (one, two or three dimensional), by the shape of the items (e.g., rectangular, circular or irregular), by the structure of the packing constraints

✉ Erik Diessel
erik.diessel@itwm.fraunhofer.de

Heiner Ackermann
heiner.ackermann@itwm.fraunhofer.de

[1] Fraunhofer ITWM, Institute for Industrial Mathematics, Fraunhofer Platz 1, 67663 Kaiserslautern, Germany

and allowed patterns (e.g., guillotine cuts, orthogonal cuts, …) and by the type of the assortment (e.g., knapsack, bin-packing or cutting stock). For a comprehensive introduction to cutting and packing problems we refer the reader to Dyckhoff (1990).

Especially in production however, one additionally needs to consider setup times between the processing of consecutive packings. In general, such sequencing problems can be described as scheduling problems with sequence-dependence setup times (Allahverdi 2015). The simultaneous consideration of both aspects gives rise to *integrated packing and sequence-optimization problems*.

In this paper, we introduce one such problem arising in the production of glued laminated timber (glulam) in sawmills: Wood beams of different height and length shall be packed into a sequence of pressing steps subject to certain packing constraints of the press and subject to requirements on subsequent pressing steps. The packing problem can be described as a two-dimensional packing problem with exact guillotine cuts. The objective is to minimize the amount of material (waste) required in order to fill up the entire press. The sequence-optimization problem requires to produce all beams for one customer just-in-sequence. The objective is to minimize the number of height changes of subsequent pressing steps as each height change requires some setup time.

In this paper we describe a three-stage heuristics for this problem. The approach can be summarized as follows:

1. Identify alternative sequences of pressing steps for various combinations of up to three orders using packing heuristics derived from the *largest-difference method* (Karmarkar and Karp 1983).
2. Select some of the sequences such that each order is contained in exactly one. This can be achieved using a set covering approach.
3. Compute an optimal order of the selected sequences using a scheduling algorithm.

We motivate and justify our approach by structural insights on the problem and conclude with a comprehensive numerical study. Our study shows that the heuristics can quickly compute efficient production plans for real-world instances.

1.1 Introduction to glued laminated timber

Glued laminated timber (glulam) is an engineered wood product made of thin slats in sawmills. It is used whenever long and thick wood beams of high quality are required (Serrano 2003), e.g., in roof construction. Such beams cannot be sawed from a single trunk as they would twist and rip. Combining thin slats however allows to circumvent such effects. Moreover, it allows to substantially increase the material efficiency in sawmills.

Glulam is made by stacking and glueing thin slats of standardized thickness. By varying the number of slats and their length it is possible to produce beams of almost every dimension. Note that the width of the slats determines the width of the beams. The slats are sawed from trunks, dried and finally checked for defects. By finger-jointing slats it is possible to produce slats of almost arbitrary length. Afterward the slats are stacked with glue in between and pressed. The press perfectly aligns the slats and triggers hardening of the glue. By not inserting glue between two slats it is possible

to produce two (or even more) beams in one pressing step. The press however comes with the following packing constraints:

1. All beams simultaneously processed in one pressing step must be of the same dimension, i.e., of the same width and length.
2. The height of a pressing step, i.e., the total height of all beams, must not fall below (exceed) the minimum (maximum) pressing height.
3. The length of the beams must not fall below (exceed) the minimum (maximum) pressing length.
4. It requires some setup time whenever two consecutive pressing steps have different heights. This is why one searches for sequences of pressing steps with few height changes, as the setup times after height changes dominate the time needed for the whole pressing.

In order to reduce production efforts, glulam production is performed in batches. Within each batch, slats of the same width are processed only. However, note that within a batch glulam of different length and height may be produced.

These constraints give not yet rise to a very complex optimization problem. The difficulty steams from the fact that customers typically order multiple beams of different lengths. To this end, the sawmill could produce few standard lengths to stock and cut the requested lengths on demand. However, it turns out that this would generate huge amounts of waste. For that reason it makes much more sense to rely on a make-to-order production policy and directly produce the requested lengths.

Doing so generates various questions related to optimal batch sizes, optimal supply of slats, etc. In this paper however, we focus on computing optimal sequences of pressing steps. That is, we assume that a feasible set of orders (all ask for beams of the same width) have been selected and that a sufficient number of slats of this width are available.

The task then is to find a sequence of pressing steps subject to the constraints described above and additional constraints and opportunities as described next.

1. Multiple customers place orders. Each order requires the production of beams with a shared height but different lengths. This is why the same customer may place multiple orders with different heights.
2. Within a batch, all beams belonging to an order must be produced just-in-sequence, i.e., beams belonging to an order should not be interrupted by beams belonging to other orders. Similarly, all orders belonging to one customer should be produced just-in-sequence as well, that is without orders from other customers processed in between. This is required because of limited capacities in subsequent packaging and loading.
3. It is possible and often required because of the minimum length constraint, to combine several order lengths into one production length and cut it into the order lengths afterward.
4. It is feasible to lengthen slats with filling material in order not to violate the packing constraints of the press. This material will be cut off afterward and considered as waste as often it is too short for further usage. This is why one searches for pressing steps with few filling material.

An example of a pressing step is depicted in Fig. 1.

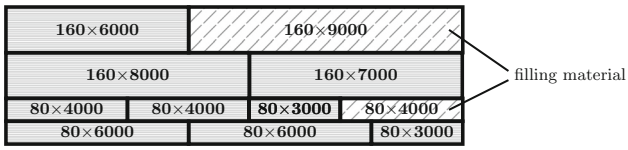

Fig. 1 A feasible pressing step for two orders: an order with height 80 of two pieces with length 3000, two pieces with length 4000 and two pieces of length 6000 and an order with height 160, containing one piece each of length 6000, 7000 and 8000. In total, filling material of $160 \cdot 9000 + 80 \cdot 4000 = 1{,}760{,}000$ units is used

As already indicated above there are two objectives to take into account. Firstly, minimize the amount of filling material (waste) in order to fill up the entire press. Secondly, minimize the number of height changes between consecutive pressing steps within a batch. In general, two objectives give rise to a bi-objective optimization problem. Below, however we present an approach that finds sequences of pressing steps that are efficient with respect to both objectives simultaneously.

1.2 Problem description

For describing our problem, the following notation will be useful. A multiset is a collection of elements where elements can occur multiple times. We denote the number of elements of a multiset X counted with multiplicities by $|X|$ and the sum of all contained numbers (also counted with multiplicities) by $||X|| := \sum_{x \in X} x$. We assume that multisets are always encoded as lists (i.e., no high-multiplicity encoding is used). We use the notation $\mathcal{M}(S)$ for the set of all multisets over a given set S. For the indicator function with respect to a condition C, we use the notation $\mathbb{1}(C)$. For our press parameters, we use the following notation which is illustrated in Fig. 2:

- the minimal pressing length is denoted by $\ell_{\min}$,
- the maximal pressing length by $\ell_{\max}$,
- the minimal pressing height is given by $h_{\min}$
- and the maximal pressing height by $h_{\max}$.

We now give a formal definition of the glulam production problem described above. In the following, we use the term *piece* for a beam of wood ordered by a customer.

Definition 1 *(Glulam production problem)* **Input:** The input of the glulam production problem is a set of orders which is grouped by corresponding customers. An order (h, L) is a pair of a height $h \in \mathbb{N}$ and a multiset $L \in \mathcal{M}(\mathbb{N})$ of piece lengths. In the input, L is given as a list without multiplicities.

Output and type of packing: The output consists of a sequence of pressing steps $P_1, \ldots, P_k$ that contains all ordered pieces, where k can be chosen arbitrarily. A pressing step is denoted by $P = \{(h_1, L_1), \ldots, (h_m, L_m)\}$ and consists of a set of layers, described by a pair (h_i, L_i), $i = 1, \ldots, m$ where $h_i \in \mathbb{N}$ is a height and $L_i \in \mathcal{M}(\mathbb{N})$ a multiset of piece lengths. In each layer, only pieces from one order can be contained (thus, each layer has a uniform height given by the height of the corresponding order).

