


Hardness and Tight Approximations of Demand Strip Packing

Klaus Jansen ✉ 

Kiel University, Kiel, Germany

Malin Rau ✉ 

University of Hamburg, Hamburg, Germany

Malte Tutas ✉ 

Kiel University, Kiel, Germany

Abstract

We settle the pseudo-polynomial complexity of the DEMAND STRIP PACKING (DSP) problem: Given a strip of fixed width and a set of items with widths and heights, the items must be placed inside the strip with the objective of minimizing the peak height. This problem has gained significant scientific interest due to its relevance in smart grids [Deppert et al. APPROX’21, Gálvez et al. APPROX’21]. Smart Grids are a modern form of electrical grid that provide opportunities for optimization. They are forecast to impact the future of energy provision significantly. Algorithms running in pseudo-polynomial time lend themselves to these applications as considered time intervals, such as days, are small. Moreover, such algorithms can provide superior approximation guarantees over those running in polynomial time. Consequently, they evoke scientific interest in related problems [Jansen and Rau ESA’19].

We prove that DEMAND STRIP PACKING is strongly NP-hard for approximation ratios below $5/4$. Through this proof, we provide novel insights into the relation of packing and scheduling problems. Using these insights, we show a series of frameworks that solve both DEMAND STRIP PACKING and PARALLEL TASK SCHEDULING optimally when increasing the strip’s width or number of machines. Such alterations to problems are known as resource augmentation. The framework of resource augmentation has been studied in closely related problems such as 2D geometric Bin Packing. For both problems, the use of resource augmentation is a new idea. Applications are found when penalty costs are prohibitively large. Finally, we provide a pseudo-polynomial time approximation algorithm for DSP with an approximation ratio of $(5/4 + \varepsilon)$, which is nearly optimal assuming $P \neq NP$. The construction of this algorithm provides several insights into the structure of DSP solutions and uses novel techniques to restructure optimal solutions.

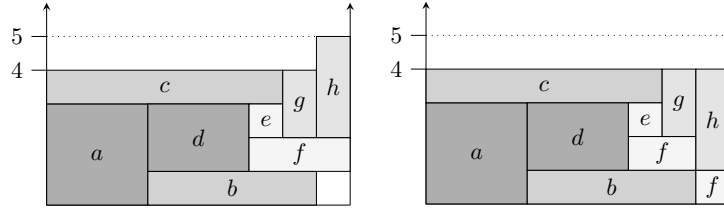
2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases strip packing, peak demand minimization, approximation, hardness, resource augmentation

Funding *Klaus Jansen*: Research supported by German Research Foundation (DFG) project JA 612/25-1.

Malin Rau: DFG Research Group ADYN (FOR 2975) under grant DFG 411362735

Malte Tutas: Research supported by German Research Foundation (DFG) project JA 612/25-1.



■ **Figure 1** An illustration of the difference slicing can make to the packing height. On the left, you can see the optimal solution for the classical strip packing problem, whereas the right displays the optimal packing in the demand variant. As you can see, the height of the packing decreased through slicing. The sliced item contains its name to easier identify it. We can see the difference in optimal heights of $\frac{5}{4}$ here.

1 Introduction

Increased adoption of technologies such as electric cars as well as global economic growth, lead to a growing demand for electric power. To better handle this demand, new technologies are developed. An example of such a technology is the so-called Smart Grid [19, 25, 28]. Traditional power systems have to incorporate infrastructure to support the peak load on the entirety of the system [30]. However, this peak demand is seldom reached, meaning that, at non-peak times, there is some capacity for additional power demands in the power grid. Smart Grids use automated digital communication between power consumers and electricity providers to utilize the available capacity more efficiently by, for example, shifting a user's power-demanding tasks onto off-peak hours. This directly results in a decreased peak load on the entire system. The benefits of this are plentiful. For the consumer, modern devices can automatically shift their power usage to off-peak hours to avoid surcharges through congestion pricing [4]. On the other hand, for electricity suppliers, the utilization of Smart Grids and the accompanying reduction in peak load reduces the required infrastructure to supply the ever-increasing electricity demand across the globe. It has been estimated that implementation of these Smart Grids across the United States alone leads to savings between 46 and 117 billion US-Dollars over 20 years [18].

To effectively generate these benefits, efficient algorithms to balance the load on the electrical system are required. We can model the power consumption of individual applications through a rectangle. The width of this rectangle then represents the duration for which the application consumes power, and its height represents the amount of power it consumes. Modeling individual demands in this way lets us inspect larger time intervals over which tasks may recur. We can model these time intervals as a single strip with a set width, where the width represents the length of the interval. The time intervals in which tasks repeat are usually rather short, i.e. days. Furthermore, when thinking of appliances such as washing machines, their operating time can be quantified in minutes. Finding a packing of all rectangles into this strip with a minimum height then represents a minimization of peak load on the electrical grid. However, the load on the system is the sum of all demands at a certain point in time, which we can not model through these simple rectangles yet. To compensate we introduce the concept of slicing, which allows us to place vertical cuts inside any generated item and place these cut parts at any height as long as they are placed contiguously in their width. This modeling motivates the problem studied in this paper.

Demand Strip Packing

We are given a set \mathcal{I} of n items. Each item $i \in \mathcal{I}$ has a width $w(i)$ and a height $h(i)$. Further, we are given a strip S of width W . The goal is to pack all items into S such that no items overlap each other. Such a packing Λ is defined by a placement function $\lambda : \mathcal{I} \rightarrow \mathbb{N}$ that assigns a starting point $\lambda(i)$ to each item. A starting point $\lambda(i)$ is only feasible if $0 \leq \lambda(i) \leq \lambda(i) + w(i) \leq W$ holds, i.e., the items do not overlap the edges of the strip. All items must be placed contiguously, i.e., no part of the item is allowed to be placed after $\lambda(i) + w(i)$. The objective is to minimize the peak height $H := \max_{k \in \{0, \dots, W\}} (\sum_{i \in \mathcal{I}: \lambda(i) + w(i) = k} h(i))$,

i.e., the largest sum of heights for all items placed at a given point k . Slicing an item refers to placing a vertical cut at any point inside the item. Either part of the item is then allowed to be placed at any height, see Figure 1 for an illustration. This property allows for lower peak heights in the packing when compared to integral packings.

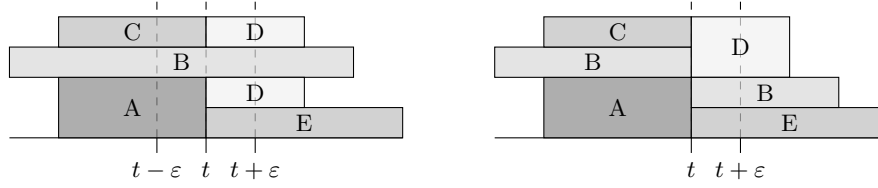
We study the pseudo-polynomial time setting. Pseudo-polynomiality refers to the fact that the algorithm is polynomial in the unary representation of the input. Applying this to DSP means that we are allowed to iterate over the width of the strip W , which is why this setting lends itself especially well to the discussed and widespread application of smaller time intervals.

Related Work

DEMAND STRIP PACKING is an adaptation of the classical STRIP PACKING (SP) problem. SP was first studied in 1980, by Baker et al. They provide a 3-approximation [1]. Study into this problem continued with Coffman et al. [17] providing two shelf-based algorithms. These algorithms achieve a ratio of 3 and 2.7, respectively. This was improved by Sleator [26], who gives a 2.5 approximation. Research stalled until 1994, where Schiermeyer [24] presented a 2 approximation. This ratio later got matched by Steinberg [27], whose algorithm is of particular interest because it only relies on the size of items to ensure its ratio. This makes the algorithm applicable to DSP as well. It took until twelve years later for further improvement to be found, when Harren and van Stee [11] presented a 1.936-approximation. The current best known result is a $(5/3 + \varepsilon)$ -approximation, given by Harren et al. in 2014 [10]. Bladek et al. show that there exists a gap of $5/4$ between the heights of optimal SP and DSP solutions [2]. This is commonly referred to as an integrality gap. As such, algorithms for the classical STRIP PACKING problem produce solutions that are worse by a factor of up to $5/4$ for the demand case. Through a simple reduction from the PARTITION-problem, a hardness of $3/2$ can be shown for SP.

However, in pseudo-polynomial time, improved approximation algorithms and lower hardness results are achievable. Study into this setting began in 2010 by Jansen and Thöle [16]. They provide a $(3/2 + \varepsilon)$ -approximation. This was later improved by Nadiradze and Wiese [21], who present a $(7/5 + \varepsilon)$ -approximation. Research into this setting remained steady and the next improvement was made by Gálvez et al. in the following year [7]. They present a $(4/3 + \varepsilon)$ -approximation. This ratio was matched by Jansen and Rau in the following year, who provide an improvement in running time for their algorithm [14]. The best approximation ratio known is given by Jansen and Rau [15] as well, who provide a $(5/4 + \varepsilon)$ -approximation. These improved approximations are accompanied by a lower theoretical hardness of $5/4$, shown by Henning et al. in 2018 [12].

Variations of strip packing also became fields of thorough research. DEMAND STRIP PACKING was first studied in 2013 by Tang et al., who present a 7-approximation [29]. In the following year, Yaw et al. provide an improved approximation of 4 for cases where all items



■ **Figure 2** An illustration of the transformation from a *PTS* instance (left) to a *DSP* instance (right). The line t represents the current state of the transformation algorithm. No item is sliced by a vertical line in the *PTS* instance. Similarly, no item is sliced by a horizontal line in the *DSP* instance. Note that the height did not change.

have the same width [31]. They also show that DSP is *NP*-hard to approximate for ratios below $3/2$ [31]. In that same year, Ranjan et al. present a 3-approximation [22], which they improve to a 2.7-approximation in the following year [23]. In 2021 two groups independently achieved the best-known approximation ratio of $(5/3 + \varepsilon)$ using different techniques [3, 6]. As a result, both hardness and the best approximation algorithm known match results from the classical STRIP PACKING problem.

Our Contribution

We prove the inapproximability of DSP, even in pseudo-polynomial time, below an approximation ratio of $(5/4)$ through a novel reduction. Through this reduction, we discover a previously unknown connection between DSP and PARALLEL TASK SCHEDULING (PTS). PTS is a scheduling problem where jobs require a certain number of machines, and the aim is to minimize the makespan of the generated schedule. See Section 2 for a complete, formal introduction of the problem. Using this connection, we are able to incorporate some existing algorithms to generate optimal solutions for both DSP and PTS when admitting some resource augmentation. Finally, we present an approximation algorithm almost matching the inapproximability result we show. We provide techniques to Finally, we add techniques to the best-known pseudo-polynomial approximation algorithm for the classical SP problem to make it applicable to DSP. Techniques used in that algorithm do not translate to DSP directly. Structurally, sliced solutions are more difficult to analyze. We provide techniques to partition optimal packings such that they can be structured. We provide a subroutine to place certain items integrally, enabling the structuring. Through it, we can assign items to be packed integrally, even in this demand setting. These new techniques enable us to adapt the existing SP algorithm. This generates an algorithm with an approximation ratio almost matching the inapproximability result we show. Below is a brief summary of the presented results.

- Proof that DSP is strongly *NP*-hard to approximate below a ratio of $(5/4)$.
- A novel transformation algorithm between DSP and PTS.
- A framework for an algorithm producing an optimal solution for DSP when augmenting the width of the strip by a factor of $(3/2 + \varepsilon)$ in polynomial time.
- Frameworks for algorithms producing optimal solutions for PTS when augmenting the number of used machines by either $(5/3 + \varepsilon)$ or $(5/4 + \varepsilon)$ in polynomial and pseudo-polynomial time respectively.
- A $(5/4 + \varepsilon)$ -approximation algorithm for the DSP problem.

Structure of the document

This paper is divided into two parts. We begin by approaching the DSP problem in pseudo-polynomial time by providing a lower bound, proving the inapproximability below ratios of $(5/4)$ even in pseudo-polynomial time. The techniques we develop yield further interesting results for both DSP and PTS when admitting some resource augmentation. These results are also presented in this part.

Afterward, we provide an algorithmic upper bound. We achieve an approximation ratio of $(5/4 + \varepsilon)$. In this section, we provide an overview of the algorithm's steps and the required analysis.

2 Inapproximability of DSP for ratios below $5/4$

We show that the pseudo-polynomial strip packing problem is *NP*-hard to approximate for ratios better than $5/4 \cdot \text{OPT}$. We achieve this result by providing a reduction onto the PARALLEL TASK SCHEDULING problem, where we are given a set of m machines and a set \mathcal{J} of n jobs. Each job $j \in \mathcal{J}$ has a processing time $p(j) \in \mathbb{N}$ and a number of required machines $q(j) \in \mathbb{N}$ assigned. We aim to place all jobs inside a schedule with a minimum makespan. A schedule Σ is a combination of two functions σ, ρ . The first function $\sigma : \mathcal{J} \rightarrow \mathbb{N}$ assigns each job to a starting point in the schedule. The second function $\rho : \mathcal{J} \rightarrow \{M \mid M \subseteq \{1, \dots, m\}\}$ maps each job to a set of machines it is processed on. We can express the makespan $T := \max_{i \in \mathcal{J}} \sigma(i) + p(i)$ as the latest finishing point of any job in \mathcal{J} .

It is known that PARALLEL TASK SCHEDULING is strongly *NP*-hard to solve for all values $m \geq 4$ [5, 12] while there exist pseudo-polynomial time algorithms to solve it for all values $m \leq 3$, see [5]. As a consequence, providing a reduction from DSP onto PTS shows that there is no pseudo-polynomial time algorithm to approximate DSP with a ratio better than $5/4$. This yields the following theorem.

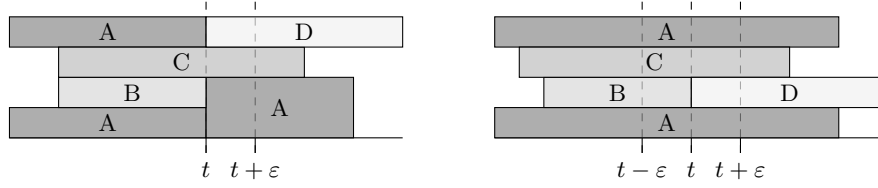
► **Theorem 1.** *DEMAND STRIP PACKING is strongly NP-hard to approximate with a ratio lower than $5/4$.*

Proof. The transformation of an instance I of PTS to an instance I^* of DSP is defined as follows. We begin by creating an item i_j for every job j in I . The width $w(i_j)$ is equal to the processing time $p(j)$ of job j , i.e. $w(i_j) = p(j)$. Similarly, we have $h(i_j) = q(j)$.

As we create exactly one item per job, the value n does not change. Finally, we map the processing time T onto the width of the strip W and the number of machines m onto the desired height of the strip H .

Using this transformation, we show a feasible packing for I^* with height H and width W exists if and only if a feasible schedule for I with makespan T and m machines exists.

Let I be a yes-instance of PTS with m machines and makespan T , and Σ be the generated feasible schedule. Create a strip of width T . Place all transformed items i_j according to their assignment $\rho(j)$, i.e. an item representing a job that has $\rho(j) = m_0, m_1, m_4$ gets placed at height 0, 1 and 4. The items start at $\lambda(i_j) = \sigma(j)$, the starting point of the corresponding job in the schedule. As the height $h(i_j)$ and width $w(i_j)$ of all items is identical to the dimensions $q(j), p(j)$ of their respective jobs, this packing must fit into the strip and have height at most $H = m$. However, note that this packing is not a feasible solution for the DSP problem yet, as items may contain horizontal gaps, i.e. gaps inside items that separate it vertically. See item D on the left side in Figure 3 for an example of such a gap. To remove these gaps and create a feasible packing we use the following procedure.



■ **Figure 3** An illustration of the transformation from a *DSP* instance on the left to a *PTS* instance. The line at t represents the current state of the transformation algorithm. The job A is sliced after this point in the packing because D is placed such that it occupies the top machine. After the swap, all jobs are wholly scheduled on the machines they start on. The height does not change.

Traverse the packing from left to right until you find the first point t in the packing that is infeasible, i.e. where an item i_j contains a horizontal gap. Inspect the point at $t + \varepsilon$. Draw a vertical line and sort all the items that are intersected by this line in ascending order of their heights. Place these items according to this order, beginning at the bottom. After this swap i_j no longer contains a horizontal gap. We can iteratively repeat this procedure, traversing the schedule until the end. After this, the solution Σ is transformed into a feasible solution Λ for DSP. The height of the packing did not change at any point in the procedure, so we have $H = m$. Similarly, the total width of the strip was not exceeded, so we have $W = T$. For an illustration, see Figure 2.

Next, we show that a yes-instance I^* of DSP can be transformed into a feasible schedule Σ' . Let Λ' be the generated packing for I^* with height H and width W . The start point of any item i in Λ is given by $\lambda(i)$. Generate a schedule Σ' containing H machines. Then, place all jobs j_i given by transformed items i at their positions in Λ' , according to $\lambda(i)$. Since the number of required machines $q(j_i) = h(i)$, we still require at most m machines for the packing. Similarly, the makespan of the schedule is at most W , as we have $p(j_i) = w(i)$ for the processing times. However, note that this packing is not a feasible solution for PTS yet, as there may be vertical slices present in some jobs. We utilize the following procedure to remove these slices and create a feasible schedule Σ' .

Traverse the schedule from the beginning, i.e. left, to the end. At the beginning of the schedule, sort all jobs according to their required machines $q(j_i)$, then assign them in ascending order to machines. This does not increase the number of required machines but removes any slices for all these jobs, as they are wholly scheduled on their assigned machines. Traverse the schedule further until we reach a job k that is infeasibly scheduled, i.e. placed on an occupied machine or sliced in some way. This can only occur at the start of some job j' . Call this point in the schedule t . Draw a vertical line at $t - \varepsilon$ and one at $t + \varepsilon$. Note the order of jobs at $t - \varepsilon$. These are scheduled feasibly. Since we had a packing of height $H = m$, we know that the total number of processors required at both lines is at most m . Therefore, there must be gaps inside the schedule at the vertical line $t + \varepsilon$. The total number of empty processors is at least $q(k)$, as all items placed at this line in the packing sum up to a height of at most m . Thus, we can feasibly assign k to some $q(k)$ processors. Since k is a job that is not assigned to processors, we can use the empty processors arbitrarily to place k . For an illustration, see Figure 3. We repeat this for every job at t and traverse the schedule further. As all jobs to the left of t are scheduled feasibly, we must generate a feasible schedule Σ' .

Both procedures we describe here only act once for every beginning item/job. As such, their running time is polynomial. We have shown that any solvable instance I of PTS can be transformed into a feasible packing Λ and vice versa. There exists a transformation from the strongly *NP*-complete problem 3-PARTITION onto PTS, given in [12]. They show

that a schedule using 4 machines with makespan W can not exist unless you can solve the underlying 3-PARTITION instance I_{3P} . Applying our transformation result, therefore, yields the following: If there was an algorithm that solves DSP in pseudo-polynomial time with an approximation ratio less than $5/4$, we can apply the transformation to the instance I_{3P} to generate a feasible demand strip packing instance. Then, the algorithm produces a packing of height $< 5/4 \cdot \text{OPT} = 5$. After transforming the instance back into the PTS instance, we would generate a schedule that utilizes 4 machines and has makespan W . This contradicts the results in [12], proving Theorem 1. \blacktriangleleft

Next, we prove that the running times of these transformation algorithms is indeed polynomial

► **Lemma 1.** *The transformation algorithms from PTS to SSP and vice versa have a running time of $\mathcal{O}(n \cdot n \log(n))$ and $\mathcal{O}(n^2)$ respectively.*

Proof. The procedure to transform a schedule Σ into a feasible packing Λ is computable in $\mathcal{O}(n \cdot n \log(n))$. This is because a horizontal gap can only exist once for every job, at its starting point, because jobs are wholly scheduled on the same machines. Therefore, there can only be at most n points t at which we have to employ the procedure. Due to the sorting, we utilize, a single reordering takes $\mathcal{O}(n \log(n))$ time. This yields a total transformation time of $\mathcal{O}(n \cdot n \log(n))$.

Similarly, the procedure to transform a packing Λ' into a feasible schedule Σ' can be computed in $\mathcal{O}(n^2)$. Any application of the described procedure only occurs once at the beginning of the schedule and then at most once more for every starting point $\lambda(j)$ of a job j . Thus, we have $\mathcal{O}(n)$ many applications. While the initial application requires some sorting of jobs, the remaining applications simply need to check all other jobs j' present at that point in time for their assigned machines. The sorting is feasible in $\mathcal{O}(n \log(n))$, checking the values $\rho(j')$ is feasible in linear time $\mathcal{O}(n)$. As the sorting is only required once, the total running time of this procedure is $\mathcal{O}(n^2)$. \blacktriangleleft

2.1 Applying our transformation using resource augmentation

An important consequence of this computable transformation is that these problems can now be treated as the dual of each other. This directly yields results for both the DSP and PTS problems when allowing some resource augmentation. In the general case, a $(3/2 + \varepsilon)$ approximation algorithm is known for PTS [16]. Implementing the transformations given above and embedding them in a dual approximation framework yields the following.

► **Corollary 2.** *There is a polynomial time algorithm that yields a packing with optimal height for DEMAND STRIP PACKING when we are allowed to augment the width of the strip by a factor of $(3/2 + \varepsilon)$, i.e. consider the strip with width $(3/2 + \varepsilon)W$*

Proof. Let I be an instance of DSP containing n items and a strip of width W . Transform I into a set of jobs \mathcal{J} as described above. Guess the number of machines m required via a binary search. Generate the upper and lower bound via the Steinberg algorithm for I , which guarantees a 2-approximation.

As the lower bound, use $\sum(w(i) \cdot h(i))/W$, i.e. the total area of items distributed evenly across the strip. As the upper bound use the height of the packing given by the Steinberg algorithm for I . This is at most a 2-approximation.

For every guess H of the binary search, use the algorithm given in [16]

to compute a solution for the transformed PTS instance. Check whether the generated makespan T is at most $T \leq (3/2 + \varepsilon) \cdot W$. If this holds, decrease the guess H and repeat the procedure. If we have $T > (3/2 + \varepsilon) \cdot W$, we know that there is no feasible packing with height H and width W . We know that there is an optimal packing of height H and width W if and only if there is a schedule of makespan $T = W$ that uses $m = H$ machines. Further, we know that the algorithm in [16] yields a schedule of makespan $(3/2 + \varepsilon) \cdot \text{OPT} = (3/2 + \varepsilon) \cdot W$. Thus, if the algorithm does not find a solution with makespan less than $(3/2 + \varepsilon) \cdot W$, we know that there can not be a packing with width W and height H . In this case, increase the guess H .

After completing the binary search, we save the solution with the lowest number of required machines $m = H$. Transforming this solution yields a feasible packing with height H and width at most $(3/2 + \varepsilon) \cdot W$. As all algorithm steps run in polynomial time, the whole algorithm also runs in polynomial time. ◀

Similarly, we can generate schedules with optimal makespan T for the PTS problem when we are allowed to augment the number of machines used. If the set of jobs \mathcal{J} is arbitrary, we obtain an optimal makespan by augmenting m by a factor of $(5/3 + \varepsilon)$. To achieve this, we utilize the results given in [3, 6], as they both provide a $(5/3 + \varepsilon)$ for DSP. If the jobs are further parameterized in their processing time, we can apply the pseudo-polynomial time algorithm presented in this paper to reduce the required augmentation to $(5/4 + \varepsilon)$. This yields the following.

► **Corollary 3.** *There is a polynomial time algorithm that yields a packing with optimum makespan for PARALLEL TASK SCHEDULING when we are allowed to augment the number of used machines m by a factor of $(5/3 + \varepsilon)$.*

Proof. The proof closely resembles that of Corollary 2. The core difference is that we implement the dual approximation framework by guessing the makespan $T = W$. We iterate the binary search depending on whether we find a packing with height at most $(5/3 + \varepsilon) \cdot m$. Again, all subroutines used run in polynomial time, so the entire algorithm does as well. ◀

► **Corollary 4.** *There is a polynomial time algorithm that yields a packing with optimum makespan for PARALLEL TASK SCHEDULING where the processing time is parameterized, when we are allowed to augment the number of used machines m by a factor of $(5/4 + \varepsilon)$.*

Proof. The proof resembles that of Corollary 2. The dual approximation framework is implemented by guessing the makespan $T = W$. We iterate the binary search depending on whether we find a packing with height at most $(5/3 + \varepsilon) \cdot m$. All subroutines used run in polynomial time, so the entire algorithm does as well. ◀

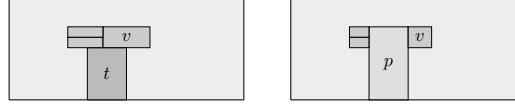
With the hardness and transformation algorithms shown, let us now focus our attention on constructing the algorithm that achieves the approximation ratio of $(5/4 + \varepsilon)$.

3 A $(5/4 + \varepsilon)$ approximation for DSP in pseudo-polynomial time

We provide an algorithm that proves the following theorem.

► **Theorem 5.** *There is an algorithm to solve the DSP problem with approximation ratio $(5/4 + \varepsilon)\text{OPT}$ in $\mathcal{O}(n \log(n)) \cdot W^{\mathcal{O}_\varepsilon(1)}$ time, where \mathcal{O}_ε neglects all factors depending on $1/\varepsilon$.*

To prove this theorem, we present an algorithm that fulfills the desired properties. Before we can begin summarizing this algorithm however, we must briefly introduce some concepts essential to the functionality of the algorithm.



■ **Figure 4** A basic illustration of the pseudo item generation. To the left we see the original items, in particular the defining tall item t . To the right we see the newly generated pseudo item p , with increased height but the same width as the original item constructing it. The remaining vertical items might get separated horizontally by later routines.

Boxes. When talking about boxes, we refer to some rectangular, axis-aligned objects with a width and a height. All boxes we describe are also able to be sliced as long as they are placed contiguously, i.e. their end point is at the sum of their starting point and their width. When illustrating proofs, we mostly ignore this sliceability of boxes in an effort to increase clarity. When discussing boxes, we *do not* imply that they are not sliced.

Pseudo Items Pseudo items are a set of combined items that are placed adjacent to each other inside some packing. We combine these items and call this combination a new pseudo item. These items are especially relevant when we discuss the separation of vertical and tall boxes in the following. For a simple illustration of this process, see Figure 4. Note that the construction of pseudo items may introduce horizontal separation in some *vertical* items. Such items are later placed integrally into gaps left open by our generated packing.

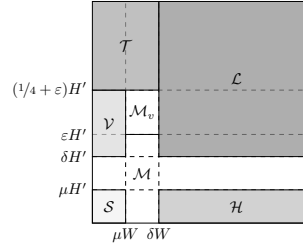
First, we give an overview of its functionality and steps. Then, we continue by proving the correctness of each step. Due to space concerns, we omit some technical analyses and instead focus on providing a sketch of the desired algorithm.

We begin by rounding and categorizing the instance. Then, we show that any optimal solution can be transformed into a structured solution, producing only a small loss in packing height. Finally, we show that we can fill this structure with items to generate a feasible packing. The algorithm can be described as such:

- 1 Define $\varepsilon' = \mathcal{O}(\varepsilon)$, formulate lower and upper bounds on the optimal packing height $H, 2H$. Scale the item heights in the instance such that they are in $\{1, \dots, n/\varepsilon'\}$
- 2 Guess values $H' \in [H, \dots, 2H]$ via binary search and test each value for a feasible packing in the following manner:
 - 3 Round and scale all items in the instance according to H' and ε' . Categorize items into item-types depending on their dimensions.
 - 4 Discard small and medium items. Guess the partition of the optimal packing into boxes. For each guessed partition, attempt the following:
 - 5 Place items in item-types inside their respective boxes. If this placement is impossible, we know the guessed partition must be wrong and discard it. If no partition permits a feasible placement, the guessed optimal value H' must be wrong.
 - 6 Place the discarded medium items atop the generated packing.
- 7 Return the packing generated for the smallest value H' .

Step 1

Set $\varepsilon' = \min\{1/4, (1/\lceil 10/\varepsilon \rceil)\}$ for computational reasons. The values $H, 2H$ can be generated using Steinbergs algorithm [27], which computes a solution with height at most $2 \cdot \text{OPT}$. Thus, we apply Steinbergs algorithm to the instance and take the resulting packing height as $2H$, and halve it to obtain H .



■ **Figure 5** An illustration of the partitioning of items. Items are partitioned by their width and height according to this figure.

Step 2

We use a framework pioneered in [13]. Instead of computing the optimal height directly, we guess the optimal height inside the bounds given in step 1 and attempt to compute a packing of this height for every guess. As long as this is feasible, we decrease the guessed height, increasing it otherwise.

Step 3

From this point on, we assume that the guessed value H' given by step 2 is correct, i.e. $H' = \text{OPT}$. Showing that all steps successfully compute proves the correctness of the algorithm. We round the items according to their sizes. We select some values μ, δ such that medium items have a small total area and discard them later. Small items are discarded as well. We categorize items as either large (\mathcal{L}), tall (\mathcal{T}), vertical (\mathcal{V}), medium-vertical (\mathcal{M}_v), medium (\mathcal{M}), horizontal (\mathcal{H}) or small (\mathcal{S}) according to some values μ, δ . See Figure 5 for an intuitive illustration. We partition items i depending on their width $w(i)$ and height $h(i)$ as follows:

- $\mathcal{L} := \{i \in I \mid h(i) > \delta H', w(i) \geq \delta W\}$ as the set of large items,
- $\mathcal{T} := \{i \in I \mid h(i) \geq (1/4 + \varepsilon)H', w(i) < \delta W\}$ as the set of tall items,
- $\mathcal{V} := \{i \in I \mid \delta H' \leq h(i) < (1/4 + \varepsilon)H', w(i) \leq \mu W\}$ as the set of vertical items,
- $\mathcal{M}_v := \{i \in I \mid \varepsilon H' \leq h(i) < (1/4 + \varepsilon)H', w(i) \leq \mu W\}$ as the set of vertical medium items,
- $\mathcal{H} := \{i \in I \mid h(i) \leq \mu H', w(i) \geq \delta W\}$ as the horizontal items,
- $\mathcal{S} := \{i \in I \mid h(i) \leq \mu H', w(i) \leq \mu W\}$ as the set of small items and
- $\mathcal{M} := \{i \in I \mid h(i) < \varepsilon H', \mu W < w(i) \leq \delta W\} \cup \{i \in I \mid \mu H' < h(i) \leq \delta H'\}$ as the set of medium sized items.

We use techniques developed for the classical SP problem [15] to generate values for μ, δ accordingly to group the items. As a result, the total area of medium and medium-vertical items is small. Next, we introduce the values for μ, ε used in our rounding scheme and categorization of items.

► **Lemma 2.** *Let $f : \mathbb{R} \rightarrow \mathbb{R}$ be any function such that $1/f(\varepsilon)$ is integral. Consider the sequence $\sigma_0 = f(\varepsilon), \sigma_{i+1} = \sigma_i^2 f(\varepsilon)$. There is a value $i \in \{0, \dots, (2/f(\varepsilon)) - 1\}$ such that the total area of the items in $\mathcal{M} \cup \mathcal{M}_v$ is at most $f(\varepsilon) \cdot W \cdot \text{OPT}$, if we set $\delta := \sigma_i$ and $\mu := \sigma_{i+1}$.*

Proof. This is a standard rounding technique that can be proven using the pigeonhole principle. The sequence σ and the corresponding choices of δ and μ builds a sequence of $2/f(\varepsilon)$ sets $\mathcal{M}_{\sigma_i} \cup \mathcal{M}_{v\sigma_i}$. Each item $i \in I$ can occur in at most two of these sets, either because of its width or its height. Since the total area of all items is at most $W \cdot \text{OPT}$ one of these sets must have an area which is at most $f(\varepsilon) \cdot W \cdot \text{OPT}$. ◀

For this application, it is sufficient to choose $f(\varepsilon) = \varepsilon^{13}/k$ for a constant $k \in \mathbb{N}$ which has to fulfill certain properties we address later on. Since $1/\varepsilon \in \mathbb{N}$, we have that $1/f(\varepsilon) \in \mathbb{N}$. Let δ and μ be values defined as in Lemma 2. Note that $\sigma_i = f(\varepsilon)^{2^{i+1}-1}$ and, therefore $\delta \geq \sigma_{f(\varepsilon)-1} \geq (\varepsilon^{13}/k)^\ell \in \varepsilon^{\mathcal{O}(\ell)}$, where $\ell := 2^{2k/\varepsilon^{13}}$, i.e. $\delta \geq \varepsilon^{2^{\mathcal{O}(1/\varepsilon^{13})}}$. In the following steps, we need δ to be of the form ε^x for some $x \in \mathbb{N}$. Therefore, define $\delta' := \varepsilon^x$, such that $x \in \mathbb{N}$ and $\delta' \leq \delta \leq \delta'/\varepsilon$. Note that $\mu := \delta^2 \varepsilon^{13}/k \leq (\delta'/\varepsilon)^2 \varepsilon^{13}/k = \delta'^2 \varepsilon^{11}/k$ and $\mu = \delta^2 \varepsilon^{13}/k \geq \delta^2 \varepsilon^{13}/k \geq \delta'^2 \varepsilon^{13}/k$. In the following, we rename $\delta := \delta'$ for all the steps. By this choice, it still holds that the set of medium sized items has a total area of at most $(\varepsilon^{13}/k) \cdot W \cdot \text{OPT}$ because, by reducing δ and not changing μ , we only remove items from this set.