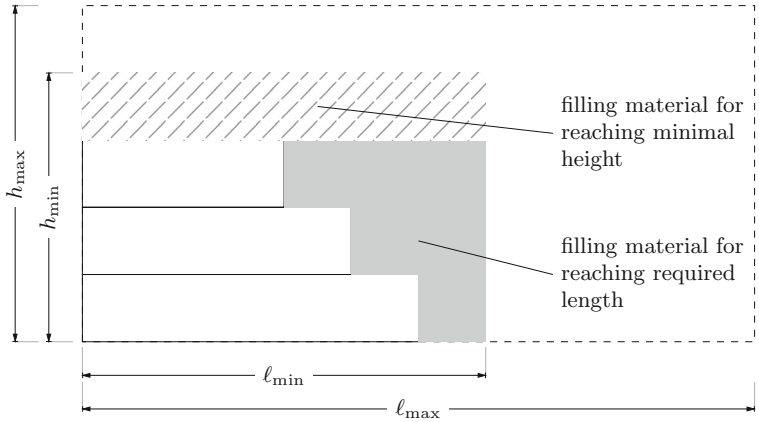

Fig. 2 Illustration of the press parameters and their implications on a pressing step. If the ordered pieces do not reach the required minimal length $\ell_{\min}$ or minimal height $h_{\min}$, filling material needs to be added. The total length of the pressing step cannot exceed the maximal length $\ell_{\max}$, and the height needs to be below the maximal height $h_{\max}$

Sequence constraints: We have the following requirements on the sequences: All orders belonging to a customer have to be placed just-in-sequence. All pieces of an order have to be placed just-in-sequence as well.

Packing constraints: We denote by

$$\ell(P) := \max\{\ell_{\min}, \max_{(h,L)\in P} ||L||\}$$

the total length of the pressing step P, by

$$h_{\text{total}}(P) := \sum_{(h,L)\in P} h$$

the total height of a pressing step and by

$$\Delta h(P) := \max\{h_{\min} - h_{\text{total}}, 0\}$$

the additional height required to reach the minimal height $h_{\min}$.
Each pressing step P should fulfill:

- the maximal height constraint $h(P) \leq h_{\max}$,
- the maximal length constraint $\ell(P) \leq \ell_{\max}$.

Objectives: The following two quantities should be minimized:

Filling material: The total amount of filling material is given by:

$$f(P_1, \ldots, P_k) := \underbrace{\sum_{i=1}^{k} \sum_{(h,L)\in P_i} h\,(\ell(P_i) - ||L||)}_{\text{filling up to largest length}} + \underbrace{\sum_{i=1}^{k} \ell(P_i)\Delta h(P_i)}_{\text{filling up to minimal height}} .$$

We want to minimize the total amount of filling material in order to reduce the waste.

Height changes: The number of changes of height between consecutive pressing steps is given by

$$c(P_1, \ldots, P_k) := \sum_{i=1}^{k-1} \mathbb{1}\,(h_{\text{total}}(P_i) \neq h_{\text{total}}(P_{i+1}))\,.$$

We want to minimize the number of height changes to keep total production time low.

We require the following assumption on the press parameters, which ensures that we can always obtain a feasible packing with a total length between the minimal and maximal length and similarly for the height. It is fulfilled in practice.

Assumption 1 *(Constraints on press parameters)*

- The maximal pressing length is at least twice as large as the minimal pressing length: $\ell_{\max} \geq 2\ell_{\min}$.
- The maximal pressing height is at least twice as large as the minimal pressing height: $h_{\max} \geq 2h_{\min}$.

1.2.1 Discussion of the model

The two objectives in Definition 1 represent the relevant goals when optimizing the production process. The filling material corresponds to the amount of waste and the number of height changes leads to increased production times because of the required setup.

The type of packing modeled in the above definition corresponds to a two-dimensional packing problem with two-stage exact guillotine cuts and variable container sizes. This packing type is due to the characteristics of the press and the corresponding handling: The pieces need to be cut in a regular way with an automatic process. The restriction to this type of cuts makes the modeling as layers simpler than using two-dimensional coordinates. The height changes can be interpreted as sequence-dependent setup times. The sequence constraints follow from logistical reasons: When all parts of an order appear just-in-sequence in the pressing steps, they can directly packaged together. Similarly, all orders belonging to one customer can be immediately packaged together.

1.3 Our results

The described basic approach follows that of Ackermann and Dinges (2017). However, in this paper, we develop new algorithms for each of the steps and provide a theoretical reasoning for their efficiency. By establishing connections to other well-known combinatorial optimization problems and techniques we systematically analyze these algorithms.

We divide our problem into three separate steps:

1. Find a packing into pressing steps for single orders and for combinations of up to three orders. The combinations can be overlapping, i.e., an order can be contained in multiple combinations. The packing of single orders is discussed in Sect. 2. Using a theoretical consideration (Theorem 1), we can reduce the packing problem to a simpler partitioning problem, which we solve with the largest difference method. The packing of combinations is discussed in Sect. 3.
2. Using the packings found in the previous step, compute a set covering of the orders with minimal total filling material, as discussed in Sect. 3.1. For this, dynamic programming can be used.
3. Find a permutation of the packings (subject to just-in-sequence constraints) which minimizes the number of occurring height changes. We formulate this as a scheduling problem with sequence-dependent setup times. An efficient exact algorithm for finding this permutation based on Eulerian extensions and dynamic programming is presented in Sect. 4.

The advantage of this decomposed approach is the possibility of choosing algorithms for each step independently which greatly increases the flexibility. In general, each of the individual steps is NP-hard to solve; thus, we focus on heuristics that are fast enough but yield good results in practice. In final Sect. 5, we summarize our heuristics and present numerical results on real-world inputs. We show that our method is able to compute a plan for multiple months of production in a few minutes. The computed solutions achieve both a small amount of filling material and a small number of height changes.

1.4 Related work

For our problem, Ackermann and Dinges (2017) developed an algorithm based on various heuristics which is successfully used in combination with an enterprise resource planning system in a sawmill. Even though it is only an approximation algorithm, the solution quality exceeds that of human experts. Although the same framework is used there, we significantly improve the algorithms used in each of the steps and supplement them with a theoretical reasoning for their efficiency. The bachelor thesis by Diessel (2018) gives some additional theoretical analysis of the used algorithms and extended numerical results.

An exact algorithm, based on a mixed-integer program with a specifically designed Branch-Price-and-Cut scheme, was developed by Leoff (2016). Particular emphasis was given on the relaxation of the constraint that all items belonging to one order are appearing consecutively, e.g., multiple open orders were allowed. This algorithm

however has a prohibitively large running time, exceeding one hour for almost all realistic inputs.

Integrated packing and sequence-optimization problems arise in other industries, where also material waste and production time have to be minimized. Wuttke and Heese (2018) address such a two-dimensional cutting stock problem in the textile industry. They discuss a mixed-integer program formulation and heuristics. A similar problem encountered in a plastic company is solved with a heuristic by Song (2006). Their algorithm takes into account multiple criteria: due dates, material waste and processing times. Another instance of such problems has been considered by Zhi-Long and Guruprasad (2009). Despite the similarity of the problems on the surface, these algorithms cannot be applied to our problem. The reason is that the employed heuristics are tailored to very specific constraints which depend heavily on the considered application.

There is a large variety of different packing problems considered in the literature, see Dyckhoff (1990) and Wäscher et al. (2007) for overviews. Our problem is related to the two-dimensional bin-packing problem (see the surveys by Lodi et al. (2002) and Lodi et al. (2002)). The difference to our problem is that in the bin-packing problem, the number of bins is minimized instead of the remaining capacity in the bins. Additionally, the dimension of the bins is not fixed in our problem, but is variable within some constraints.