Observation 2. Since each item in \mathcal{M}_v has a height of at least εOPT and width of at least $\mu W \geq \delta^2 \varepsilon^{13}/kW$, i.e. an area of at least $(\delta^2 \varepsilon^{14}/k) \cdot W \cdot \text{OPT}$, it holds that

$$|\mathcal{M}_v| \leq (\varepsilon^{13}/k) \cdot W \cdot \text{OPT} / ((\delta^2 \varepsilon^{14}/k) \cdot W \cdot \text{OPT}) = 1/\delta^2 \varepsilon.$$

After we have found the corresponding values for δ and μ and partitioning the set of items in the instance accordingly, we round the height of all items with height at least δOPT .

We then round the instance such that all items with significant height have some canonical height.

► **Lemma 3.** *Let $\delta = \varepsilon^k$ for some value $k \in \mathbb{N}$. At loss of a factor $(1+2\varepsilon)$ in the approximation ratio, we can ensure that each item i with height $\varepsilon^{\ell-1} \text{OPT} \geq h(i) \geq \varepsilon^\ell \text{OPT}$ for some $\ell \in \mathbb{N} \leq k$ has height $k_i \varepsilon^{\ell+1} \text{OPT}$ for a value $k_i \in \{1/\varepsilon, \dots, 1/\varepsilon^2 - 1\}$. Furthermore, the items upper and lower borders can be placed at multiples of $\varepsilon^{\ell+1} \text{OPT}$.*

Proof. Let there be a given packing inside the strip with height OPT . We stretch it vertically by a factor of $1 + 2\varepsilon$. As a result, any point (x, y) in the original packing now corresponds to a point $(x, (1 + 2\varepsilon)y)$ in this stretched packing. Let $i \in \mathcal{L} \cup \mathcal{T} \cup \mathcal{V}$ be an item with $\varepsilon^{\ell-1} \cdot \text{OPT} \geq h_i \geq \varepsilon^\ell \cdot \text{OPT}$ and let y_T and y_B be the y -coordinates of its top and bottom edges in the original strip. As i can be sliced, both y_T and y_B are sets that can contain up to $w(i)$ many values. As items cannot be sliced horizontally, there must be the same amount of values k in the sets y_T and y_B . Define the stretched y -coordinates as $\bar{y}_{T_k} = (1 + 2\varepsilon)y_{T_k}$ for every entry $k \in y_T$. Define \bar{y}_B analogously. As a consequence, we have

$$\bar{y}_{T_k} - \bar{y}_{B_k} = (1 + 2\varepsilon)(y_{T_k} - y_{B_k}) = (1 + 2\varepsilon)h_i.$$

Now, we change the y -coordinates of i in the new strip to

$$y'_{B_k} := \bar{y}_{B_k} + \varepsilon h_i$$

and

$$y'_{T_k} := \bar{y}_{T_k} - \varepsilon h_i.$$

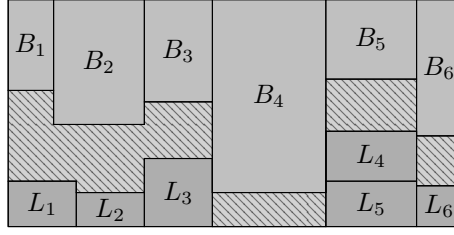
For every value of k we get

$$y'_{T_k} - y'_{B_k} = \bar{y}_{T_k} - \bar{y}_{B_k} - 2\varepsilon h_i = (1 + 2\varepsilon)h_i - 2\varepsilon h_i = h_i.$$

We have $h_i \geq \varepsilon^\ell \text{OPT}$, which implies

$$y'_{T_k} - y'_{B_k} = \bar{y}_{T_k} - \bar{y}_{B_k} = \varepsilon h_i \geq \varepsilon^{\ell+1} \cdot \text{OPT}.$$

This ensures that there is an integer j_{T_k} such that $j_{T_k} \cdot \varepsilon^{\ell+1} \cdot \text{OPT} \in [y'_{T_k}, \bar{y}_{T_k}]$ for the interval $[y'_{T_k}, \bar{y}_{T_k}]$. Analogously, there exists a set of integers j_{B_k} so that $j_{B_k} \cdot \varepsilon^{\ell+1} \cdot \text{OPT} \in [\bar{y}_{B_k}, y'_{B_k}]$.



■ **Figure 6** The partitioned optimal packing generated by Lemma 5. Named boxes B_i contain only tall and vertical items. The hatched area contains only boxes for horizontal items. Finally, the lowest areas contain only boxes for large or medium vertical items. These boxes are named according to the item placed inside. Boxes for horizontal items may be sliced.

We change the top y -coordinates of item i to $y''_{T_k} = j_{T_k} \cdot \varepsilon^{\ell+1} \cdot \text{OPT}$ and similarly the bottom coordinates to $y''_{B_k} = j_{B_k} \cdot \varepsilon^{\ell+1} \cdot \text{OPT}$. It may occur that $y''_{T_k} - y''_{B_k} > h'_i := \lceil h_i / (\varepsilon^{\ell+1} \text{OPT}) \rceil \cdot \varepsilon^{\ell+1} \cdot \text{OPT}$. In this case we increase y''_{B_k} by $(y''_{T_k} - y''_{B_k}) - h'_i$, such that $h'_i = y''_{T_k} - y''_{B_k}$. Any part of the item i cannot intersect another item part because it is placed inside the stretched version of that item part. Thus, when we change the height h_i of each item $i \in \mathcal{L} \cup \mathcal{T} \cup \mathcal{V}$ to $h'_i = \lceil h_i / (\varepsilon^{\ell+1} \text{OPT}) \rceil \cdot \varepsilon^{\ell+1} \cdot \text{OPT}$, where $\ell \in \mathbb{N}$ is chosen such that $\varepsilon^{\ell-1} \text{OPT} \geq h_i \geq \varepsilon^\ell \text{OPT}$. Note that $\lceil h_i / (\varepsilon^{\ell+1} \text{OPT}) \rceil \in \{1/\varepsilon, \dots, 1/\varepsilon^2\}$ since $\varepsilon^{\ell-1} \text{OPT} \geq h_i \geq \varepsilon^\ell \text{OPT}$. Since the value h' does not exceed the stretched item height, we increased the optimal solution by a factor of $1 + 2\varepsilon$ at most. ◀

This eases the placement of these items. Further, all of these items can be placed on some imaginary grid-lines inside an optimal packing of height H' . This is significant when we argue the structure of an optimal packing.

Step 4

We show that we can partition an optimal packing into a set of boxes in this step. Each box only contains a certain subset of item types. For this to be feasible, we first must reduce the different starting points of horizontal jobs to $\mathcal{O}(1/\varepsilon\delta)$ with only a loss of $\mathcal{O}(\varepsilon H')$ in the packing height. This is done using the technique given in [3] for DSP.

► **Lemma 4.** [3] *At a loss of at most $\mathcal{O}(\varepsilon H)$, we can reduce the number of used starting points of horizontal items to $\mathcal{O}(1/\varepsilon\delta)$.*

Next, we guess these points to partition the packing. We have to take special care when discussing the optimal structure of an DSP-packing, as we can not argue over the vertical placement of items in the optimal solution. Therefore, the following proof requires some novel insights into the utilization of slicing.

► **Lemma 5.** *We can partition the rounded instance into at most $\mathcal{O}_\varepsilon(1)$ boxes. The set of boxes can be partitioned into sets $\mathcal{B}_L, \mathcal{B}_H$ and $\mathcal{B}_{T \cup V}$ such that*

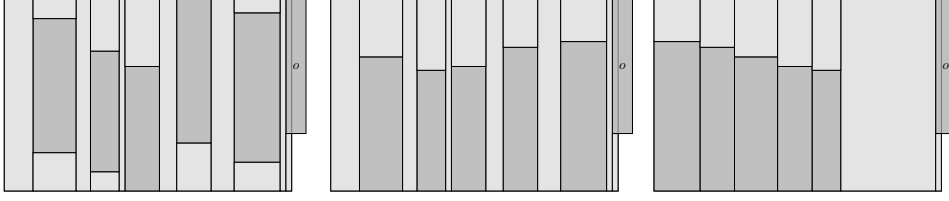
- *boxes in \mathcal{B}_L contain only one item that is either large or medium vertical, i.e. it is in \mathcal{L} or \mathcal{M}_v . The total number of such boxes is $|\mathcal{L}| + |\mathcal{M}_v| \leq \mathcal{O}_\varepsilon(1)$,*
- *\mathcal{B}_H consists of $\mathcal{O}_\varepsilon(1)$ boxes of height $\varepsilon\delta \text{OPT}$, each of them containing at least some item in \mathcal{H} but only items in $\mathcal{H} \cup \mathcal{S}$,*
- *$\mathcal{B}_{T \cup V}$ consists of $\mathcal{O}_\varepsilon(1)$ boxes, each of them containing items in $\mathcal{T} \cup \mathcal{V} \cup \mathcal{S}$.*

Proof. Consider the rounded optimal packing. In it, each item has a defined starting point on the horizontal axis. As such, we can consider any singular item to be packed fully integrally and simply shift the slicing of other items around such that they retain their original starting positions. We utilize this technique to define the boxes items fit in fully integrally, while still allowing them to be sliced again in order to be identical to the optimal packing again. We begin this procedure with the items in $\mathcal{L} \cup \mathcal{M}_v$. Each of these items i gets a personal box with width w_i , and height h_i , placed at exactly its starting position λ_i . Since the starting point remains the same, the packing remains optimal.

Next, we define the boxes \mathcal{B}_H . We begin at the left side of the packing. Traverse the packing towards the right until you find the starting point of a horizontal item i . Then, check all items in \mathcal{H} with the same starting point $\lambda(i)$ and select the item j with the largest width. The box B we now define starts at $\lambda(i) = \lambda(j)$ and ends at $\lambda(i) + w(j)$. Its height is $\varepsilon\delta$. We iteratively add items to this box in the following manner: first, find all items ℓ with $\lambda(i) \leq \lambda(\ell) \leq \lambda(i) + w(j)$, i.e. all items that can be fully contained inside the newly formed box. Next, order these items in descending order regarding their width. Then, take the item with the largest width and add it into the box. Repeatedly add such items until the next item we added would break the height of the box, i.e. there is a coordinate x with $\sum_{i \in B, \lambda(i) \leq x \leq \lambda(i) + w(i)} h(i) > \varepsilon\delta$. Continue checking whether we can add items to this box until there are no possible items left. As the last step, take the widest possible item ℓ to add that exceeds the top border of the box. Take further items that do not overlap with ℓ or with each other, until no such items exist anymore, and add them to the box as well. Finally, we simply repeat this procedure until we have reached the right border of the packing. Clearly, these boxes contain only items in \mathcal{H} . At every point in the box, at most one item overlaps the top border. The items in \mathcal{H} have a width of at least δW and the boxes have a height of $\varepsilon\delta\text{OPT}$. Thus, we can stack at most $(1 + 2\varepsilon)/(\varepsilon\delta)$ such boxes atop one another, and can have at most $1/\delta - 1$ boxes beside each other. Therefore, the total amount of boxes in \mathcal{B}_H is at most $(1 + 2\varepsilon)/(\varepsilon\delta) \cdot (1/\delta - 1) \leq (1 + 2\varepsilon)/(\varepsilon\delta^2) - 2$. Because we reduced the amount of starting points of horizontal items inside the packing, we generate at most this amount of boxes.

Finally, consider the boxes $\mathcal{B}_{T \cup V}$. For these, imagine the boxes \mathcal{B}_H and \mathcal{B}_L forming a profile. That is, they are all sliced towards the top at their original positions. Since both of these sets of boxes have a height that is a multiple of $\varepsilon\delta$, this profile must be at multiples of $\varepsilon\delta$ as well. Traverse this profile from left to right, until we reach the first point x at which the height of the profile changes, i.e. $\sum_{i, \lambda(i) \leq x \leq \lambda(i) + w(i)} h(i) \neq \sum_{j, \lambda(j) \leq x-1 \leq \lambda(j) + w(j)} h(j)$. Draw a vertical line at this coordinate x . This defines the right border of the first box, and the left border of the second box. Continue this procedure, iteratively drawing vertical lines that define box borders, until we arrive at the right border of the strip. These borders define the boxes in $\mathcal{B}_{T \cup V}$. Clearly, they contain only items in $\mathcal{T} \cup \mathcal{V} \cup \mathcal{S}$. We have at most $(1 + 2\varepsilon)/(\varepsilon\delta^2) - 2$ boxes in $\mathcal{B}_L \cup \mathcal{B}_H$. Since we drew at most two lines for each box in \mathcal{B}_H and \mathcal{B}_L and each such line touches at most two boxes in $\mathcal{B}_{T \cup V}$, we have at most $2(1 + 2\varepsilon)/(\varepsilon\delta^2)$ boxes in $\mathcal{B}_{T \cup V}$. For an illustration of this procedure, see Figure 6.

All types of boxes we generated in this step have their upper and lower border at a multiple of $\varepsilon\delta\text{OPT}$. This is because, for large, tall, vertical and medium vertical items, our rounding in Lemma 3 ensures that the items contained in those boxes start at multiples of such a value, as $\delta = \varepsilon^x$ for a value $x \in \mathbb{N}$. Additionally, while the horizontal items might not have their borders at such values, recall how we generated their boxes. We have shifted the packing such that they were on top of the remaining two sets of boxes, $\mathcal{B}_{T \cup V}$ and \mathcal{B}_L . Since these have their upper and lower borders at multiples of $\varepsilon\delta\text{OPT}$, the lowest set of horizontal



■ **Figure 7** Illustration of a packing of tall items for a box \check{B} with height less than $1/2 H'$. Tall items are colored orange and vertical items green. To the left we see the box in the optimal packing, with the slices for pseudo items already generated. The second image represents the box with all tall items sliced to the bottom and their pseudo items fused on top of them. The final image represents the resulting box, sorted by height of tall items in movable slices. Note that the overlapping item o or its slice is never moved, and thus requires one additional box.

boxes in \mathcal{B}_H must have the same property. As each of these boxes has a height of $\varepsilon \delta \text{OPT}$, all boxes further above must also have their upper and lower borders at those values. ◀

After this partition, we discard the small items. We show later that we can place them inside some gaps without increasing the packing height. The medium items get discarded and placed atop the packing using Steinberg's algorithm, as they have a small total area due to Lemma 3.

The boxes \mathcal{B}_L and \mathcal{B}_H only contain either a single item or items of one type. As such, we can feasibly fill them in *step 5*. However, the boxes \mathcal{B}_{TUV} do not share this property yet. To make them fillable, we must therefore partition them further.

We partition each box $B \in \mathcal{B}_{TUV}$ separately. We utilize different routines depending on the height of B . The core complexity here is derived from the number of tall items that can be stacked atop one another. As such, the three types of boxes we inspect are ones with height $h(B) \leq 1/2 H'$, those with height $h(B) \in (1/2 H', 3/4 H']$, and boxes with height $h(B) \in (3/4 H', H']$. The boxes are categorized as such because there can only be one tall item at any given point in the smallest boxes, two tall items atop another in the second type, and finally, up to three tall items atop another in the largest type of box.

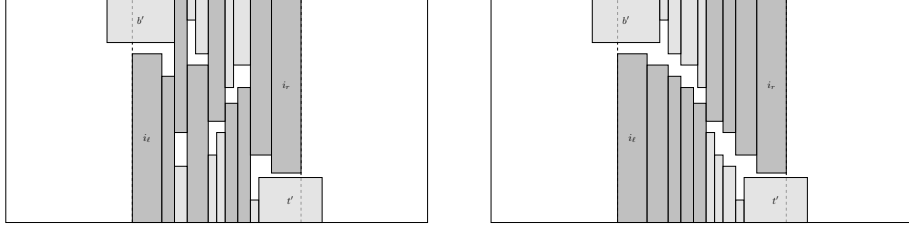
An important new challenge derives for these boxes compared to the classical SP problem. The core idea of the restructuring arguments for each box is the *assignment* of tall items to the bottom, top, or middle of the box. However, finding such an assignment is much harder when admitting sliced items. Several items i_1, i_2, i_3 that are placed atop another can be sliced in a way where each of these items is the bottom-most at one point in the optimal packing. The most complex assignment is required for boxes with height $h(B) \in (3/4 H', H']$.

Using this assignment subroutine, it becomes feasible to structure the three types of boxes. In this structure, each type of box contains only at most $\mathcal{O}_\varepsilon(1)$ sub-boxes.

Boxes \check{B} with $h(\check{B}) \in (1/4 \cdot H', 1/2 \cdot H']$

We solve these boxes by assigning each tall item to the bottom of the box. Then, we slice the box at the borders of tall items. We sort these slices in descending order of the height of tall items inside them and place them as such.

► **Lemma 6.** *Let $\check{B} \in \mathcal{B}_{TUV}$ be a box with height $h(\check{B}) \in (1/4 H', 1/2 H']$. There is a rearrangement of items in \check{B} such that there are at most $\mathcal{O}(1/\varepsilon)$ different boxes for tall items, and at most $\mathcal{O}(1/\varepsilon)$ different boxes for vertical items.*



■ **Figure 8** An illustration of the repacking for a box \tilde{B} with height in $(1/2H', 3/4H']$. Illustrated here is the first step where we find the tallest items i_ℓ, i_r and generate a sub-box between these two. To the left we see the original sub-box and to the right we see the sorted result. All items either touch the top or bottom of the packing after our assignment, as is required.

Proof. This proof is an adaptation of a similar procedure from [14]. The height of \tilde{B} clearly implies that there can never be two tall items placed above one another, since they do not fit into the box otherwise. Therefore, through slicing, we can assume that all tall items touch the bottom of the box with their bottom border. If this were not the case, we simply slice these tall items downward and the vertical items touching the ground upwards. Similarly, we can stack all vertical items into pseudo items that touch the ground as well.

Before we reorder the inside of the box, we define any items that intersect the left or right box border as immovable. Next, we construct slices around the tall items. For that, we draw a vertical line at the left and right border of a tall item. Each area between two such lines is considered a movable slice. We sort these movable slices in descending order of the tall items inside them. For an illustration of this procedure, see Figure 7. Through this reordering, we get at most $1/\varepsilon$ boxes for movable tall items, since the heights of tall items were rounded to multiples of εOPT . Since one tall item can overlap each box border, we might need two further boxes for these items. Therefore, the total number of boxes for tall items is in $\mathcal{O}(1/\varepsilon)$.

On top of each box for tall items, we introduce at most one box for vertical items. Finally, there might be movable slices that do not contain any tall item. These are placed adjacent to each other at the right side of \tilde{B} after our reordering. For these, we construct one box with height $h(\tilde{B})$. In total, that leaves us with $1/\varepsilon$ boxes above the movable tall items, two boxes above the tall items that may intersect the borders of \tilde{B} and one final box of height $h(\tilde{B})$. Thus, the total number of boxes for vertical items is in $\mathcal{O}(1/\varepsilon)$. ◀

Boxes \tilde{B} with $h(\tilde{B}) \in (1/2 \cdot H', 3/4 \cdot H']$

► **Lemma 7.** *Let $\tilde{B} \in \mathcal{B}_{T \cup V}$ be a box with height $h(\tilde{B}) \in (1/2H', 3/4H']$. We find a rearrangement of items in \tilde{B} such that we generate at most $\mathcal{O}_\varepsilon(1)$ sub-boxes containing only either tall or vertical items, where each sub-box \tilde{B}_T for tall items only contains items of the same height and the vertical items can be packed fractionally into boxes for vertical items.*

Proof. Before we begin with the proof itself we have to handle the assignment of tall items to either the top or bottom of the packing. Thankfully, as there can be at most two tall items placed on the same vertical line, this is straightforward. We draw two horizontal lines, one at $1/4H'$ and the other at $h(\tilde{B}) - 1/4H'$. If a tall item t overlaps both of these lines, we know that no other tall item can be placed at the same x-coordinate as t . Thus, we shift all such items to the bottom of the box, shifting vertical items up accordingly. If any tall

item only intersects the bottom line, we shift it down to the bottom of the box. Finally, if any tall item only intersects the top line, we shift it upwards such that it touches the top of the box. Clearly, no overlaps can be generated from this procedure and the packing stays feasible as we did not shift the x-coordinates of any item.

We proceed by generating our pseudo-items for this box. We draw a vertical line at the beginning and end of each tall item that is placed at the bottom of the box. We generate pseudo-items by fusing the items between these lines. If any tall items are placed at the top of the box we select the one with the lowest bottom between each of these lines and let this item mark the top of the pseudo item, if enough items are present. Thereby we have at most one pseudo item at each vertical line, which is placed either at the top of the box if there is a tall item below this pseudo item intersecting the line at $1/4H'$ or at the bottom of the box otherwise.

We separate this proof into two cases. In the trivial case, no tall item overlaps the left or right border of \tilde{B} . In this case, we know from [21] that we can simply sort the items touching the top in descending order of heights and items touching the bottom of the box in ascending order of height. In the case where we have tall and pseudo items of the same height, we place them such that the tall items are left of the pseudo items. Through this reordering, no items can overlap and we generate at most one box for any tall or pseudo item at the top and the same amount for items at the bottom. Because of the minimum width of tall items, we can have at most $\mathcal{O}_\varepsilon(1)$ many boxes. It can happen that we separate some vertical items horizontally in this procedure while reordering the pseudo items. We show how to pack these items integrally again after handling all box sizes of boxes in $\mathcal{B}_{T \cup V}$.

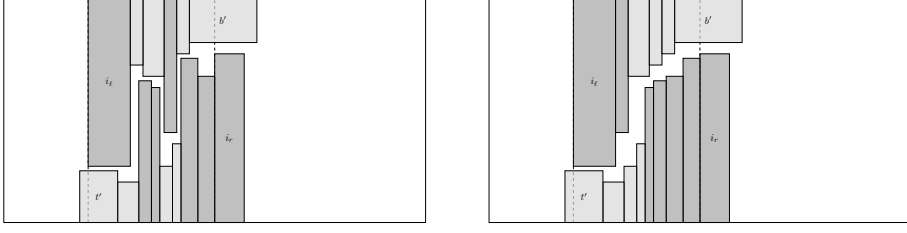
Let us now consider the case in which at least one tall item intersects the box border. Let h_b be the height of the tallest item touching the bottom of the box and b_ℓ and b_r be the left- and rightmost items with height h_b . Define h_t, t_ℓ, t_r in the same manner. Further, let i_ℓ be the leftmost item in $\{b_\ell, t_\ell\}$ and i_r be the rightmost in $\{b_r, t_r\}$. Let us assume, w.l.o.g. that $i_\ell = b_\ell$ and $i_r = t_r$.

We draw a vertical line on the left border of b_ℓ . The item cut with this line is defined as a new immovable item b' . We do the same on the right side of t_r and name the cut item t' . Now, we sort all movable items between the two lines. Items touching the top are sorted in ascending order regarding their heights, while items at the bottom are sorted in descending order.

This reordering ensures that no two items overlap. There is no item that overlaps b' as the tallest item that was reordered at the bottom has height h_b . Because b' was placed above b_ℓ , this cannot lead to any overlap. The same holds for t' .

The other points at which overlaps can occur is between these two items b' and t' . Assume that there is a point $p = (x, y)$ at which two items i_b and i_t overlap. Since all items at the top to the left of i_t and all items at the bottom to the right of i_b have a height of at least $h(i_t)$ and $h(i_b)$ respectively, the total width of the strip between b' and t' is covered by items that must overlap if they are placed above each other in any configuration. Therefore, the strip must have had an overlap before this reordering, leading to a contradiction. Finally, this means there cannot be a new overlap created by this reordering.

As the next step, we inspect the items placed at the top with height $h(b')$. We remove these items, shift the items with height smaller than $h(b')$ to the right as far as possible, and place the removed items in this area. Clearly, the width of the packing is the same and does



■ **Figure 9** The continuation of the algorithm to solve a box \tilde{B} . Here, i_r is the same item as i_ℓ in the first step of the algorithm, i.e. we are inspecting the sub-box to the left of our starting point. Again, we find another tall item to generate this sub-box, and solve it in the same manner we solved the sub-box in the first step. This procedure is repeatable until the entire box \tilde{B} is packed in a structured manner.

not lead to overlap. Furthermore, there cannot be any vertical overlap, as the item i_r below b' is the tallest and fits below the items of height $h(b')$, so all other items must do so as well. Similarly, the items with height smaller than $h(b')$ can obviously not lead to overlap if we shift them above smaller items. We do the same thing for items of height $h(t')$ on the other side. We do this to only require one box of height $h(b')$ and $h(t')$. For an illustration of this step, see Figure 8.

We have again generated $\mathcal{O}_\varepsilon(1)$ boxes up to this point. This is because we generate at most one box for either size on the top and the bottom of the strip each. Importantly, the total number of different heights touching the top or bottom left of i_ℓ is reduced by one. This is due to the strip containing all items of height $h(i_\ell)$. The same holds for the right side of i_r .

The procedure described above can be repeated iteratively to fully reorder the box \tilde{B} . To do that, we simply repeat the procedures both to the left of i_ℓ and the right of i_r with the box borders of \tilde{B} as the furthest outer part. Again, by finding the outermost tallest items inside these areas, we reduce the number of heights in the remaining areas by at least one. For an illustration of this continuation, see Figure 9. We repeat this procedure until one of the following occurs:

1. The height of the tallest item touching the top and the height of the tallest item touching the bottom summed up is at most $h(\tilde{B})$.
2. The item i_r becomes the immovable item overlapping the left border. To the left of i_ℓ , the procedure ends when i_ℓ becomes the item overlapping the right border.

If the first condition occurs, we can simply sort the items on the top and bottom in ascending order of heights, never generating any overlap. This fully solves the box \tilde{B} , again generating at most $\mathcal{O}_\varepsilon(1)$ boxes for tall and pseudo items.

If the second condition occurs, we simply arrived at the last iteration of the reordering procedure. Afterwards, the vertical line is placed exactly at the box border, thereby finishing this procedure. Again, we generate at most $\mathcal{O}_\varepsilon(1)$ boxes in this final iteration.

Due to the first break condition, we repeat the procedure at most until the tallest item in the packing has a height of $h(\tilde{B}/2)$. Since we reduce the number of different heights inside the box by at least one in each iteration of this procedure, we can have at most $\mathcal{O}_\varepsilon(1)$ steps until both the tallest item at the top and bottom have at most this height. Finally, since we double this procedure on the left and right side of the initial sub-box, we create at most $\mathcal{O}_\varepsilon(1)$ boxes for tall items, and $\mathcal{O}_\varepsilon(1)$ boxes for pseudo items in total. ◀

We later fill the boxes for pseudo items with vertical items that were separated horizontally in any procedure.

Boxes B with $h(B) \in (3/4H', H']$

This is the most complex type of box. To solve it, we first utilize Lemma 8 to infer some structure.

► **Lemma 8.** *Given a box $B \in \mathcal{B}_{T \cup V}$ with height $h(B) > 3/4H'$ only containing tall and vertical items, we can find an assignment for tall items such that they are fully assigned to either the bottom, top or middle of the box.*

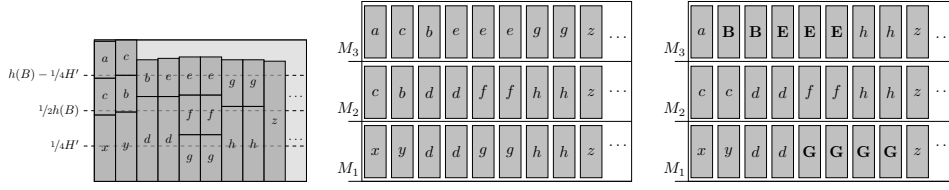
Proof. Begin by partitioning the box into slices of width 1. For each such slice, we sort the tall items inside according to their height and order them tallest at the bottom ascending. The vertical items inside these slices get placed at the top of the last tall item. Afterward, we draw three horizontal lines throughout the box. One at $1/4H'$, the second at $1/2h(B)$ and the final one at $h(B) - 1/4H'$. After sorting the tall items, this selection ensures that each tall item is intersected by at least one of these lines.

We now transform the box into a scheduling problem on three machines. We discard vertical items for this transformation. Each machine represents one of the horizontal lines. Item-parts get assigned to a machine. One part gets assigned for every vertical slice of the box. They are assigned to the machine representing the horizontal line that intersects the item in a given vertical slice. The height of such a transformed item is the number of machines it is initially scheduled on. We show that we can transform the schedule generated by this transformation into one where each item is assigned to the correct number of machines, items assigned to at least two machines are always assigned to the middle machine and items are fully assigned to the machines they start on.

After this transformation, we obtain a schedule. Let us first discuss the trivial cases. If there are three items, each occupying one machine at any point in time, we can reorder them in any manner we want and fulfill the criteria. Similarly, an item requiring all three machines always fulfills the criteria above. Thus, the only point at which we must carefully reorder the items is when an item occupies two machines. As these items must always occupy the middle machine, we can place their remaining part either at the top or bottom. Furthermore, there can only be another item with a height of 1 scheduled in parallel to this item.

Traverse the schedule from left to right. Let d be the first item of height 2. Schedule d entirely on the machines it starts on, w.l.o.g. the bottom two machines. All items of height 1 scheduled in parallel get fully assigned to the top machine and are called immovable. Note that, through this procedure, we can only ever have two immovable items present in any vertical slice. This is because we only call items immovable when they intersect an item with a height of 2. Because the maximum height is 3, only one such item can exist to the right and left of any vertical slice respectively.

We repeat the procedure described above. If there is ever a point where we assign items such that two immovable items would intersect, we swap the order of the item that begins later. Let e be an item of height 2 that is marked immovable in an earlier step of the procedure. It intersects an item d of height 2. Let d be placed at the bottom and e at the top. Further, let g be an item of height 1 that overlaps a different item h with height 2. We want to mark g as immovable and assign it to the top machine, but e and g overlap at some point in the schedule. Because we can only have two immovable items at any point in the schedule, we must be able to swap the order of g and h such that g is assigned to the bottom machine and h to the top two machines. Now, neither item can intersect an immovable item.



■ **Figure 10** An illustration of the transformation from a packed box of tall items to the desired scheduling problem from Lemma 8. We see the sorted original packing on the left, the initially generated schedule in the middle, and the final schedule to the right. Items with bold and capitalized names are those we mark as immovable during the transformation of the schedule.

We repeat this procedure until we reach the end of the schedule. At this point, the schedule has the three desired properties. We then assign each item according to the machines it is scheduled upon. See Figure 10 for an illustration.

If an item is scheduled on the bottom machine, it is assigned to the bottom of the box. If an item is assigned to the top machine but not the bottom, it is assigned to the top of the box. Finally, the remaining items are assigned to the middle of the box.

After extending the height of the box by $1/4H'$, we can place all items according to their assignment without needing to slice them. ◀

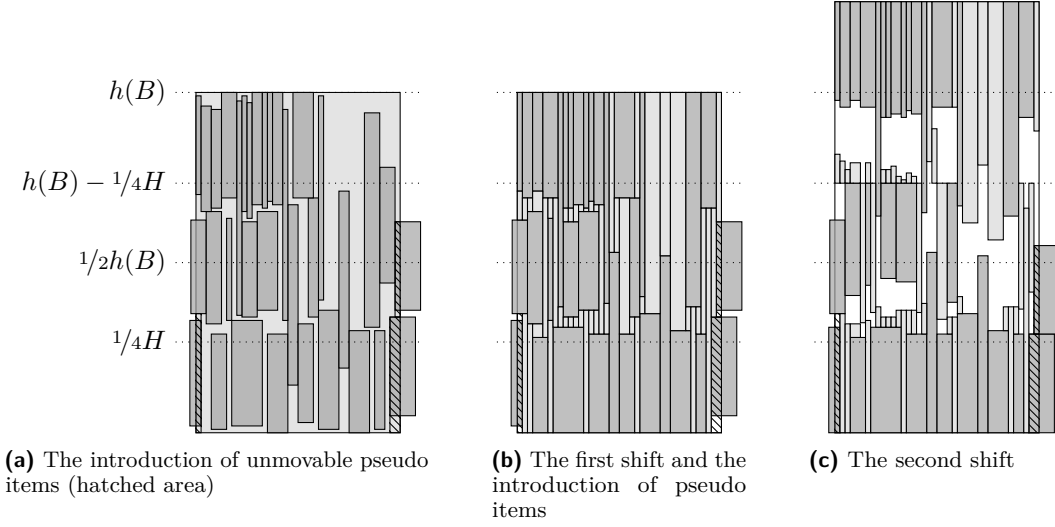
After assigning items to their respective height-lines, we are ready to repack the entire box. To complete this repacking, we utilize an involved routine that separates the box B into up to eighteen different subareas. We solve each of these areas with a unique procedure and obtain a fully structured box as the result. Finally, the proof for the most complex boxes. Note that this proof is very elaborate, readers discretion is advised.

► **Lemma 9.** *Let $B \in \mathcal{B}_{T \cup V}$ be a box with height $h(B) > 3/4H$ such that at most two tall items overlap the left or right box border, while at most three such items overlap any box border. By adding $1/4H$ to B 's height, we can rearrange the items in B such that we generate at most $\mathcal{O}(N^2)$ boxes for tall and $\mathcal{O}(N^2)$ boxes for vertical items without moving the immovable items. The vertical items are sliced and may be separated horizontally while tall items are not separated.*

Proof. Disclaimer: All figures displayed in this proof are adapted from [15] with explicit permission of the authors. This procedure is an evolution of similar ideas in [15]. We present a reordering strategy for items in these boxes. Let $h(B)$ be the height of B . To keep notation simple, we assume that each box has its lower border at 0, which can be achieved through slicing.