The two-dimensional strip-packing problem is also different from our problem since there is an upper bound on the maximal size of a bin. Hence, efficient approximation algorithms for the strip-packing problem, as for example the PTAS by Kenyon and Réemila (2000), cannot be applied.

We can consider our packing problem as a special case of the two-dimensional variable-sized bin-packing problem analyzed by Pisinger and Sigurd (2005), where there are multiple bin dimensions available, each with their own cost. They present an integer linear-programming model and an algorithm based on branch-and-price. The running time of their algorithms (up to one hour on instances with 100 items) is however to large for our application.

Since we also need to find an optimal sequence of the generated packings, we are faced with an integrated packing and sequence-optimization problem. A survey on the corresponding type of scheduling problems with sequence-dependent setup times is given by Allahverdi (2015). This particular scheduling problem is analyzed in another paper by the authors (Diessel and Ackermann 2019).

2 Packing single orders

In this section, we treat the case of packing a single order, before considering the complete planning which can consist of multiple orders from different customers in a later chapter. Since pieces belonging to the same order can be permuted arbitrarily, we do not have to deal with scheduling problems which will be addressed later in Sect. 4. Also, the height of all pieces is the same.

2.1 Problem description

When considering the special case of packing a single order, the full glulam production problem of Definition 1 reduces to the following Press Packing problem for single orders. The sequence constraints of Definition 1 are always fulfilled when only a single order is considered.

We are able to simplify many of the constraints because all pieces of an order share the same height h. By grouping the pressing steps according to their height, the number of height changes can be reduced to the number of occurring heights of the pressing steps minus one. Due to the shared height of all pieces, the height of a pressing step is proportional to the number of layers.

Definition 2 *(Press Packing problem for single orders)*

Input: a height $h \in \mathbb{N}$ and a multiset L of lengths, press parameters $h_{\min}$, $h_{\max}$, $\ell_{\min}$, and $\ell_{\max}$

Output: A sequence of pressing steps $P_1, \ldots, P_k$, where each pressing step is a multiset of layers. A layer is again a multiset of lengths.

Constraints: For each pressing step P_i, $1 \leq i \leq k$, the maximal height constraint

$$|P_i| \cdot h \leq h_{\max}$$

must hold. For all layers $L \in P_i$, the total length has to be at most the maximal length, i.e., :

$$||L|| \leq \ell_{\max}.$$

Criteria: The following two quantities should be minimized:

Filling material: For $1 \leq i \leq k$, let $\ell_i := \max\{\ell_{\min}, \max_{L \in P_i} ||L||\}$ be the total length of the ith pressing step.

The total amount of filling material is given by:

$$f(P_1, \ldots, P_k) := \underbrace{h \sum_{i=1}^{k} \sum_{L \in P_i} (\ell_i - ||L||)}_{\text{filling up to largest length}} + \underbrace{\sum_{i=1}^{k} \ell_i \cdot \max\{h_{\min} - h \cdot |P_i|, 0\}}_{\text{filling up to minimal height}}. \tag{1}$$

Height changes: The number of height changes depends on the amount of distinct numbers of layers used in pressing steps with the relation

$$c(P_1, \ldots, P_k) := |\{|P_i| : 1 \leq i \leq k\}| - 1.$$

It can be shown that under the assumption $\mathsf{P} \neq \mathsf{NP}$ and the requirement on the press parameters $h_{\min} \leq \frac{3}{4} \frac{\ell_{\min} + \ell_{\max}}{\ell_{\max}} h_{\min} \leq h_{\max}$, there is no polynomial time algorithm which always return an approximation for the Press Packing problem (with these

fixed press parameters) with an additive gap less than $\frac{\ell_{\min}+\ell_{\max}}{4} \cdot h_{\min}$ (Diessel (2018), Theorem 2.6).

2.2 Theoretical derivation of the algorithmic approach

We start with a theoretical derivation of the approach used by our heuristics. The algorithm for finding good pressing steps is based on the following theorem which guarantees a decent solution as long as we find a partition into layers of approximately equal total length.

Theorem 1 (Combination of layers into pressing steps) *Assume that the height of each piece is h and that Assumption 1 is satisfied. Suppose the following two conditions hold:*

(a) *Let the pieces with lengths $L = \{a_1, \ldots, a_n\}$ be partitioned as $L = \dot{\bigcup}_{i=1}^{m} L_i$ in layers $L_1, \ldots, L_m$ such that $\ell_{\min} \leq ||L_i|| \leq \ell_{\max}$ holds for all $1 \leq i \leq m$ and let*

$$d(L_1, \ldots, L_m) := \max_{i \in \{1,\ldots,m\}} ||L_i|| - \min_{i \in \{1,\ldots,m\}} ||L_i||$$

be the maximal difference of the layer sums.

(b) *Additionally, let $m \in \mathbb{N}$ and numbers $a_1, \ldots, a_k \in \mathbb{N}$ be given with $\sum_{i=1}^{k} a_i = m$ and $h_{\min} \leq h \cdot a_i \leq h_{\max}$ holding for all $i \in \{1, \ldots, k\}$ and at most two distinct values occurring in $a_1, \ldots, a_k$.*

Then, there exists a sequence of pressing steps $P_1, \ldots, P_k$ for the order (L, h) such that the total amount of filling material is bounded by:

$$f(P_1, \ldots, P_k) \leq h \cdot \left(\left\lfloor \frac{h_{\max}}{h} \right\rfloor - 1 \right) \cdot d(L_1, \ldots, L_m) \leq h_{\max} \cdot d(L_1, \ldots, L_m)$$

and the number of height changes $c(P_1, \ldots, P_k)$ is at most one.

Proof Sort the layers $L_1, \ldots, L_m$ according to their sum $||L_{(i)}||$ in increasing order, to get a permutation $L_{(1)}, \ldots, L_{(m)}$ with $||L_{(1)}|| \leq \cdots \leq ||L_{(m)}||$. By definition of the maximal difference, we then have $d(L_1, \ldots, L_m) = ||L_{(m)}|| - ||L_{(1)}||$.

By assumption there is a sequence $a_1, \ldots, a_k \in \mathbb{N}$ with $\sum_{i=1}^{k} a_i = m$ and $h_{\min} \leq h \cdot a_i \leq h_{\max}$ for all $1 \leq i \leq k$ with the additional property that in $a_1, \ldots, a_k$ at most 2 distinct values occur. We now insert the layers in the order of their sums into pressing steps

$$\begin{aligned} P_1 &= \{L_{(1)}, \ldots, L_{(a_1)}\}, \\ P_2 &= \{L_{(a_1+1)}, \ldots, L_{(a_1+a_2)}\}, \\ &\ldots \\ P_k &= \{L_{(a_1+\cdots+a_{k-1}+1)}, L_{(a_1+\cdots+a_k)}\}, \end{aligned}$$

as also illustrated in Fig. 3.