Step 1: Shifting the items Let us first consider the tall items overlapping the box border. We slice these items down to the bottom of the box, such that they are placed integrally. Let the one that protrudes further inwards, i.e. the leftmost on the right or the rightmost on the left, be placed below the other item if there are two on a side. Next, we inspect the three horizontal lines inside the box: one at $1/4H$, the second at $1/2h(B)$, and the final one at $h(B) - 1/4H$. Since each tall item has a height of at least $1/4H$, any optimal packing must have these items intersect at least one of these lines. We assign each tall item to at least one of these lines using Lemma 8.

After this assignment, we shift each tall item assigned to the line at $1/4H$ downward to the bottom of the box, as well as parts of all vertical items intersected by this line. Vertical



■ **Figure 11** An overview of the shifting steps. First we define immovable pseudo items; second we shift all the tall items assigned to the horizontal line $1/4H$ and $h(B) - 1/4H$ to the bottom and top and introduce pseudo items; last we extend the box by $1/4H$ such that all tall items touch one of three horizontal lines, and shift the immovable items to the bottom of the packing. These illustrations are taken, and minorly adjusted, from [15].

items where only parts are sliced downwards here might be separated horizontally later. Afterwards, we repeat the same procedure for tall items assigned to the horizontal line at $h(B) - 1/4H$ and vertical items intersected by that line, only placing them such that their upper border touches the top of the box. For each tall item i that is assigned to both of these horizontal lines, i.e. items with height greater than $h(B) - 1/4H$, we create a pseudo item with height $h(B)$ and width $w(i)$.

Next, we extend the box upwards by $1/4H$, and shift all items intersected by the line $h(B) - 1/4H$ up by $1/4H$. All pseudo items between the lines $h(B) - 1/4H$ and $1/2h(B)$ are shifted up so their lower border touches $h(B) - 1/4H$. Afterwards, shift all tall items assigned only to the line at $1/2h(B)$ so that their top touches $h(B) - 1/4H$. Do the same for all pseudo items that were not shifted yet and that are intersected by that line. Through both of these shifts, we simply moved slices inside the optimal packing upwards or downwards. As such, no overlaps can exist at this point.

Step 1.5: Fusing pseudo items As an intermediary step, we need to fuse some pseudo items together. As a result of this, we want to gain the property that all tall and pseudo items have at least one border that touches the horizontal lines at 0 , $h(B) - 1/4H$ or $h(B) + 1/4$. For now, there can still be pseudo items located between $1/4H$ and $1/2h(B)$. However, these items can only exist if there is a tall item touching the top of the packing, and a different tall item touching the bottom of the packing at this width, as they would have been sliced downwards otherwise. Consider two consecutive vertical lines we had drawn to generate the pseudo items. If a tall item overlaps the vertical strip between these lines, its right and left borders lie either on the strips borders or outside of the strip, due to the definition of pseudo items.

Case 1: In the first case, there are three tall items t_1, t_2 and t_3 from bottom to top that

overlap the strip. Since the first two shifting steps have already occurred, t_1 must have its lower border at 0, t_2 its upper border at $h(B) - 1/4H$ and t_3 its upper border at $h(B) + 1/4H$. Therefore, there can be at most two pseudo items. One between t_1 and t_2 , and one between t_2 and t_3 . We stack these pseudo items together such that their lower border begins at $h(B) - 1/4H$. This is possible without overlapping t_3 due to the total height of the pseudo items. Taken together, both pseudo items have a height of at most $h(B) - h(t_1) - h(t_2) - h(t_3)$. The total area not occupied by tall items is $h(B) - h(t_1) - h(t_2) - h(t_3) + 1/4H$ as we have extended the box. The distance between t_1 and t_2 is at most $1/4H$ since they are both tall items and t_1 has its lower border at 0, while t_2 has its upper border at $h(B) - 1/4H$. Thus, the distance between t_2 and t_3 is at least $h(B) - h(t_1) - h(t_2) - h(t_3)$. This is because we added $1/4H$ to the height of the box, but the area between t_1 and t_2 has a lesser height. Therefore, we can fuse and place the pseudo items as desired.

Case 2: Next we consider the case where there is only one tall item t_1 touching the bottom of the box, and one tall item with height at least $1/2h(B)$ touching the top of the box. Since there are two tall items placed above each other, t_2 must have a height of at most $3/4h(B)$. Furthermore, there is at most one pseudo item, and it is placed between $1/4H$ and $1/2h(B)$. We shift this item up until its bottom border touches $1/2h(B)$. This does not construct any overlap because the distance between the top of t_1 and $1/2h(B)$ is less than $1/4H$, which is the distance we extended the box by.

After this, we further fuse each tall item t with height greater than $1/2h(B)$ to all pseudo items placed below the item such that their bottom borders touch $1/2h(B)$. We do this by generating a new pseudo item with height $1/2h(B) + 1/4H$ and width $w(t)$. Thus, all such pseudo items have their top border at $h(B) + 1/4H$ and their lower border at $1/2h(B)$.

Case 3: In the final case, we have two tall items t_1 and t_2 and two pseudo items. One of the tall items t_1 or t_2 touches the top or bottom of the box, while the other ends at $h(B) - 1/4H$. Therefore, the distance between the tall items has to be smaller than $1/4H$. Additionally, one of the pseudo items touches the top or bottom of the box, while the other is positioned between t_1 and t_2 . Since the distance between t_1 and t_2 is less than $1/4H$ one of the distances between the box border and the lower border of t_1 or the top border of t_2 is at least $h(B) - h(t_1) - h(t_2)$. Therefore, we can fuse both pseudo items together. We place this fused pseudo item depending on where the original pseudo item touched the box border. If it was at the top, the fused pseudo item gets placed such that its top touches the top of the box. Otherwise, place the fused pseudo item such that its bottom border touches the bottom of the box.

With the fusing of pseudo items out of the way, our box now has the desired property that all tall or pseudo items touch one of the horizontal lines at 0, $h(B) - 1/4H$ or $h(B) + 1/4H$. Furthermore, we can assume that each item t touching the bottom of the box with $h(t) > 1/2h(B)$ has height $h(B) - 1/4H$: There can be at most two items above t , one tall and one pseudo item. The pseudo item now has its lower border at $h(B) - 1/4H$. Therefore, we can extend t to the horizontal line $h(B) - 1/4H$.

After this shift, each movable item has one border at one of the horizontal lines 0, $h(B) - 1/4H$ or $h(B) + 1/4H$. Additionally, only items with height at least $1/2h(B) + 1/4H$ are crossing the line at $h(B) - 1/4H$.

Step 2: Reordering. For now, assume there is no item with height $h(B)$ in B . We handle the case where there is at least one of those later. In the following we reorder items step by

step, by considering a constant number of smaller subareas of the box. We number these subareas from one to nine. We generate these subareas symmetrically on either side of the box, and call the i -th subarea $B_{r,i}$ or $B_{\ell,i}$ on the right or left respectively. We only describe the steps for the left side in the following. The right side gets handled analogously.

Area $B_{\ell,1}$: Consider the leftmost (pseudo) item i_ℓ with height $1/2h(B)$ in B and let i_r be the rightmost of these items. Left of i_ℓ inside B , there is no item intersecting the horizontal line $h(B) - 1/4H$ since only (pseudo) items with height $1/2h(B) + 1/4H$ touching $h(B) + 1/4H$ overlap this horizontal line. See Figure 12b for an illustration of this area. Therefore, each item left of i_ℓ above $h(B) - 1/4H$ either touches $h(B) - 1/4H$ with its lower border or $h(B) + 1/4H$ with its upper border. Since there is no item intersecting the left box border at this height, we can sort the items left of i_ℓ touching $h(B) + 1/4H$ in descending order and the items touching $h(B) - 1/4H$ in ascending order of heights, without constructing any overlap. We may, however, have separated some vertical items such that they are no longer placed contiguously. These items get handled later on. We do the same for the area right of i_r and call these areas $B_{\ell,1}$ and $B_{r,1}$.

Area $B_{\ell,2}$: We draw a vertical line at the left border of i_ℓ to the bottom of the box. If this line cuts a tall item ℓ_b at the bottom, it defines a new unmovable item. Let us consider the area between the line and the left box border below $h(B) - 1/4H$. We call this area $B_{\ell,2}$. In $B_{\ell,2}$, each (partial) item either touches the horizontal line at 0 or $h(B) - 1/4H$ and on each side, there are at most two tall unmovable items. We extend the unmovable item intersecting $h(B)/2$ to the top, such that it touches the horizontal line at $h(B) - 1/4H$ and reorder this box according to Lemma 7. We do the same for the right side of i_r . The area is illustrated in Figure 12c.

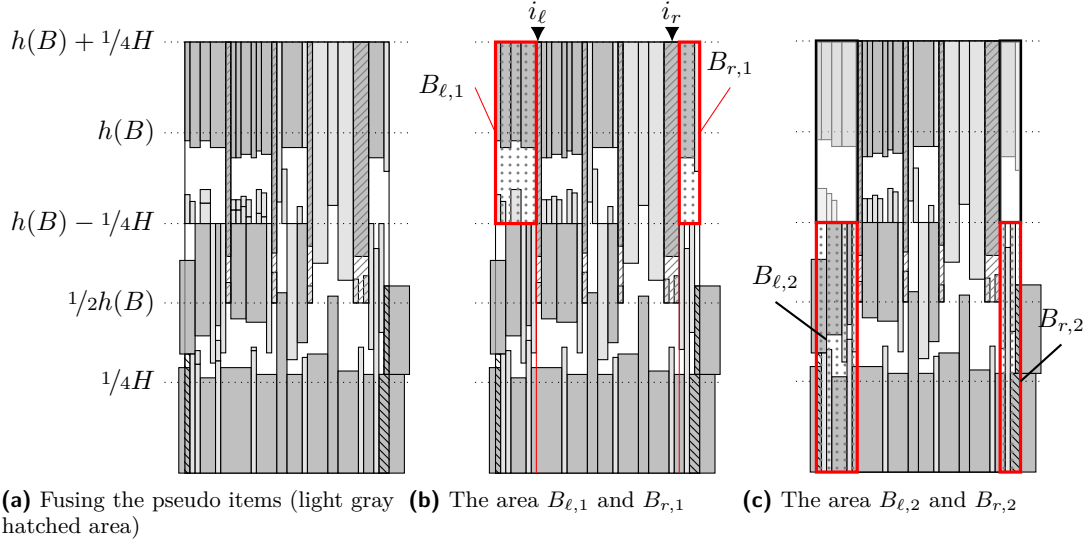
Cases for i_ℓ and i_r : If i_ℓ and i_r do not exist, there are no items overlapping the horizontal line $h(B) - 1/4H$ and we can partition the box in two areas B_1 and B_2 . We reorder B_1 as described for $B_{\ell,1}$ and B_2 as described for $B_{\ell,2}$. In the case that i_ℓ is the same item as i_r , we introduce the areas $B_{\ell,1}$, $B_{\ell,2}$, $B_{r,1}$ and $B_{r,2}$ as described and order the items completely below i_ℓ such that items with the same height are positioned next to each other.

Area $B_{\ell,3}$: Next, we look at the area between the left border of i_ℓ and the right border of i_r . For an example of this area, see Figure 13a. Let $r(i)$ be the right border of an item i . If $r(\ell_b)$ is to the right of $r(i_\ell)$, we draw a vertical line at $r(\ell_b)$, called L_1 . If L_1 intersects a tall item with upper border at $h(B) - 1/4H$, we call this item ℓ_m . Left of L_1 and right of i_ℓ , we shift up each item touching $h(B) - 1/4H$ with its top, including ℓ_m , such that its lower border touches $1/2h(B)$ and shift down each pseudo item touching $h(B) - 1/4H$ with its lower border such that it touches $1/2h(B)$ with its upper border. All pseudo items right of L_1 above ℓ_m are shifted, such that they touch the top of ℓ_m with their bottom. No pseudo item is intersected by the line L_1 .

▷ **Claim 1.** After this shift no item overlaps another item.

Proof. Consider an item i that was shifted up such that its lower border is at $1/2h(B)$. Note that the distance between the upper border of ℓ_b and $1/2h(B)$ is less than $1/4H$ because the upper border of ℓ_b is above $1/4H$ by definition. Hence, there has to be some free space left between the upper border of i and the lower border of each item above since we added $1/4H$ to the packing height.

Now consider an item i' that was shifted down such that its top border is at $1/2h(B)$. Above this item, there has to be a tall item i'' that has its lower border at $1/2h(B)$, which has a height greater than $1/4H$. The item i''' above i'' , an item with its top border at $h(B) + 1/4H$, has a height larger than $1/4H$ as well, since all items with their top borders at $h(B) + 1/4H$ have this property. Therefore, the vertical distance between i'' and i''' is smaller than $1/4H$.



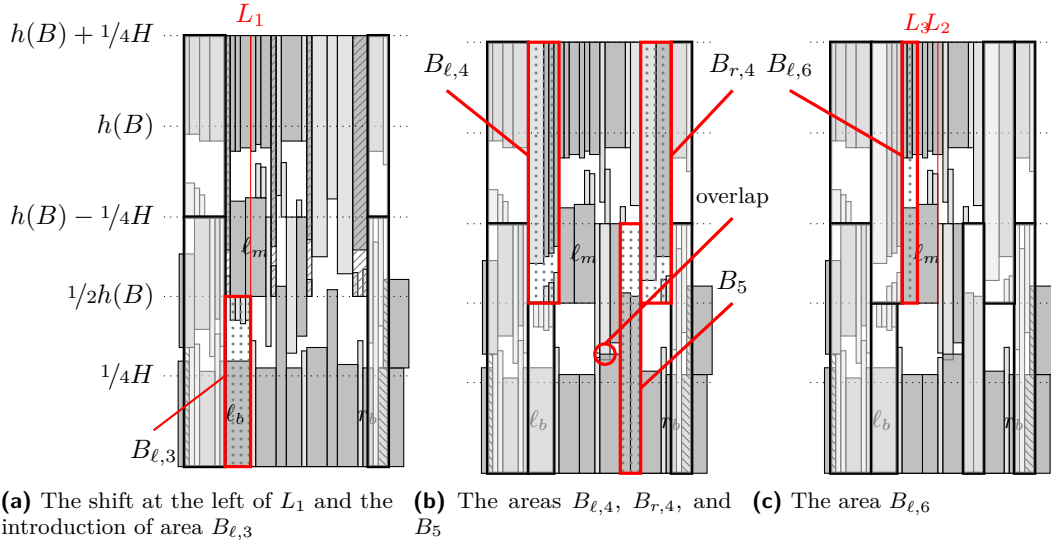
■ **Figure 12** We fuse pseudo items as discussed in Step 1.5. Then we begin defining areas as in Lemma 9. In the middle, we define the areas $B_{\ell,1}$ and $B_{r,1}$, to the right the areas $B_{\ell,2}$, $B_{r,2}$. Each area is packed in a structured manner in the image following its introduction, i.e. $B_{\ell,1}$ is structured in the right image.

Since we have added $1/4H$ to the packing height, the vertical distance between the bottom of i' and the top of ℓ_b has to be larger than zero. This, in turn, means that no item parts can overlap each other vertically. ◀

Let $I_{\ell,1/2h(B)}$ be the set of shifted items now touching $1/2h(B)$ with their lower border. All the items in $I_{\ell,1/2h(B)}$ have a height of at most $1/2h(B)$ because their upper border was at $h(B) - 1/4H$ before. The area left of L_1 and right of the left border of ℓ below $1/2h(B)$ is called $B_{\ell,3}$. This area contains pseudo items touching $1/2h(B)$ and a part of ℓ_b at the bottom. We sort the pseudo items above ℓ_b touching $1/2h(B)$ in descending order of their heights.

On the other hand, if $r(\ell_b)$ is left of $r(i_\ell)$, we introduce the line L_1 but do not shift any items. On the right of i_r , we introduce the same line, just named R_1 , and the area $B_{r,3}$ analogously.

Simple cases: It is possible that ℓ_b and r_b are the same item. Similarly, one of the lines L_1 or R_1 might intersect with i_ℓ or i_r respectively. Finally, L_1 and R_1 might be at the exact same spot. For any of these cases to occur, there cannot be an item with height larger than $1/2h(B)$ touching the bottom of the box between L_1 and R_1 . If $\ell_b = r_b$, then this item cannot be taller than $1/2h(B)$ because i_ℓ and i_r are the left and rightmost items with height larger than $1/2h(B)$. Since they are placed above $\ell_b = r_b$, this item cannot be larger than $1/2h(B)$ as the total would be larger than the box. Similarly, if L_1 intersects i_r , the items ℓ_b and i_r are placed above each other leading to the same contradiction. Obviously, the reverse holds as well. Finally, if L_1 and R_1 are in the same spot the items ℓ_b and r_b must be placed next to each other. Again, neither of these items can have a height greater than $1/2h(B)$, and they extend to the bottom of the items i_ℓ and i_r respectively. When there is no item with height greater than $1/2h(B)$ touching the bottom of the box, we shift all items between L_1 and R_1 touching $h(B) - 1/4H$ with their top such that they touch $1/2h(B)$ with their lower borders. The pseudo items touching $h(B) - 1/4H$ with their lower borders in this area are placed such that their upper border touches $1/2h(B)$, just as we did with the items above ℓ_b . Now,



■ **Figure 13** Illustration of the areas $B_{\ell,3}$ through $B_{\ell,6}$. When reordering the items to create areas $B_{\ell,4}$, $B_{r,4}$, and B_5 , some items may overlap. We resolve this further on, when considering areas $B_{\ell,9}$ and $B_{r,9}$.

there is no item intersecting the horizontal line $1/2h(B)$. Hence, we can sort the items above $1/2h(B)$ between i_ℓ and i_r by their heights in descending order from left to right. We can do the same for items that are below $1/2h(B)$. After this step, no further reordering is required.

Areas $B_{\ell,4}$ and B_5 : Next, we consider the case that there is an item with height taller than $1/2h(B)$ touching the bottom of the box between i_ℓ and i_r . This case requires further reordering. The aim is to reorder the (pseudo) items with height $1/2h(B) + 1/4H$ touching the top of the box in a way that they build two blocks next to i_ℓ and i_r respectively. These blocks will be the areas $B_{\ell,4}$ and $B_{r,4}$, see Figure 13b. To facilitate this reordering, we have to define a border between i_ℓ and i_r such that all of the items with height $1/2h(B) + 1/4H$ are shifted to i_ℓ on the left and i_r on the right. As we know one such item must exist, let i be an item of height larger than $1/2h(B)$ touching the bottom of the packing. Clearly, no items of height $1/2h(B) + 1/4H$ can be above i , even in the box we extended by $1/4H$. This item defines the border between i_ℓ and i_r .

Consider items with height $1/2h(B) + 1/4H$ touching the top of the box between i_ℓ and i . We shift those items left of i to the left until they touch i_ℓ . We shift the items between i and i_r analogously, but to the right. All other items with parts above $1/2h(B)$ are shifted to the left and right accordingly to avoid overlap. This must be possible because these items could not be placed below the items of height $1/2h(B) + 1/4H$ originally, and the total available width left and right of i has not changed. We sort the items with height $1/2h(B) + 1/4H$ such that the pseudo items that include tall items of equal height are placed adjacently. The area including these items to the left of i , including i_ℓ , is called $B_{\ell,4}$.

While doing the shift discussed above, we shift all items between i_ℓ and i_r with height $h(B) - 1/4H$ touching the bottom of the box such that they are next to i and shift the other items to the left or right accordingly. This is feasible for the same reasons as above. This comprises the area B_5 , which can also be seen in Figure 13b.

However, as an interplay between the areas $B_{\ell,4}$ and B_5 , it may occur that items touching $h(B) - 1/4H$ are overlapping with items that touch the bottom of the box. This overlap gets handled later on, in the areas $B_{\ell,9}$ and $B_{r,9}$ respectively.

Area $B_{\ell,6}$: Note that items in the set $I_{\ell,1/2h(B)}$ are now placed next to each other. Before the last step it was possible that items with $1/2h(B) + 1/4H$ were positioned between them. Additionally, there is no item part touching the top of the box above an item that touches $h(B) - 1/4H$ with its bottom border. Furthermore, the total width of items with their bottom border between $1/4H$ and $1/2h(B)$, that are placed entirely between L_1 and the right border of i did not change.

If ℓ_m exists, we draw the vertical line L_2 at the right border of ℓ_m and a vertical line L_3 at the left border of ℓ_m . Define $\ell_{t,r}$ and $\ell_{t,\ell}$ to be the tall items touching the top of the border intersected by these lines if there are any. We look at the left of L_3 and right of $B_{\ell,4}$, which is bounded at the top by the top of the box, while the bottom border is at $1/2h(B)$. We call this area $B_{\ell,6}$. In this area each item touches the bottom or the top and there is at most one item $\ell_{t,\ell}$ intersecting the border. We use the reordering from Lemma 7 to reorder the items in $B_{\ell,6}$. See this area in Figure 13c.

Area $B_{\ell,7}$: The area above the item ℓ_m is called $B_{\ell,7}$, find an illustration of it in Figure 14a. In this area, all items are touching either the top of the box or ℓ_m . All items that touch only ℓ_m with their bottom are pseudo items. We order the items touching the top of the box in ascending order regarding their heights and move those pseudo items below with them. Next, we look at the overlapping items $\ell_{t,r}$ and $\ell_{t,\ell}$. We move items with their heights next to the items themselves, i.e. items with height $h(\ell_{t,r})$ next to $\ell_{t,r}$ and the same for $\ell_{t,\ell}$. This results in three areas to place pseudo items in. The first two areas are below the respective overlapping items $\ell_{t,r}$, $\ell_{t,\ell}$ and their accompanying items with identical height, while the last is between these two areas. For all those areas, we sort the pseudo items in them by order of descending height.

The areas $B_{\ell,6}$ and $B_{\ell,7}$ only exist if the item ℓ_m exists. If this is not the case, we introduce the vertical line L_2 at the left border of the area $B_{\ell,4}$. The same holds for the areas $B_{r,6}$, $B_{r,7}$ and the vertical line R_2 , which will be placed at the right border of $B_{r,4}$.

Area B_8 : Next, we inspect the area above $h(B) - 1/4H$ and between the lines L_2 and R_2 . This area is called B_8 , see Figure 14b. There are at most two immovable items overlapping this area. One item $\ell_{t,r}$ on the left touching the top of the box, and a similar item $r_{t,\ell}$ on the right, also touching the top of the box. Since B_8 does not contain any item of height $1/2h(B) + 1/4H$ or items from the sets $I_{\ell,1/2h(B)}$ or $I_{r,1/2h(B)}$, each item either touches the top or bottom of this area. Furthermore, all items touching the bottom are pseudo items. Therefore, we can sort this area in the same manner we did $B_{\ell,7}$.

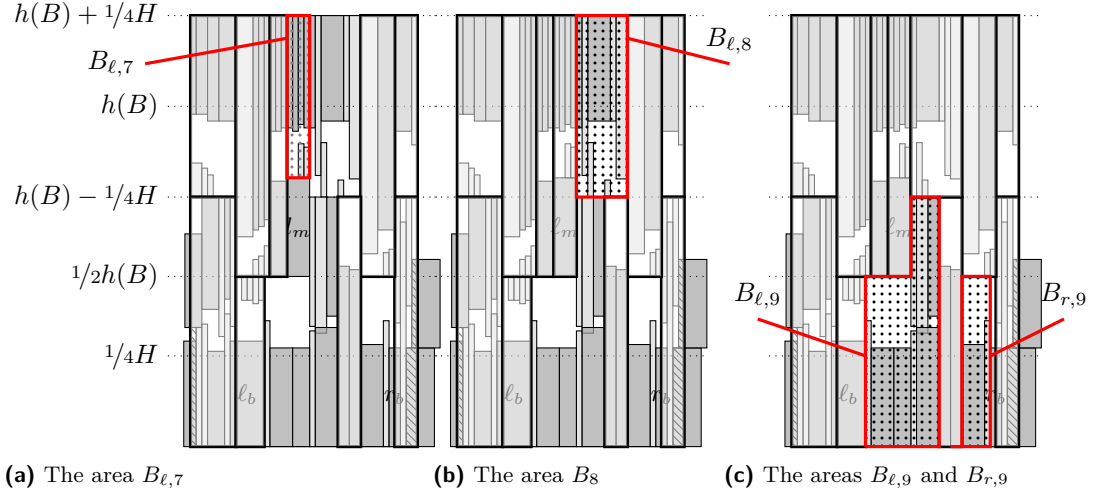
Area $B_{\ell,9}$: Finally, we have to look at the items on the bottom between L_1 and R_1 , as well as the items touching $h(B) - 1/4H$ with their top between L_2 and R_2 . We consider the items touching the bottom between L_1 and the left border of B_5 and the items touching $h(B) - 1/4H$ with their top between L_2 and the left border of B_5 . The area containing these items is called $B_{\ell,9}$. In this area, we sort all items touching $h(B) - 1/4H$ in ascending order of their heights and the items at the bottom in descending order of their heights, from outside inward. We do the same for the right side in the area $B_{r,9}$. This procedure is shown in Figure 14c.

▷ **Claim 2.** After this step, there is no item that overlaps another item in the area $B_{\ell,9}$.

Proof. First of all, let us note that all items that have their bottom border on the bottom of the box have a height of less than $1/2h(B)$. Otherwise, they would be contained in the area B_5 . Thus, they cannot overlap with items that have their upper border on the top of the box, as those have a maximum height of $1/2h(B) + 1/4H$.

Let us now assume the claim is false, i.e. there is an item b from the bottom of the box

intersecting an item t from the top of the box. Let this intersection be at a point (x, y) in the area $B_{\ell,9}$. Due to the manner in which we sorted the items touching either box border, all items touching $h(B) - 1/4H$ left of (x, y) and inside $B_{\ell,9}$ must also overlap the horizontal line at height y . Analogously, all items to the right of (x, y) and the left of L_1 must also overlap this horizontal line. Note that the total width of items with lower border below y and above $1/4H$ between L_1 and the left border of i has not changed after the shifting of items with height $1/2h(B) + 1/4H$ on the top of the box. This is because items left of ℓ_m have their lower border at $1/2h(B)$. Additionally, the total width of items touching the bottom of the box with upper border above y in this area has not changed either. Therefore, the total width of items overlapping the horizontal line at y in this area is larger than the width of this area. Thus, the items must have had some overlap before the first horizontal shift. This is a contradiction because we assumed a feasible optimal packing. Therefore, there cannot be an item overlapping another item inside $B_{\ell,9}$. ◀



■ **Figure 14** The definition of the final areas, and the reordering of items inside them.

Items with height $h(B)$: Finally, we consider the case that there are (pseudo) items with height $h(B)$ inside the box. This can be handled in a very simple manner. We select one of these items, and shift it all the way to right of the box. Next, we shift all other items with height $h(B)$ right next to this item. Since these items completely overlapped the box vertically due to their height, shifting them to one border of the packing results in the remainder still being feasible, after shifting them over correspondingly. Afterwards, we can continue with the procedure described above, effectively treating the box as being smaller by the width of the items with height $h(B)$.

Analyzing the number of constructed boxes. In the worst case, we have (pseudo) items with height $h(B)$ and both ℓ and r exist. Furthermore, the left border of ℓ_b should be right of the left border of ℓ , as well as the left border of r_b being left of r . For repacking purposes, we did not require the assumption we got from our rounding, that tall items are placed on an arithmetic grid. To analyze the the number of boxes, however, it is convenient to use this assumption. To begin, we analyze the number of boxes for tall items we generate.

▷ **Claim 3.** The number of boxes for tall items is bounded by $2N^2 + (14N)/4 + 8$, where H/N is the distance between the grid lines.

Proof. We show that the above is true by iteratively inspecting the generated areas and counting the number of generated boxes inside each of them. In the following, whenever we name a area, we only refer to the left area, but the arguments hold for both the left and right areas, as they are analogous.

Area $B_{\ell,1}$: $N/4$ boxes. In these areas, there are tall items with heights between $1/4H$ and $1/2H$ at the top of the box. For each of these sizes, we generate at most one box in each area. Therefore, both contain at most $N/4$ boxes for tall items.

Area $B_{\ell,2}$: $N^2/2 + N/4 + 3$ boxes. These areas contain at most one box for each item height larger than $1/2h(B)$ and smaller than $1/3h(B)$. These are at most $N/4$ sizes. For the remaining sizes, we know by Lemma 7, that we create at most $4S_T S_{P \cup T} + 3$ boxes in total, since there are at most three immovable items overlapping this area. We know that $S_T \leq N/4$ since the tall items have heights between $1/4H$ and $1/2h(B)$, $S_P \leq N/2$ since they have heights smaller than $1/2h(B)$. Finally, because both the tall and pseudo items in this area have a height of at most $1/2h(B)$, we have $S_{P \cup T} \leq N/2$ as well. This leaves us with at most $4N/4 \cdot N/2 + N/4 + 3 = N^2/2 + N/4 + 3$ boxes for the tall items in these areas.

Area $B_{\ell,3}$: 0 boxes. The only tall items inside these areas are ℓ_b and r_b respectively. As they overlap the areas $B_{\ell,2}$, they were counted already.

Area $B_{\ell,4}$: $N/4$ boxes. These areas contain only tall items with heights between $1/2h(B)$ and $1/2h(B) + 1/4H$. We simply create one box for each of these sizes, leaving us with $1/4N$ boxes.

Area $B_{\ell,5}$: $N/4$ boxes. Similarly to the area $B_{\ell,4}$, these areas only contain tall items with heights between $1/2h(B)$ and $1/2h(B) + 1/4H$, again leaving us with $1/4N$ boxes.

Area $B_{\ell,6}$: $N^2/2 + 1$ boxes. In these areas, there are only tall and pseudo items with height less than $1/2h(B)$. As we solve this area with Lemma 7, we create at most $N^2/2$ boxes for tall items in each of the areas. Further, there might be an overlapping tall item, requiring another single box, resulting in at most $N^2/2 + 1$ total boxes.

Area $B_{\ell,7}$: $N/4$ boxes. These areas contain ℓ_m and r_m respectively. Above these items we create at most $N/4$ boxes for tall items each, since the tall items have heights between $1/4H$ and $1/2h(B)$. The box for the item overlapping L_3 is already counted.

Area B_8 : $N/4$ boxes. In this area the tall items have a height of at most $1/2h(B)$ and we create one box per item size. Thus, this leaves us with a total of $N/4$ boxes for B_8 . Any items overlapping the lines L_2 or R_2 are already counted in the area $B_{\ell,7}$.

Area $B_{\ell,9}$: $2N/4$ boxes. Next, we consider these areas. Here, all items have a height of at most $1/2h(B)$ and we create a box for each size at both the bottom and top of the areas. This leaves us with $2N/4$ boxes for each of the areas.

Area B_{10} : $N/4$ boxes. This area contains pseudo items with height $h(B)$. Here, we create at most one box for each item with height larger than $h(B) - 1/4H$ resulting in at most $N/4$ boxes.

After doubling the count of boxes for all duplicate areas, this leaves us with at most

$$\begin{aligned} & 2(N/4 + N^2/2 + N/4 + 3 + N/4 + N^2/2 + 1 + N/4 + 2N/4) \\ & + 2N/4 \\ & = 2N^2 + 14N/4 + 8 \end{aligned}$$

boxes, proving the claim. ◀

Next we consider the boxes for vertical items.

▷ **Claim 4.** The number of boxes for vertical items is at most $4N^2 + 41N/4 + 5$.

Proof. Similarly to the last claim, we prove this by considering the areas iteratively. In the following, whenever we name a area, we only refer to the left area, but the arguments hold for both the left and right areas, as they are analogous.

Area $B_{\ell,1}$: $N/2$ boxes. In these areas, items touching the bottom have a height of at most $1/4H$. The same holds true for items touching the top, as they are vertical. This yields at most $2N/4 = N/2$ boxes in each of these areas.

Area $B_{\ell,2}$: $N^2 + 2$ boxes. In these areas, the pseudo items touching the bottom have sizes between $1/4H$ and $1/2h(B)$ and the items touching the top have sizes up to $1/2h(B)$. Using Lemma 7, we generate at most $4S_P S_{P \cup T} \leq 4N/2 \cdot N/2 = N^2$ boxes plus the two boxes for extending the immovable items in each area. Thus, we create at most $N^2 + 2$ boxes in each area.

Area $B_{\ell,3}$: $N/4$ boxes. These areas contain pseudo items with heights up to $1/4H$. For each of these sizes, we generate at most one box, resulting in $N/4$ boxes for each of these areas.

Area $B_{\ell,4}$: $N/2$ boxes. Here, below the items with height larger than $1/2h(B)$, we packed items with height at most $1/4H$. We have two blocks of these items, one each at ℓ and r . Thus, we create $N/4$ boxes in these areas, resulting in a total of $N/2$ boxes for these areas.

Area B_5 : $N/4$ boxes. This area does not contain pseudo items above its tall items with height between $1/2h(B)$ and $1/2h(B) - 1/4H$, as they were shifted up so their lower border is at $h(B) - 1/4H$. Therefore, this area only contains pseudo items with height between $1/2h(B)$ and $1/2h(B)$, and we create at most one box for each size, yielding $N/4$ boxes.