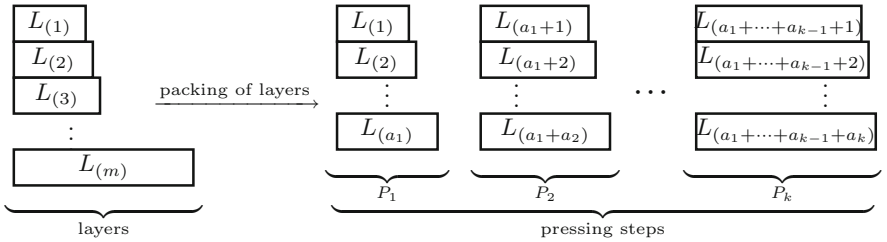

Fig. 3 A set of layers $L_{(1)}, L_{(2)}, \ldots L_{(m)}$, sorted in increasing order by their total lengths, i.e., $||L_{(1)}|| \leq \cdots \leq ||L_{(m)}||$, is packed into pressing steps $P_1, \ldots, P_k$. The number $a_i, i = 1, \ldots, k$, denotes the number of layers in the pressing step P_i

Because $h_{\min} \leq h \cdot a_i \leq h_{\max}$ holds for all $1 \leq i \leq k$, the pressing steps satisfy the constraint on the height and because $\ell_{\min} \leq ||L_i|| \leq \ell_{\max}$ holds for all $1 \leq i \leq n$, also the constraint on the length. As in the number of layers per pressing step $a_1, \ldots, a_k$ at most 2 distinct values occur, the number of height changes is bounded by

$$c(P_1, \ldots, P_k) = |\{|P_i| : 1 \leq i \leq k\}| - 1 = |\{a_i : 1 \leq i \leq k\}| - 1 \leq 2 - 1 = 1$$

For the amount of filling material, we have by definition:

$$f(P_1, \ldots, P_k) = h \cdot \left(\sum_{i=1}^{k} \sum_{A \in P_i} (\ell_i - ||A||) \right),$$

where $\ell_i := \max_{A \in P_i} ||A||$ is the largest total length in the pressing step P_i. By the sorting of the layers $L_{(1)}, \ldots, L_{(m)}$ and the choice of the pressing steps $P_1, \ldots, P_k$, we get $||L_{(1)}|| \leq ||L_{(2)}|| \leq \cdots \leq ||L_{(m)}||$ and $\ell_i = ||L_{(a_1+\cdots+a_i)}||$.

Hence, we have for all $1 \leq i \leq n$:

$$\begin{aligned} \sum_{A \in P_i} (\ell_i - ||A||) &\leq \sum_{j=1}^{a_i} ||L_{(a_1+\cdots+a_i)}|| - ||L_{(a_1+\cdots+a_{i-1}+j)}|| \\ &\leq (a_i - 1) \cdot \left(||L_{(a_1+\cdots+a_i)}|| - ||L_{(a_1+\cdots+a_{i-1}+1)}|| \right). \end{aligned}$$

Plugging this into the whole sum leads to the desired bound for the filling material:

$$\begin{aligned} f(P_1, \ldots, P_k) &= h \cdot \left(\sum_{i=1}^{k} \sum_{A \in P_i} (\ell_i - ||A||) \right) \\ &\leq h \cdot \left(\sum_{i=1}^{k} (a_i - 1) \cdot \left(||L_{(a_1+\cdots+a_i)}|| - ||L_{(a_1+\cdots+a_{i-1}+1)}|| \right) \right) \end{aligned}$$

$$\overset{\text{telescoping sum}}{\leq} h \cdot \max_{i \in \{1,\ldots,k\}} (a_i - 1) \cdot \left(||L_{(m)}|| - ||L_{(1)}|| \right)$$
$$\overset{a_i \cdot h \leq h_{\max}}{\leq} h \cdot \left(\left\lfloor \frac{h_{\max}}{h} \right\rfloor - 1 \right) \cdot d(L_1, \ldots, L_m).$$

The second inequality now follows directly from

$$h \cdot \left(\left\lfloor \frac{h_{\max}}{h} \right\rfloor - 1 \right) \leq h \cdot \left(\frac{h_{\max}}{h} - 1 \right) \leq h_{\max}.$$

□

It can be shown that the bound of the above theorem is tight.

We conclude from Theorem 1 that a partition of the set of piece lengths into layers can be used to obtain a sequence of pressing steps. The filling material decreases proportionally with the maximal difference of the layer sums. Thus, it is sufficient to find partitions with small maximal difference volume. In order to turn the construction in the proof of Theorem 1 into an algorithm, we need to solve the following two subproblems to fulfill the conditions of the theorem:

- In order to fulfill condition (a), we need to find a partition of the pieces into layers such that their lengths do not differ too much. This is done in following Sect. 2.3, making use of the largest difference method (LDM).
- To achieve condition (b), a feasible number of layers to put into each pressing step must be computed. Section 2.4 addresses this.

We can then combine Algorithm 2 described later with the constructive proof of Theorem 1 to following Algorithm 1 for packing a single order. The idea is to try partitions of different sizes and to pack the resulting layers in decreasing order of their total length into pressing steps. In this way, we ensure that the length difference in one pressing step remains as small as possible.

Algorithm 1: Packing of single orders

Input: an order consisting of a multiset of positive piece lengths S and a height
Output: a packing of all items of the order into feasible pressing steps
foreach *possible total number of layers* m **do**
  Run LDM (Algorithm 2) to partition S into m subsets $L_1, \ldots, L_m$
  if *the layer sums* $||L_1||, \ldots, ||L_m||$ *do not exceed the maximal length* $\ell_{\max}$ **then**
    Pack the layers $L_{(1)}, \ldots, L_{(m)}$ in the order $||L_{(1)}|| \leq \cdots \leq ||L_{(m)}||$ in pressing steps using the construction of Theorem 1 with the number of layers in each pressing step computed by Algorithm 3
    Store the corresponding packing
  end
end
return the packing found with minimal filling material

Algorithm 1 has a running time of $O(|S|^3 \log |S|)$ on the input S.

2.3 Packing pieces into layers

We now describe an approach to find a good solution for condition (a) of Theorem 1. This subproblem can be formalized as a number partitioning problem.

Definition 3 *(Number partitioning problem)* Given a multiset S of positive numbers and a number of bins $m \in \mathbb{N}$, find a partition $\dot{\bigcup}_{i=1}^{m} L_i = S$ that minimizes the *maximal difference*

$$d(L_1, \ldots, L_m) := \max_{i \in \{1,\ldots,m\}} ||L_i|| - \min_{i \in \{1,\ldots,m\}} ||L_i||.$$

If the number of bins m in the number partitioning problem is chosen appropriately, the average value $\frac{||S||}{m}$ of the sums in each subset $||L_1||, \ldots, ||L_m||$ is approximately in the middle of $\ell_{\min}$ and $\ell_{\max}$. Therefore, a small value of the maximal difference $d(L_1, \ldots, L_m)$ ensures that all sums are in the interval $[\ell_{\min}, \ell_{\max}]$. Thus, the length constraints can also be handled by focusing on the number partitioning problem.

Since the number partitioning problem can be seen as a special case of the classic partition problem (Karp 1972) by setting $m = 2$ and comparing the minimal value of d with 0, the number partitioning problem is NP-hard. Hence, we focus on polynomial-time approximation algorithms.

Because we will later solve number partitioning problems multiple times, with different numbers of sets m, the algorithms must be very fast. The recent exact algorithm by Schreiber and Korf (2014) as well as other state-of-the-art algorithms by Schreiber and Korf (2013) for this problem are too slow, surpassing one second running time for medium-sized problems with $n = 50$ items. A small decrease in the solution quality is acceptable, since we only use this as a subproblem to our packing problem.

Following Algorithm 2 for the number partitioning problem was first given by Karmarkar and Karp (1983). Their algorithm was randomized to facilitate the analysis of the performance. We present a deterministic version of the algorithm which can be implemented more efficiently, following the presentation of Michiels (2007). The algorithm is called largest difference method (LDM) since in each step the two subsets with the largest difference between largest and smallest element are combined. Here, the largest element of one set is added to the smallest element of the other set, the second-largest element to the second-smallest element and so on. In this way, a partition of the multiset into m subsets with approximately equal sum is achieved.

When implementing P as a priority queue data structure, for example a binary heap, using the d-values as the key, LDM has a running time of $O(|S|(\log |S| + m \log m))$.

2.4 Computing the number of layers for the pressing steps

We now address the issue of fulfilling condition (b) of Theorem 1: finding optimal numbers of layers to put into the pressing steps.

The number of layers in one pressing step is bounded from above by $\frac{h_{\max}}{h}$. The number of layers should preferably be at least $\frac{h_{\min}}{h}$, so that the minimal height is reached and no filling material is required.

Algorithm 2: Largest difference method (LDM)

Input: A multiset of positive numbers and a number $m \in \mathbb{N}$
Output: A partition $\dot{\bigcup}_{i=1}^{m} L_i = S$
$P \leftarrow$ empty list
for $a \in S$ **do**
  Add the multiset $\{\{a\}, \underbrace{\emptyset, \ldots, \emptyset}_{m-1 \text{ times}}\}$ of cardinality m to P
end
while *P contains more than one set* **do**
  Let A, A' be the two multisets in P with largest maximal difference
  Remove A and A' from P and instead add their combination C defined as follows:
    Let $\{B_1, \ldots, B_m\} = A$ and $\{B'_1, \ldots, B'_m\} = A'$ such that the sums $||B_1|| \leq \cdots \leq ||B_m||$ and $||B'_1|| \leq \cdots \leq ||B'_m||$ are increasing.
    Then their combination is: $C = \{B_1 \cup B'_m, B_2 \cup B'_{m-1}, \ldots, B_m \cup B'_1\}$
end
return the last remaining multiset in P

To reach a small number of height changes, only a small number of different number of layers should be used. Ideally, only one number of layers would be used, but this can be impossible for example when the number of pieces is a prime number. However, we are able in most situations to find solutions with only two different numbers of layers.

We can give the following number theoretic version of the problem:

Definition 4 *(Layer Partitioning problem)* For a given number m of layers, find a partition $m = s_1 + \cdots + s_k$ into numbers $s_1, \ldots, s_k \in \mathbb{N}$, such that the set $\{s_1, \ldots, s_k\}$ consists of at most two distinct values, with the additional requirement that $\frac{h_{\min}}{h} \leq s_i \leq \frac{h_{\max}}{h}$ holds for $i = 1, \ldots, k$.

A formula for the number of such partitions (without the requirement that the s_i are in a particular range) has been given by Tani and Bouroubi (2011).

We will now describe an algorithm to solve the Layer Partitioning problem. The algorithm proceeds in such a way that the ideal solution with only one number of layers is preferred over two distinct number of layers which is in turn preferred over a solution where filling material is required. In all cases however, the algorithm ensures that only two distinct numbers of layers are used and that filling material is only needed for one pressing step.

The running time of Algorithm 3 is bounded by $O\left(m \cdot \frac{h_{\max}^2}{h \cdot h_{\min}}\right)$.

It can be shown under the assumption $h_{\max} \geq 2h_{\min}$ that in the case where m is even, there always exists a solution with $\frac{h_{\min}}{h} \leq h_i \leq \frac{h_{\max}}{h}$ for all $i = 1, \ldots, k$, such that $h_1, \ldots, h_k$ contains only two distinct values, see Diessel (2018, Lemma 2.7). This also implies that only one additional layer of filling material is required, since we can make an odd number of layers even by adding one layer. The algorithm will always find such a solution.

In practice, for odd m we can also almost always find a solution with no filling material, provided that m is not too small. Hence, the situation where filling material

Algorithm 3: Finding optimal numbers of layers

Input: a number m of layers each having height h, parameters $h_{\min}, h_{\max}$
Output: numbers of layers $h_1, \ldots, h_k$, such that $h_1 + \cdots + h_k = m$, $h_i \leq \frac{h_{\max}}{h}$ and preferably $h_i \geq \frac{h_{\min}}{h}$ for all $i = 1, \ldots, k$ and the number of distinct values in $h_1, \ldots, h_k$ is small

for $h \in \{\lceil \frac{h_{\min}}{h} \rceil, \ldots, \lfloor \frac{h_{\max}}{h} \rfloor\}$ **do**
  if h *divides* m **then**
    return $h_1 = \cdots = h_{m/h} = h$
  end
end
for $a \in \{\lceil \frac{h_{\min}}{h} \rceil, \ldots, \lfloor \frac{h_{\max}}{h} \rfloor\}$ **do**
  for $b \in \{\lceil \frac{h_{\min}}{h} \rceil, \ldots, a\}$ **do**
    for $x \in \{1, \ldots, \lfloor \frac{m}{a} \rfloor\}$ **do**
      if b *divides* $m - x \cdot a$ **then**
        return $h_1 = \cdots = h_x = a, h_{x+1} = \cdots = h_{(m-x\cdot a)/b} = b$
      else
        Store the fill number $\max\left\{\lceil \frac{h_{\min}}{h} \rceil - ((m - x \cdot a) \mod b), 0\right\}$ along with a, b, x
      end
    end
  end
end
Choose a, b and x with minimal fill number
return $h_1 = \cdots = h_x = a, h_{x+1} = \cdots = h_{\lfloor (m-x\cdot a)/b \rfloor} = b, h_{\lfloor (m-x\cdot a)/b \rfloor + 1} = (m - x \cdot a) \mod b$

is required can be avoided in most cases. Otherwise, it is optimal to choose the layers with the smallest lengths to be in the pressing step which is to be filled.

3 Multiple orders

The goal of this section is to extend the algorithms for the packing of single orders to a combination of orders.

In some cases, an order cannot be packed alone without significant filling material. Due to the packing constraints in Definition 1 of the glulam production problem, this occurs when the order does not have a sufficient total size to reach the minimal length or minimal height. By packing combinations of two or more orders together, we can also achieve less filling material in such situations.

As it is not directly clear which orders should be combined, we generate many combinations and use a Set Cover approach to find a set of combinations such that each order is produced exactly once and the total filling material is minimized.

The Set Cover approach for computing an optimal subset of the found combinations is described in Sect. 3.1. The following subsections show how pressing steps for combinations of two (Sect. 3.2), respectively, three (Sect. 3.3) orders can be formed.

3.1 Set covering

We model the problem of finding a subset of combinations such that each order is produced exactly once and the total filling material is minimized as a Weighted Exact Set Cover problem. The requirement to find an exact cover ensures that orders are not produced multiple times.

Definition 5 *(Weighted Exact Set Cover problem)*

Input: a family $\mathcal{A}$ of nonempty sets over a finite universe U with a weight function $w : \mathcal{A} \to \mathbb{R}^+$

Output: a subfamily $\mathcal{S} \subseteq \mathcal{A}$ of disjoint sets which cover the universe in the sense that $\dot{\bigcup}_{S \in \mathcal{S}} S = U$, such that the weight of the cover $\sum_{S \in \mathcal{S}} w(S)$ is minimized.

In our application the sets are combinations of orders, with the weight being the corresponding filling material of the packing as computed by Algorithms 4 and 5 . Additionally, for each order we compute a packing for it alone using Algorithm 1. We then also include the corresponding singleton sets in the Set Cover instance. This ensures that there always exists at least one exact cover, since we can choose just these singleton sets as the cover. However, this might be far from optimal.

By using a standard dynamic programming approach, the Weighted Exact Set Cover problem can be solved in time $O(2^{|U|} \cdot |\mathcal{A}| \cdot |U|)$.

But before considering a Set Cover, we need to generate the sets with their weights by computing for all possible combinations good packings to evaluate their corresponding filling material. We restrict ourselves to combinations of at most three orders. A higher number of orders in a combination would increase the number of possible combinations by a large factor, rendering their computation infeasible in practice. Also, since in practice the height h is usually at least a fifth of the maximal height $h_{\max}$, it only rarely occurs that more than three orders can appear in one pressing step.

We now discuss how good pressing steps for a combination of two or three orders can be obtained.

3.2 Combinations of two orders

We try to find a packing of the form depicted in Fig. 4 where one of the pressing steps contains pieces from two different orders.

Note that all allowed packings of a combination of two orders have such a form, since all pieces from one order have to appear consecutively. By using such combinations of two orders, the filling material can be reduced in comparison with packing each order individually: It is then easier to achieve the goal of finding a packing with each total height being within the bounds of minimal and maximal height.

In following Algorithm 4, such a combination of two orders is computed. We do not use Algorithm 1 for packing single orders directly. Instead, we use the idea of Theorem 1 and LDM to generate completely new packings.