Area $B_{\ell,6}$: N^2 boxes. These areas contain pseudo items with heights between $1/4H$ and $1/2h(B)$ at the top and items with heights up to $1/2h(B)$ at the bottom. Therefore, by Lemma 7, we generate at most N^2 boxes, analogously to the area $B_{\ell,2}$. We do not create the additional pseudo item here, as $\ell_{t,\ell}$ already touches the top of the area. Thus, the number of boxes is at most N^2 .

Area $B_{\ell,7}$: N boxes. In these areas, we have at most three areas for pseudo items touching ℓ_m with their lower border. These items have a height of at most $1/4H$. Therefore, we create at most $3 \cdot 1/4N$ boxes for these items. Finally, the items touching the $h(B) + 1/4H$ with their top have a height between $1/4h(B)$ and $1/2h(B)$. Again, we generate one box for each height, resulting in at most $1/4N$ boxes. This leaves us with at most N boxes in each area.

Area B_8 : $N/2$ boxes. In this area, we have (pseudo) items with height up to $1/4H$ touching the bottom and pseudo items with sizes between $1/4H$ and $1/2h(B)$ touching the top. For each of these sizes, we generate at most one box, yielding $N/2$ boxes.

Area $B_{\ell,9}$: N boxes. In these areas, the pseudo items have height between $1/4H$ and $3/4 \cdot h(B)$. We create at most one box for each of these sizes, yielding $N/2$ boxes. At the top of the area, we have items with height at most $1/2h(B)$, again generating one box per size. This results in another $N/2$ boxes, totalling at N boxes for the area.

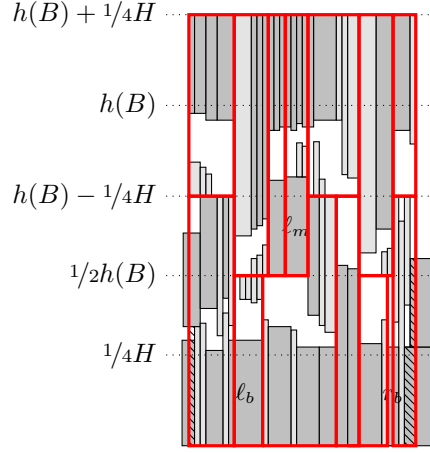
Area B_{10} : $N/4 + 1$ boxes. Finally, we consider the items with height larger than $3h(B)/4$ in the area B_{10} . There can be pseudo items with height at most $1/4H$ above these items. Since we create at most one box per size, we create at most $N/4$ boxes. Further, there may be exactly one box for pseudo items with height exactly $h(B)$.

Summing the generated boxes up and considering duplicated areas, we create at most

$$\begin{aligned} & 2(N/2 + N^2 + 2 + N/4 + N/2 + N^2 + N + N/2 + N + N/4) \\ & + N/4 + 1 \\ & = 4N^2 + 41N/4 + 5 \end{aligned}$$

boxes for vertical items.

This concludes the proof of Lemma 9. For an illustration of the resulting box, see Figure 15.



■ **Figure 15** The packing that results from the procedure given in Lemma 9. All sub-areas are highlighted individually and sorted as discussed above.

Step 5

Having shown that we can separate the optimal packing into $\mathcal{O}_\varepsilon(1)$ many boxes that each contain only a single item or item type, we now show that we can fill these boxes with their respective items.

The boxes for horizontal and vertical items are both filled using configuration IPs. Vertical items might have been treated as pseudo items and have been separated. During their placement, we ensure that they are placed integrally again. The configuration IPs for both types function in a similar manner and ensure that all items are placed integrally while not exceeding the border of the strips. First, consider the placement of vertical items.

► **Lemma 10.** *Let H_V be the set of different heights of vertical items and μW the maximal width of a vertical item. Furthermore, let \mathcal{B}_P be the set of boxes containing all separated vertical items and only them. There exists a non separated placement of vertical items into the boxes \mathcal{B}_P and at most $7(|H_V| + |\mathcal{B}_P|)$ additional boxes \mathcal{B}'_P , each of height at most $1/4H$ and width μW , such that the boxes $\mathcal{B}_P \cup \mathcal{B}'_P$ are partitioned into at most $\mathcal{O}((|H_V| + |\mathcal{B}_P|)/\delta)$ sub-boxes \mathcal{B}_V , containing only vertical items of the same height and at most $\mathcal{O}(|H_V| + |\mathcal{B}_P|)$ empty boxes \mathcal{B}_V^S with total area $a(\mathcal{B}_V^S) \geq a(\mathcal{B}_P) - a(\mathcal{V})$.*

Proof. This procedure is an evolution of the ideas in [15]. We prove this lemma via a configuration LP. We define a configuration as a multiset of items that can be placed atop each other without exceeding the boundaries of a given box B . We then place these configurations of items inside their assigned boxes. Therefore, let $C = \{a_h : h \in H_V\}$. The height of a configurations is the sum of all item-heights inside the configuration, thus $h(C) := \sum_{h \in H_V} h \cdot a_h$. Let \mathcal{C}_B be the set of configurations with height of at most $h(B)$, i.e. configurations that fit inside this box. We use $X_{C,B}$ to indicate how many configurations C are assigned to a box B and define $w(C)$ as the width of a configuration. Finally, we define

w_h to be the total width of all vertical items with height h for all $h \in H_V$.

The configuration LP is defined as follows:

$$\begin{aligned} \sum_{C \in \mathcal{C}_B} X_{C,B} w(C) &= w(B) & \forall B \in \mathcal{B}_P \\ \sum_{B \in \mathcal{B}_V} \sum_{C \in \mathcal{C}_B} X_{C,B} \cdot a_{h,C} &= w_h & \forall h \in H_V \\ X_{C,B} &\geq 0 & \forall B \in \mathcal{B}_V, C \in \mathcal{C}_B \end{aligned}$$

The first equation ensures that all items inside a configuration fit inside their assigned boxes regarding their width. We later fill these configurations greedily and handle the resulting overlap through the additional boxes. The second equation ensures that we place all items of a given height across all our configurations. Doing this for every height ensures that all items are placed after solving the configuration LP. Finally, the last equation states that all configurations can only contain positive entries to ensure correctness, as there cannot be negative items that free up space.

As the configuration LP is a linear program in itself, it has a basic solution with at most $|H_V| + |\mathcal{B}_P|$ non-zero components. This is because it has that many conditions.

Given such a basic solution, we place the selected configurations inside their corresponding boxes. Afterwards, we place the items into the configurations, such that the last item overlaps the configuration border. Each configuration has a height of at most H since the boxes \mathcal{B}_P have at most that height.

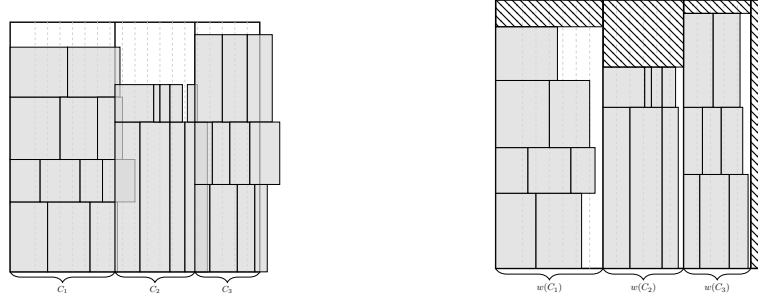
We partition the set of overlapping items in each configuration into seven boxes with height $1/4H$ and width μW in the following way: First, we stack the items in four boxes one by one atop each other such that the last item overlaps the box on top. Since the total height of the items is at most H , there are at most three overlapping items. Each item is placed into its own box. These boxes are called \mathcal{B}'_P . In total, we generate at most $7(|H_V| + |\mathcal{B}_P|)$ boxes of width μW . The items can be placed wholly inside these boxes as they each have a width of at most μW , since they are vertical items.

It is worth noting that the configuration width as defined by the basic solution of the configuration LP might not be integral. Since we removed all overlapping items, however, we only need an integral width, and can therefore reduce the configuration width to the next smaller integer. This may result in empty configurations inside the strip. These empty configurations have at least a width of the sum of all non integral fractions we removed from the configurations inside their box. Since their boxes had an integral width and all other configurations have an integral width as well, these empty configurations have integral widths themselves.

Since configurations have a height of at most H and each item has a height of at least δOPT , each configuration contains at most $H/(\delta \text{OPT}) \in \mathcal{O}(1/\delta)$ items. Therefore, the set of boxes $\mathcal{B}_P \cup \mathcal{B}'_P$ is divided into at most $2(|H_V| + |\mathcal{B}_P|)H/(\delta \text{OPT}) \in \mathcal{O}((|H_V| + |\mathcal{B}_P|)/\delta)$ sub-boxes containing only vertical items of the same height.

Consider a configuration $C \in \mathcal{C}_B$ which has a non-zero entry $X_{C,B}$ in the considered solution. Above this configuration, there is a free area of height $h(B) - h(C)$ and width $X_{C,B}$ inside the box B , see Figure 16. Furthermore, in each box there might be a new empty configuration which generates an empty box as well. Let \mathcal{B}_V^S be the set of these boxes. There are at most $\mathcal{O}(|H_V| + |\mathcal{B}_P|)$, one for each configuration and one extra for each box. Since we ensured that the configurations use exactly the width of the vertical items, the total area of these empty boxes must be at least $a(\mathcal{B}_V^S) \geq a(\mathcal{B}_P) - a(\mathcal{V})$. ◀

We place the horizontal items using a similar configuration IP.



■ **Figure 16** An illustration of a set of configurations assigned to a box B . The width of a configuration is shown by the thick black lines. The configurations are filled greedily until each height in it exceeds the width of the configuration. After removing these excess items, we can reduce the width of configurations to the next integral value. The area not used by items is shown as hatched and can later be used to pack small items.

► **Lemma 11.** *There is an algorithm with running time $(\log(1/\delta)/\varepsilon)^{\mathcal{O}(1/\varepsilon\delta^3)}$ that places the horizontal items into the boxes \mathcal{B}_H and an extra box B_H of height at most $\varepsilon^9 \text{OPT}$ and width W . Furthermore, the algorithm creates at most $\mathcal{O}(1/\varepsilon\delta^2)$ empty boxes \mathcal{B}_H^S with total area $a(\mathcal{B}_H^S) = a(\mathcal{B}_H) - a(\tilde{H})$, where \tilde{H} is the set of horizontal items that overlap the top or bottom box borders.*

Proof. This algorithm is an evolution of similar ideas in [15]. We begin by rounding the input, in this case the horizontal items. To achieve this, we stack horizontal items on top of one another ordered by descending width, i.e. widest at the bottom. This stack has a height of at most OPT/δ , because each horizontal item has a width of at least δW and the total packing area has size $\text{OPT} \cdot W$. We group the items in the stack to at most $1/\varepsilon\delta^2$ groups, each of height $\varepsilon\delta^2 \text{OPT}/\delta = \varepsilon\delta \text{OPT}$ and round the items in the groups to the widest width occurring in this group. This step reduces the number of different sizes to at most $1/\delta\varepsilon^2$, i.e. one per generated group. The rounded horizontal items can be placed fractionally into the non-rounded items of the group containing the next larger items. The group containing the widest rounded items has to be placed atop the final packing. As this is just one group, this extra box has a height of at most $\varepsilon\delta \text{OPT}$. We define an extra box of width W and that height to pack these items into. To keep notations simple we assume that \mathcal{B}_H includes this box as well.

Similar to our procedure to place the separated vertical items in Lemma 10, we place the rounded horizontal items into their boxes through the use of a configuration LP. Here, a configuration is a set of items that fit next to each other inside the boxes. Thus, a configuration C is a multiset of the form $\{a_w : w|w \in \mathcal{W}_H\}$ where \mathcal{W}_H is the set of all widths of horizontal items. The width of such a configuration is consequently defined as $w(C) := \sum_{w \in \mathcal{W}_H} a_w w$, i.e. simply lining all items inside a configuration up beside one another. Additionally, \mathcal{C}_w denotes the set of configurations with width at most w . We use $X_{C,B}$ to indicate the number of times configuration C is assigned to box B . The height of a configuration is denoted by $h(C)$. Finally, $h(w)$ is defined as the total height of all items with rounded width w .

The set of configurations \mathcal{C}_W is bounded by $\mathcal{O}((\log(1/\delta)/\varepsilon)^{1/\delta})$ because the items have a width of at least δW . Therefore, there can be at most $1/\delta$ items inside each configuration. The following configuration LP is solvable since the rounded horizontal items fit fractionally

into the boxes \mathcal{B}_H .

$$\begin{aligned} \sum_{C \in \mathcal{C}_w(B)} X_{C,B} h(C) &= h(B) & \forall B \in \mathcal{B}_H \\ \sum_{B \in \mathcal{B}_H} \sum_{C \in \mathcal{C}_w(B)} X_{C,B} a_{w,C} &= h(w) & \forall w \in \mathcal{W}_H \\ X_{C,B} &\geq 0 & \forall B \in \mathcal{B}_H, C \in \mathcal{C}_w(B) \end{aligned}$$

The first constraint ensures that all boxes are filled by configurations to exactly their height. We fill these configurations greedily in the following. The second constraint ensures that all items of every height are placed inside some configuration, i.e. that we successfully place all items. Finally, the third constraint ensures the feasibility of the packing by disallowing negative configurations to compensate for space, as such configurations are impossible.

We can solve this linear program by guessing the at most $|\mathcal{W}_H| + |\mathcal{B}_H| = \mathcal{O}(1/(\varepsilon\delta^2))$ non-zero entries of the basic solution and solve the resulting equality system using the Gauß-Jordan-Elimination. We use the first found solution where no variables are negative. Such a solution can be found in at most $\mathcal{O}(|C_W|^{|\mathcal{W}_H|+|\mathcal{B}_H|} \cdot (|\mathcal{W}_H| + |\mathcal{B}_H|)^3 \leq (\log(1/\delta)/\varepsilon)^{\mathcal{O}(1/\varepsilon\delta^3)})$ operations since the configuration LP has to be solvable for the correct partition.

We place each configuration into the corresponding box and place the original horizontal items inside these configurations greedily. We allow the last item to overlap the configuration border. One by one, we place the original items inside an area reserved by the configurations for their rounded equivalents until an item overlaps this area on top. We repeat this process for the next area. Since the total width of these parts is exactly as large as the total width of items with that rounded width, there are enough parts to place all of them. Recall that we ensured this with the second constraint.

Next, we handle the items that overlap the box borders. We place them on top of the box. Each of these removed items is horizontal, and as such has a height of at most μOPT . We add at most $\varepsilon^{10}\text{OPT}$ to the packing height by shifting the overlapping items to the top of the packing. This is due to a basic solution having at most $\mathcal{O}(1/(\varepsilon\delta^2))$ configurations. All items inside a given configuration can be placed next to each other per definition, meaning we can add at most one overlapping item for every configuration. Finally, we selected μ such that $\mu \leq \delta^2\varepsilon^{11}/k$ holds for a suitably large constant $k \in \mathbb{N}$. Taking these items together with the extra box we require due to rounding the items, the total height is bounded by $\varepsilon^{10}\text{OPT} + \delta\varepsilon\text{OPT} \leq \varepsilon^9\text{OPT}$.

Similar to the steps taken in Lemma 10 we can reduce the height of each configuration to the next smaller integer since all horizontal items have integral height. This introduces at most one new configuration per box, i.e. the empty one. Inside each box B , to the right of every used configuration C , there might be some unused space of width $w(B) - w(C)$ and height $x_{C,B}$. This area defines one of the empty boxes \mathcal{B}_H^S . We have at most $|\mathcal{W}_H| + |\mathcal{B}_H|$ configurations and at most $|\mathcal{B}_H|$ boxes for horizontal items. We introduce at most one box for every configuration in any box. Therefore, we introduce at most $\mathcal{O}(1/\varepsilon\delta^2)$ empty boxes \mathcal{B}_H^S . Their total area has to be at least as large as the empty space left inside the boxes, i.e. $a(\mathcal{B}_H^S) = a(\mathcal{B}_H) - a(\tilde{H})$, since the configurations contain exactly the total area of the rounded horizontal items.