The algorithm first computes multiple partitions of the pieces of the first order and the second order into layers. For every found partition, the filling material is computed for packing all but a specified number of remaining layers into pressing steps. Also,

Algorithm 4: Packing combinations of two orders

Input: orders (S_1, h_1) and (S_2, h_2) with two multisets of lengths S_1, S_2 and heights h_1, h_2

Output: a sequence of pressing steps $P_1^{(1)}, \ldots, P_k^{(1)}$ for (S_1, h_1) and a sequence of pressing steps $P_1^{(2)}, \ldots, P_{k'}^{(2)}$ for (S_2, h_2) such that $P_k^{(1)}$ and $P_1^{(2)}$ can be combined, i. e. $h_{\min} \leq h_1 \cdot |P_k^{(1)}| + h_2 \cdot |P_1^{(2)}| \leq h_{\max}$

foreach *possible number* m_1 *of layers for* (S_1, h_1) **do**
  Let $L_1(m_1)$ be the packing of (S_1, h_1) into m_1 layers computed by LDM (Alg. 2)
  for $s \in \{1, \ldots, \lfloor \frac{h_{\max}}{h} \rfloor\}$ **do**
    Let $f_1^{\text{small}}(m_1, s)$ be the total filling material for packing all but the smallest s layers of $L_1(m_1)$ into pressing steps using e.g. Theorem 1
    Let $f_1^{\text{large}}(m_1, s)$ be the total filling material for packing all but the largest s layers of $L_1(m_1)$
  end
end
Compute L_2, f_2^{small} and f_2^{large} for (S_2, h_2) analogously
foreach *possible number* m_1 *of layers for* (S_1, h_1) *and* m_2 *of layers for* (S_2, h_2) **do**
  foreach $s_1 \in \left\{1, \ldots, \left\lfloor \frac{h_{\max}}{h_1} \right\rfloor\right\}$ *and* $s_2 \in \left\{1, \ldots, \left\lfloor \frac{h_{\max}}{h_2} \right\rfloor\right\}$ **do**
    Compute the sum of $f_1^{\text{small}}(m_1, s_1)$, $f_2^{\text{small}}(m_2, s_2)$ and the needed filling material for packing the smallest s_1 layers of $L_1(m_1)$ and the smallest s_2 layers of $L_2(m_2)$ together into a pressing step (this is the way of combination *small-small*)
    Do this analogously for the three other possible ways *small-large*, *large-small* and *large-large*
    if *one of the feasible computed packings has smaller filling material than the previous best* **then**
      Store corresponding values m_1, m_2, s_1, s_2 and the way of combination of this packing
    end
  end
end
return the reconstruction of the packing with the stored m_1, m_2, s_1, s_2 and the way of combination

Algorithm 5: Packing combinations of three orders

Input: orders (S_1, h_1) and (S_2, h_2) with three multisets of lengths S_1, S_2, S_3 and heights h_1, h_2, h_3, where (S_3, h_3) is small enough to be packed into one pressing step

Output: a sequence of pressing steps $P_1^{(1)}, \ldots, P_k^{(1)}$ for (S_1, h_1) and a sequence of pressing steps $P_1^{(2)}, \ldots, P_{k'}^{(2)}$ for (S_2, h_2) and one pressing step $P^{(3)}$ for (S_3, h_3) such that $P_k^{(1)}$, $P_1^{(2)}$ and $P^{(3)}$ can be combined, i. e. $h_{\min} \leq h_1 \cdot |P_k^{(1)}| + h_2 \cdot |P_1^{(2)}| + h_3 \cdot |P^{(3)}| \leq h_{\max}$

Compute packings for the combination of (S_1, h_1) and (S_2, h_2) as in Algorithm 4, however without a restriction on the minimal height of the combined pressing step
Compute packings of the order (S_3, h_3) with Algorithm 1
foreach *combined packing of* (S_1, h_1) *and* (S_2, h_2) **do**
  Insert a packing with an appropriate number of layers for (S_3, h_3) into the combined pressing step of (S_1, h_1) and (S_2, h_2) and compute the resulting total filling material
  Store the packing and the corresponding filling material, if the packing is feasible
end
return the found packing with the smallest filling material

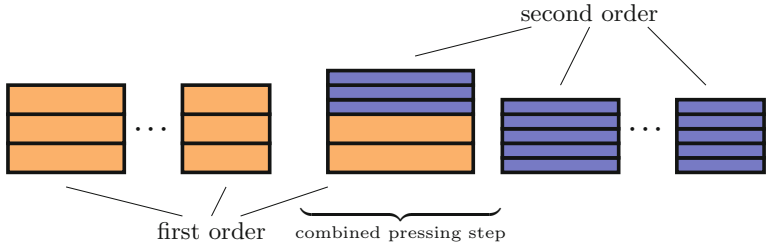

Fig. 4 A sequence of pressing steps for a combination of two orders

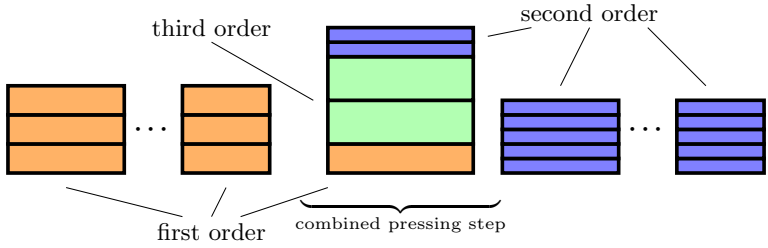

Fig. 5 A sequence of pressing steps for a combination of three orders

multiple combinations of the remaining layers of the first and second order are tried to be packed into one pressing step. In the end, the packing with the smallest total of filling material is returned.

Algorithm 4 has a running time of

$$O\left(|S_1|^3 \log(|S_1|) + |S_1|^2 \cdot \log(|S_1|) \cdot \frac{h_{\max}}{h_1} + |S_2|^3 \log(|S_2|) + |S_2|^2 \cdot \log(|S_2|) \cdot \frac{h_{\max}}{h_2} + |S_1| \cdot |S_2| \cdot \frac{h_{\max}}{h_1} \cdot \frac{h_{\max}}{h_2} \cdot \left(\frac{h_{\max}}{h_1} + \frac{h_{\max}}{h_2}\right)\right).$$

3.3 Combinations of three orders

If one order is particularly small, we can try to pack it in between two different orders, by forming a pressing step which consists of all the pieces from the small orders and parts of the two other orders, as illustrated in Fig. 5. Since packing such a small order alone would not reach the minimal height, the filling material can be significantly reduced by using these combinations.

In Fig. 5, the *small* order is the third order. Note that the presence of three orders in one pressing step is only possible if the total material of one of the orders is small enough to be packed in one single pressing step.

Another possible way of combining three orders by using two combined pressing steps is shown in Fig. 6. However, we do not compute such combinations, as doing so would provoke a large running time.

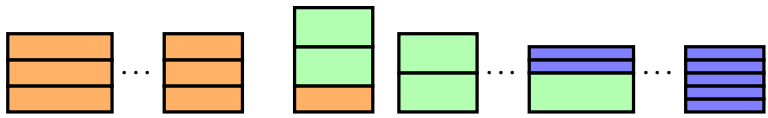

Fig. 6 An alternative way of combining three orders

Algorithm 5 for packing combinations of three orders combines a small third order with two large orders. The idea is to first find packings of the two large orders, using Algorithm 4. The residual layers of these two orders are then combined with the items of the third order into one pressing step.

Algorithm 5 has a running time of

$$O\left(|S_1|^3 \log(|S_1|) + |S_1|^2 \cdot \log(|S_1|) \cdot \frac{h_{\max}}{h_1} + |S_2|^3 \log(|S_2|) + |S_2|^2 \cdot \log(|S_2|) \cdot \frac{h_{\max}}{h_2} + |S_1| \cdot |S_2| \cdot \frac{h_{\max}}{h_1} \cdot \frac{h_{\max}}{h_2} \cdot \left(\frac{h_{\max}}{h_1} + \frac{h_{\max}}{h_2}\right) + |S_3|^3 \log(|S_3|) + |S_1| \cdot |S_2| \cdot |S_3|\right).$$

4 Finding an optimal sequence

We now discuss how we can find an optimal sequence of the orders, so that the number of height changes between pressing steps in Definition 1 is minimized. At this stage we assume that the packing for the orders and combinations are already fixed. Thus, there is a fixed start and end height for each order, namely the heights of the first and last pressing step in the packing of that order. By reversing, start and end height can be exchanged.

Formally, minimizing height changes can be seen as a scheduling problem where each order corresponds to a job each with a start and end height. Then there occurs a setup time of 1 whenever the end height of a job does not match the start height of the succeeding job. In another paper by the authors (Diessel and Ackermann 2019), this problem of minimizing the total setup time is formalized as the Domino Sequencing Problem. At first this problem seems intractable as it can be seen as a variant of the Traveling Salesman Problem. However, it has a special structure. Therefore, by a reformulation to an Eulerian Extension problem, the problem can be solved in linear time. The corresponding multigraph has an edge for each order, connecting the nodes that represent the start and end height of this order. Multiple variations of the problem, for example including weights, can also be solved in polynomial time.

In practice, there is the additional sequence constraint in Definition 1 that the orders of each customer are processed contiguously due to logistical reasons. This can be formalized as a Domino Sequencing Problem with families, where each family consists of the orders by one customer. Diessel and Ackermann (2019) describe an algorithm for this problem that has a running time which is polynomial in the number of orders and exponential only in the number of families.

5 Results

5.1 The integrated algorithm

We now describe integrated Algorithm 6 for glulam production problem (Definition 1) that combines all the previously described algorithms for the hierarchical steps as described in the introduction: pack orders and small combinations of orders, find an optimal set cover with these combinations and then put the pressing step into a sequence.

Because of the requirement that all orders by a customer must be produced just-in-sequence, we in general do not combine orders coming from different customers. However, orders by customers who placed just a single order can be combined without issues. Hence, we subsume all orders coming from customers with just a single order under a new "virtual" customer. Then, each customer is treated separately.

Algorithm 6: Integrated algorithm

Input: a set of customers with orders, the machine parameters $\ell_{\min}, \ell_{\max}, h_{\min}, h_{\max}$
Output: a sequence of pressing steps for the production of all pieces of the orders by the customers such that the total filling material and the number of height changes is small
Consider all orders by customers who have placed only a single order as placed by a single new "virtual" customer
foreach *customer* **do**
  foreach *order by the customer* **do**
    Run Algorithm 1 to find a packing for this order
  end
  foreach *combination of two orders by the customer such that each order can be packed in only a few pressing steps* **do**
    Run Algorithm 4 to find a packing for this combination of two orders
  end
  foreach *combination of three orders by the customer, such that the first two orders can be packed in only a few pressing steps and third order in only one pressing step* **do**
    Run Algorithm 5 to find a packing for this combination of three orders
  end
  Run an algorithm for the Exact Set Cover problem on an instance, where the sets are the above computed combinations with their corresponding filling material as weights, to get a set of pressing steps for all orders by the customer
end
Find a sequence of the pressing steps above which minimizes the number of height changes, under the requirement that all orders of a customer are processed in sequence
return *the computed sequence of pressing steps*

The integrated algorithm creates packings for single orders and for combinations of two or three orders. Then, an optimal subset of these combinations is found that covers every order exactly once. This is achieved by solving an instance of the Weighted Exact Set Cover problem. This step is done separately for the orders of every customer. Afterward, the packings are put into a sequence that minimizes the number of height changes.

In integrated Algorithm 6, multiple thresholds have to be chosen that specify which algorithm will run in a particular situation, e.g., up to which size of the order we should

compute packings for combinations with other orders. It is beneficial to limit the size of the orders for which combinations are computed, since the cubic dependence of the running time of Algorithms 4 and 5 on the size of the order renders the combination algorithm slow on large orders. Because large orders can usually be packed efficiently alone, this restriction does not significantly decrease the solution quality.

We have chosen these thresholds using a large number of randomized tests, to evaluate the corresponding performance for each possible value of the threshold. The choice involves a trade-off between running time and the two criteria filling material and height changes. Since a really small running time is not needed for the application, the thresholds are chosen in a way that the solution quality is good without having a prohibitively large running time.

5.2 Numerical results

We implemented integrated Algorithm 6 and all other algorithms presented in this paper in the programming language C#. As test data, we use a generated dataset with realistic characteristics: orders by 550 customers with a total production volume of 346 m^3 which represent a few months of production in a medium-sized factory. The total number of requested pieces is 23,929.

During one planning, the orders of multiple customers (whose orders must all have the same width) are planned in parallel. Since the corresponding number of customers can be chosen by the user of the software according to the current business requirements, we tested the possibilities of maximally 1, 5, 10, 15 or 20 customers per planning.

Using these different ways of partitioning, the algorithm produces a sequence of pressing steps for all orders in the data set. We recorded the resulting total filling material, the total number of height changes as well as the total running time for planning everything.

The test was done on a desktop computer with the following hardware: an Intel Core i7-4790 processor with a clock speed of 3.60 GHz and 16 GB of RAM. The used operating system was Windows 7 Enterprise (64 bit).

Table 1 shows that the filling material diminishes for larger number of customers per planning, since more possibilities for combinations are available. Compared with the total production volume of 3468 m^3, the amount of filling material represents about 1% of the volume in most cases. The filling material due to the minimal height constraint is always small since the algorithms try to ensure that the minimal height is achieved. Only in rare cases where no other possibility exists this filling material is needed.

The results also show that the incremental decrease in the filling material becomes smaller when the number of customers per planning is already sufficiently large (i.e., for at least 10 customers per planning). This is due to the fact that already enough good combinations for most of the orders which are difficult to plan have been found. The addition of more possibilities for combinations does not lead anymore to a large decrease of the filling material.

Table 1 Numerical results of our algorithm

Number of customers per planning	1	5	10	15	20
Total filling material in m^3	63.02	36.81	34.22	32.37	31.69
Percentage of total filling material in production volume	1.82%	1.06%	0.99%	0.93%	0.91%
Filling material due to filling length in m^3	62.49	36.28	33.69	31.84	31.18
Filling material due to minimal height constraint in m^3	0.53	0.53	0.53	0.53	0.53
Number of height changes	288	518	461	439	418
Total running time in seconds	31.3	37.8	40.7	57.9	238.2
Number of pressing steps	5683	5621	5616	5600	5599
Number of layers	15,508	15,398	15,392	15,301	15,316

Note that height changes between two planning stages were not counted. Therefore the number of height changes for only one customer per planning is much smaller

Table 2 Dependence of the number of times pressing steps consist of combinations of more than one order on the number of customers who are considered simultaneously in one planning

Number of customers per planning	Number of pressing steps combined of 2 orders	Number of pressing steps combined of 3 orders
1	7	0
5	54	8
10	72	5
15	74	7
20	72	7

With higher numbers of considered customers, more combinations are possible

Table 3 Distribution of the number of pieces in the generated layers for the exemplary case of 10 customers per planning

Number of pieces	1	2	3	4	5	6	7
Corresponding frequency of layers	9501	3549	2020	237	40	11	8

The table shows that a large majority of the layers consists of three pieces or less

Table 4 Distribution of the number of layers in the generated pressing steps for the case of 10 customers per planning

Number of layers	1	2	3	4	5	6	7
Corresponding frequency of pressing steps	326	2796	1131	962	260	54	87

The number of height changes also decreases with the number of customers per planning, since there are more possibilities for sequences of the customers. It can also be noticed that the total counts of pressings steps and layers decreases slightly when the number of customers per planning increases. This is due to the increasing number of pressing steps which consist of combinations of orders, leading to larger pressing steps and larger layers (see also Table 2).

Table 2 shows that an almost maximal amount of combined pressing steps is already reached when 10 customers are considered per planning. These combined pressing steps only account for at most 1.5% of all pressing steps, but lead to large decreases in the amount of filling material since orders which are difficult to pack alone are combined in these pressing steps.

The lengths and heights of the pieces in our dataset are typically large relative to the size of the press, limiting the number of pieces that can be combined. Most layers generated by our algorithm only contain a few pieces, as Table 3 shows. Similarly, most pressing steps consist of between 2 and 4 layers, see Table 4.

To summarize, our numerical results show that our approach leads to small running times for large-scale instances of the industrial problem. Also, the results are stable even if the size of the input, i.e., the number of customers for one planning step, changes. Naturally, the algorithm generates better results when more customers are taken into account for one planning, but already a rather small number (about 10 cus-

tomers) is sufficient to generate results with a good quality. This makes the algorithm flexible for practical applications, since the number of customers considered for one planning might depend on current organizational constraints in the factory.

6 Conclusion and outlook

We have presented a packing and sequencing problem arising in the production of glued laminated timber, with the goal of minimizing the required filling material and number of height changes. It can be shown that already subproblems of this integrated problem are computationally infeasible to solve exactly. Hence, heuristics have to be used.

For solving this complex problem, we used the following algorithmic approach which is applicable to a wide range of integrated packing and scheduling problems: We first compute individual solutions for each part of the problem instance. For promising candidates, we build solutions for combinations of two or three parts. We then compute an optimal choice of a subset of the generated combinations by solving a Set Cover instance. Finally, we find an optimal sequence of the computed solutions for the selected combinations by using a scheduling algorithm. The advantage of this framework is that it enables to combine techniques from different domains of discrete optimization: packing, covering and scheduling.

We applied this framework to the production of glued laminated timber. For each of the steps of the framework we developed specific algorithms, combining a wide range of different techniques. We showed that a good packing can be obtained based on a homogeneous partition of the pieces into layers and suitable numbers of layers per pressing steps (Theorem 1). Finding a partition into layers of approximately equal length can be formulated as a number partitioning problem which we solved with the largest difference method. Appropriate numbers of layers are obtained with a number theoretic heuristic. Combining these two steps yielded Algorithm 1 for packing single orders.

We continued with algorithms for computing packings for combinations of two or three orders. We discussed how an optimal subset of the combinations can be selected by solving a Weighted Exact Set Cover problem. Using a scheduling algorithm, the packings can be arranged in a sequence that minimizes the number of height changes. The algorithms discussed before were then combined into an algorithm following the proposed framework for solving the integrated problem. The numerical results showed that our implementation of the framework has a good performance on realistic inputs: It needs only a few minutes to compute production plans for months of production while achieving both a small amount of filling material and height changes. This application example shows that the framework can be highly efficient.

6.1 Outlook

Our algorithmic framework based on a hierarchical decomposition is applicable to a large variety of integrated packing and scheduling problems which are infeasible to

solve optimally. Hence it is interesting to evaluate the performance of this framework when applied to other packing and scheduling problems with different constraints. The generic nature of the framework allows to easily deal with changes to the constraints, since only the relevant algorithms in the framework have to be adapted. A change to the packing constraints for example only requires changes to the packing algorithm for parts and combinations. Additionally, the threshold parameters of the integrated algorithm can be adapted to make use of the relevant characteristics of the input.

The flexibility of this approach also allows to evaluate the impact of using different heuristics for each step in the framework. We can use this to find an optimal trade-off between running time and solution quality of the final result. The optimal allocation of running time to the different steps of the framework might depend on the concrete problem. Therefore it would be beneficial to develop methods that automatically find an optimal allocation of the running time.

Acknowledgements Open Access funding provided by Projekt DEAL. The authors would like to thank Andreas Dinges for help with preparing the test data and Cristina Collicott for suggestions improving the presentation. They are also grateful to comments by two anonymous reviewers that lead to an improvement of the exposition.

Funding Information Fraunhofer ITWM, Institute for Industrial Mathematics.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Ackermann H, Dinges A (2017) Computing efficient pressing operations for glued laminated timber production. In: Proceedings of the 13th workshop on models and algorithms for planning and scheduling problems. MAPSP '17. Seeon Abbey, Germany, pp 122–124

Allahverdi A (2015) The third comprehensive survey on scheduling problems with setup times/costs. Eur J Oper Res 246:345–378. https://doi.org/10.1016/j.ejor.2015.04.004

Diessel E (2018) Design and analysis of algorithms for packing and sequencing in the production of glued laminated timber. B.Sc. thesis, Technische Universität Kaiserslautern

Diessel E, Ackermann H (2019) Domino sequencing: scheduling with state-based sequence-dependent setup times. Oper Res Lett 47:274–280. https://doi.org/10.1016/j.orl.2019.04.004

Dyckhoff H (1990) A typology of cutting and packing problems. Eur J Oper Res 44:145–159. https://doi.org/10.1016/0377-2217(90)90350-K

Karmarkar N, Karp RM (1983) The differencing method of set partitioning. Tech. rep., EECS Department, University of California, Berkeley. http://www2.eecs.berkeley.edu/Pubs/TechRpts/1983/6353.html

Karp RM (1972) Reducibility among combinatorial problems. In: Miller RE, Thatcher JW (eds) Complexity of computer computations. Springer, Boston, pp 85–103. https://doi.org/10.1007/978-1-4684-2001-2_9

Kenyon C, Réemila E (2000) A near-optimal solution to a two-dimensional cutting stock problem. Math Oper Res 25:645–656. https://doi.org/10.1287/moor.25.4.645.12118

Leoff J (2016) Hierarchical scheduling and cutting stock with bounded open orders. PhD thesis, Technische Universität Kaiserslautern

Lodi A, Martello S, Monaci M (2002) Two-dimensional packing problems: a survey. Eur J Oper Res 141:241–252. https://doi.org/10.1016/S0377-2217(02)00123-6

Lodi A, Martello S, Vigo D (2002) Recent advances on two-dimensional bin packing problems. Discrete Appl Math 123:379–396. https://doi.org/10.1016/S0166-218X(01)00347-X

Michiels W et al (2007) Performance ratios of the Karmarkar–Karp differencing method. J Comb Optim 13:19–32. https://doi.org/10.1007/s10878-006-9010-z

Pisinger D, Sigurd M (2005) The two-dimensional bin packing problem with variable bin sizes and costs. Discrete Optim 2:154–167. https://doi.org/10.1016/j.disopt.2005.01.002

Schreiber EL, Korf RE (2013) Improved bin completion for optimal bin packing and number partitioning. In: Proceedings of the twenty-third international joint conference on artificial intelligence. IJCAI '13, Beijing, China, AAAI Press, pp 651–658

Schreiber EL, Korf RE (2014) Cached iterative weakening for optimal multi-way number partitioning. In; Proceedings of the twenty-eighth AAAI conference on artificial intelligence. AAAI'14. Quéebec City, Canada, AAAI Press, pp 2738–2744

Serrano E (2003) Mechanical performance and modelling of glulam. In: Thelandersson S, Larsen H (eds) Timber engineering. Wiley, Hoboken

Song X et al (2006) An iterative sequential heuristic procedure to a real-life 1.5-dimensional cutting stock problem. Eur J Oper Res 175:1870–1889. https://doi.org/10.1016/j.ejor.2004.10.034

Tani NB, Bouroubi S (2011) Enumeration of the partitions of an integer into parts of a specified number of different sizes and especially two sizes. J Integer Seq 14:1–12

Wäscher G, Haußner H, Schumann H (2007) An improved typology of cutting and packing problems. Eur J Oper Res 183:1109–1130. https://doi.org/10.1016/j.ejor.2005.12.047

Wuttke DA, Heese HS (2018) Two-dimensional cutting stock problem with sequence dependent setup times. Eur J Oper Res 265:303–315. https://doi.org/10.1016/j.ejor.2017.07.036

Zhi-Long C, Guruprasad P (2009) Integrated order scheduling and packing. Prod Oper Manag 18:672–692. https://doi.org/10.1111/j.1937-5956.2009.01029.x