Finally, let us consider the sub-boxes for horizontal items we generate in this step for every box B in \mathcal{B}_H . Each configuration contains at most $\mathcal{O}(1/\delta)$ positions for items, as each horizontal item has a width of at least δW . We generate one sub-box that has rounded width of the item assigned to this position by the configuration. The height of this sub-box is the sum of all heights of the items with that width positioned inside this box. We can combine

the height of these sub-boxes because there might be several configurations assigned to B that all contain an item of the same rounded width at the same position. Finally, we create an additional box for each shifted item. Thus, we introduce at most $\mathcal{O}(1/(\varepsilon\delta^3))$ boxes for horizontal items. Each of these boxes only contain items of the same rounded width. \blacktriangleleft

Next, we show that any optimal packing can be rearranged and partitioned into $\mathcal{O}_\varepsilon(1)$ many boxes that are structured in some way. We prove this using the partition given in Lemma 5 into sets of boxes $\mathcal{B}_L, \mathcal{B}_H$, and $\mathcal{B}_{T \cup V}$. We then utilize the reordering techniques discussed in step 4 to further partition all boxes $B \in \mathcal{B}_{T \cup V}$ into boxes for tall items \mathcal{B}_T and boxes for vertical items \mathcal{B}_V . We show that we provide an adequate amount of space to pack all vertical items that were separated during the repacking of $\mathcal{B}_{T \cup V}$.

► **Lemma 12.** *By extending the packing height to $(5/4 + 5\varepsilon)\text{OPT}$ each rounded optimal packing can be rearranged and partitioned into $\mathcal{O}(1/(\delta^3\varepsilon^5))$ boxes with the following properties:*

- *there are $|\mathcal{L}| + |\mathcal{M}_v| = \mathcal{O}(1/(\delta^2\varepsilon))$ boxes \mathcal{B}_L each containing exactly one item from the set $\mathcal{L} \cup \mathcal{M}_v$ and all items from this set are contained in these boxes,*
- *there are at most $\mathcal{O}(1/(\delta^2\varepsilon))$ boxes \mathcal{B}_H containing all horizontal items with $\mathcal{B}_H \cap \mathcal{B}_L = \emptyset$. The horizontal items can overlap the top and bottom border of the boxes, but never the left or right border,*
- *there are at most $\mathcal{O}(1/(\delta^2\varepsilon^5))$ boxes \mathcal{B}_T containing tall items, such that each tall item t is placed inside a box with rounded height $h(t)$,*
- *there are at most $\mathcal{O}(1/(\delta^3\varepsilon^5))$ boxes \mathcal{B}_V containing vertical items, such that each vertical item v is placed inside a box with rounded height $h(v)$,*
- *there are at most $\mathcal{O}(1/(\delta^2\varepsilon^5))$ boxes \mathcal{B}_S for small items, such that the total area of these boxes combined with the total free area inside the horizontal boxes and vertical boxes is at least as large as the total area of small items,*
- *the upper and lower border of each box is placed at a multiple of $\varepsilon\delta\text{OPT}$.*

Proof. This procedure is an adaptation of similar procedures in [15]. Before we get to the proof itself, we give a short overview of the process to come.

Recall that the process begins with the partition given in Lemma 5 into the sets of boxes $\mathcal{B}_L, \mathcal{B}_H$ and $\mathcal{B}_{T \cup V}$. We define the height of the strip $H := (1 + 2\varepsilon)\text{OPT}$. Therefore, each tall item has a height larger than $H/4$. This allows us to use our reordering techniques explained in the previous section to separate $\mathcal{B}_{T \cup V}$ into \mathcal{B}_T and \mathcal{B}_V . We have seen in Lemma 10 that we require some extra boxes to place the vertical items. Thus, the main thing to be proven here is that there is adequate space left in the packing to place these boxes.

We consider three options to place them. The first choice depends on the widest tall items that intersects the line at $1/2H$. We fix their position and attempt to place these extra boxes atop these tall items if their width is large enough. If this is not possible, we know that the tall items intersecting $1/2H$ are not very wide. This allows us to attempt to place the extra boxes inside the boxes B of height $h(B) > 3/4H$. Again, the feasibility of this placement depends on the total width of these boxes. Finally, if all else fails, we place them on top of boxes with height between $1/2H$ and $3/4H$.

The second part of this proof is to show that tall and medium boxes have to start and end at grid lines. Since we already know from our rounding, Lemma 3, that tall items start and end at multiples of $\varepsilon^2\text{OPT}$, we can choose these lines as the grid. Setting the start- and endpoints of the tall and medium boxes to these lines leads to a negligible loss in approximation ratio as we show in the following.

After placing the boxes on the grid as such, we have fulfilled the requirements to utilize the reordering techniques from Lemma 6, Lemma 7 and Lemma 9 to reorder items inside

the boxes $\mathcal{B}_{T \cup V}$. After this reordering, we analyze the number of containers constructed for vertical items and find a place for the additional containers we need to generate through Lemma 10. Finally, we show how to place the boxes for horizontal and small items.

Step 1: Fixing the position of the widest tall items intersecting $H/2$. We begin by inspecting the $1/(\delta^2\varepsilon)$ widest tall items crossing the horizontal line at $1/2H$. We refer to this set of items as $T_{1/2H}$. Each item in this set defines a new immovable item. Any box that contains this item gets split into three parts. The part to the left, the part to the right and the part containing the item. The parts to the left and right are reordered as any other box. The part that contains it is reordered differently. Since we have a tall item intersecting the horizontal line at $1/2H$, the parts above and below this item can have a height of at most $1/2H$. We define both of these parts as new boxes. Thus, we can reorder both of these boxes using Lemma 6. These reorderings both generate at most $\mathcal{O}(N)$ sub-boxes for tall and vertical items. This less than the number of boxes generated for a box of height greater than $3/4H$. Therefore, we can count this as a single box of height greater than $3/4H$ when calculating the number of added boxes and assume it to be at most $2/(\delta^2\varepsilon)$. After this step, the total number of boxes containing both tall and vertical items is bounded by $\mathcal{O}(1/(\delta^2\varepsilon))$. Furthermore, the number of vertical lines at box borders through the strip is bounded by the same amount, namely $\mathcal{O}(1/(\delta^2\varepsilon))$.

Ensuring the alignment of tall and medium boxes to the grid-lines. In both Lemma 7 and Lemma 9 we assume that each box with height larger than $1/2H$ starts and ends at grid points. We generate this property here. We define grid lines as integral multiples of $\varepsilon^2\text{OPT}$. Let B be a box with $h(B) > 1/2H$. Inspect the horizontal line ℓ at the smallest multiple of $\varepsilon^2\text{OPT}$ in the box B . The distance between ℓ and the bottom border clearly has to be smaller than $\varepsilon^2\text{OPT}$. Now, we remove all vertical items below ℓ and each item cut by ℓ and place them inside an extra box. We know that each item with height larger than εOPT already starts and ends at a multiple of $\varepsilon^2\text{OPT}$ due to our rounding, Lemma 3, all items we placed inside this extra box have a height less than εOPT . We generate another box in the same manner at the top of B , and repeat this procedure for every other box. This leaves us with two boxes of height less than εOPT for each box of height greater than $1/2H$. Since no two boxes of height greater than $1/2H$ can overlap, the total width of these newly generated boxes is at most $2W$. Therefore, we can place them by stacking two boxes atop another every time. This generates a new box of height at most $2(\varepsilon + \varepsilon^2)\text{OPT}$ and width W . We place this wide box atop the final packing, at height $5/4H + \varepsilon\text{OPT}$. Clearly, this does not result in any overlapping.

Step 3: Reordering tall and vertical items inside the boxes. After the necessary condition is fulfilled, we can begin reordering the boxes in $\mathcal{B}_{T \cup V}$. We do this by applying the either Lemma 6, Lemma 7 or Lemma 9, depending on which of those is applicable to the box we are currently reordering. To accurately estimate the number of generated sub-boxes we have to assume that each box in $\mathcal{B}_{T \cup V}$ is one that generates the largest number of sub-boxes. In our case, these are boxes with height greater than $3/4H$, i.e. boxes solved with Lemma 9. In every such box B , we draw a vertical line at the left border of each contained sub-box. If a sub-box for vertical items is intersected by such a line we split the sub-box there. Each of these lines intersects at most three boxes for vertical items, because there can be at most four boxes atop another at any point inside B . Therefore, we introduce at most three new boxes per vertical line. Since there are at most $\mathcal{O}(1/(\delta^2\varepsilon^5))$ sub-boxes for tall and vertical items, we generate at most $\mathcal{O}(1/(\delta^2\varepsilon^5))$ vertical lines as well. Since each of these lines splits a constant number of boxes, the number of boxes for vertical items after splitting is still bounded by $\mathcal{O}(1/(\delta^2\varepsilon^5))$.

The area between two consecutive lines defines a strip where the height of all intersected boxes does not change. Consequently, we have at most $\mathcal{O}(1/(\delta^2\varepsilon^5))$ of these strips in total. We define N_S as this number of strips.

Step 4: Placing the extra boxes for vertical items. By Lemma 10, we need at most $\mathcal{O}(|H_V| + |\mathcal{B}_P|)$ additional boxes with height $1/4H$ and width μW to place the vertical items continuously inside the boxes, where \mathcal{B}_P are the boxes for vertical items created to this point. We call this set of additional boxes $\mathcal{B}_{\mu W}$. We can bound the relevant variables as follows: There are at most $|\mathcal{B}_P| \in \mathcal{O}(1/(\delta^2\varepsilon^5))$ boxes for vertical items and at most $|H_V| \leq 1/\delta\varepsilon$ different heights of items. Therefore, we need at most $N_F \in \mathcal{O}(1/(\delta^2\varepsilon^5))$ extra boxes $\mathcal{B}_{\mu W}$.

We have to place these additional boxes inside the packing area $W \times 5/4H$. We show that this is possible by considering the three possibilities we outlined above. Recall the vertical lines at the borders of boxes in $\mathcal{B}_{T \cup V}$. These N_L many lines generate at most $N_L + 1$ strips. Let W_T be the total width of the strips containing items from $T_{1/2H}$, W_H the total width of strips containing boxes of height greater than $3/4H$ and W_R be the total width of the remaining strips. In total, we must have $W_T + W_H + W_R = W$. Let $|\mathcal{B}_{T \cup V}| = N_B$. We can assume $N_B \leq c_B/(\delta^2\varepsilon)$, $N_S \leq c_S/(\delta^2\varepsilon^5)$, $N_F \leq c_F/(\delta^2\varepsilon^5)$ and $N_L \leq c_L/(\delta^2\varepsilon)$ for some constants $c_B, c_S, c_F, c_L \in \mathbb{N}$. At this point, it is necessary to define the function f from Lemma 2 to find the values δ and μ more precisely. We specify $f(\varepsilon)$ by choosing $k \leq (4(c_B + c_F + c_L)c_S)$. Therefore, $\mu \leq \delta^2\varepsilon^{11}/(4(c_B + c_F + c_L)c_S)$ holds.

Consider the strips that make up the value W_R , i.e. strips without boxes of height greater than $3/4H$ or items in $T_{1/2H}$. These strips can contain boxes with height larger than $1/2H$. Therefore, we have a free area with total width of at least $\varepsilon^2 W_R$ in these strips.

▷ **Claim 5.** If $W_R \geq \varepsilon^4 W$, we can place the N_F boxes $\mathcal{B}_{\mu W}$ into the free areas in strips that comprise W_R .

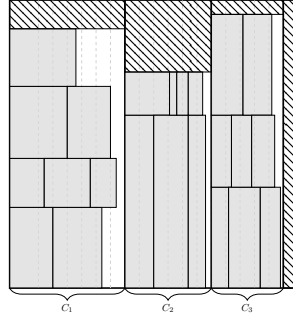
Proof. The considered strips might contain boxes with height larger than $1/2H$ but less than $3/4H$. Otherwise, they would have been counted in W_H . Therefore, the free area in these strips will be partially used by extra boxes for pseudo items from these boxes. Nevertheless, we know from Lemma 7 that these strips contain free area of width at least $\varepsilon^2 W_R$ that we can use to place the boxes $\mathcal{B}_{\mu W}$. In each of these at most $N_L + 1$ strips the free area is contiguous. However, we have to calculate a small error that might occur: Each of the boxes in $\mathcal{B}_{\mu W}$ has a width of μW and therefore, there is a residual width of up to μW where we cannot place a box from $\mathcal{B}_{\mu W}$ in each strip. See Figure 17 for an intuitive illustration of this issue.

On the other hand, we can use an area with total width of at least $\varepsilon^2 W_R - (N_L + 1)\mu W$ to place the boxes in $\mathcal{B}_{\mu W}$, since there are at most $N_L + 1$ strips. Therefore, if $\varepsilon^2 W_R - (N_L + 1)\mu W \geq N_F \mu W$, we can place all the boxes. Using $\mu := \delta^2\varepsilon^{11}/(4(c_B + c_F + c_L)c_S)$ it holds that

$$N_F \mu W + (N_L + 1)\mu W = \mu W (c_F/\delta^2\varepsilon^5 + c_L/\delta^2\varepsilon + 1) \leq \varepsilon^{11} W / \varepsilon^5 \leq \varepsilon^6 W.$$

Therefore, if $W_R \geq \varepsilon^4 W$, it holds that $\varepsilon^2 W_R - (N_L + 1)\mu W \geq N_F \mu W$ and we can place all the boxes $\mathcal{B}_{\mu W}$, proving the claim. ◀

With this option proven to be feasible, let us now focus our attention on another possibility to place the boxes in $\mathcal{B}_{\mu W}$, depending on the total width of strips containing tall items overlapping the horizontal line at $1/2H$. Since we have already shown how to finish the placement of the new boxes if there is ample space in W_R , we know that this is not the case.



■ **Figure 17** An illustration of the placement of the strips containing the additional strips of vertical items generated by Lemma 10. As you can see, we may fail to use a width of up to μW to place these items for each area we find to place these items in.

▷ **Claim 6.** If $W_H < \varepsilon^6 W / (4c_S)$ and $W_R < \varepsilon^4 W$, we can place all the boxes for vertical items inside the boxes of height at least $3/4H$.

Proof. Due to the definition of W_T , we know that $W_H = W - (W_T + W_R) > (1 - 2\varepsilon^4)W \geq \varepsilon W$. Additionally, each tall item not in $T_{1/2H}$ crossing $1/2H$ has a width of at most $W_T \cdot \delta^2 \varepsilon < \varepsilon^7 \delta^2 W / 4c_S := w_{max}$. This is because there would not be enough space left inside W_H if these items were any wider. After reordering the inside of the boxes, there are N_S strips total inside the boxes. We want to determine the total height of free area inside a strip. This area may not be contiguous, as there can be an item in the middle of any of these strips, with free area above and below it. However, we can address this issue trivially by slicing this item downwards, leaving only one contiguous area of free space. In the shifting step, we have added a total area of $W_H(1/4H + \varepsilon \text{OPT})$ to all of these strips. Let \check{W}_H be the total width of the strips containing free area with total height less than $1/4H$. Since the boxes in $\mathcal{B}_{\mu W}$ have a height of up to $1/4H$, we want to use areas with at least that height to place them. Let \hat{W}_H be the total width of these strips. Clearly, we have $\check{W}_H + \text{hat}W_H = W_H$. We have to show the following to be able to place the extra boxes as desired:

It holds that $\hat{W}_H \geq \varepsilon W_H$.

The total free area can have a height of at most $3/4\varepsilon \text{OPT}$ inside each strip, because there either are boxes at the top and bottom of height at least $1/4H$, or one box of height at least $3/4H$ at the bottom. Therefore, we have

$$1/4 \cdot \check{W}_H + (3/4H + \varepsilon \text{OPT}) \cdot \hat{W}_H \geq W_H(1/4H + \varepsilon \text{OPT})$$

since the free area in \check{W}_H has a total height of at most $1/4H$, the free area in \hat{W}_H has a height of at most $(3/4H + \varepsilon \text{OPT})$ and the total free area is bounded by $W_H(1/4H + \varepsilon \text{OPT})$.

As a consequence, we can prove that \hat{W}_H has a sufficient minimum size. The following holds:

$$\begin{aligned} W_H(1/4H + \varepsilon \text{OPT}) &\leq 1/4H \cdot \check{W}_H + (3/4H + \varepsilon \text{OPT}) \cdot \hat{W}_H \\ &= 1/4H \cdot W_H + (1/2H + \varepsilon \text{OPT}) \hat{W}_H \\ &= 1/4H \cdot W_H + ((1 + 2\varepsilon)\text{OPT}/2 + \varepsilon \text{OPT}) \cdot \hat{W}_H \\ &= 1/4H \cdot W_H + ((1 + 4\varepsilon)\text{OPT}/2) \cdot \hat{W}_H. \end{aligned}$$

This allows us to deduce that

$$\varepsilon W_H \leq ((1 + 4\varepsilon)/2) \cdot \hat{W}_H.$$

Therefore,

$$\hat{W}_H \geq 2\varepsilon W_H / (1 + 4\varepsilon) \geq \varepsilon W_H, \quad \forall 0 < \varepsilon \leq 1/4,$$

which proves the claim.

As a direct consequence of this claim, strips with total width of at least εW_H contain free area with total height of at least $1/4 H$. We have already alluded to the fact that, even when the area is split by an item in the middle, we can fuse these areas together trivially.

In each strip, there is an area with width of at most μW that we cannot use to place the boxes. Therefore, we can place all boxes for previously separated vertical items if $\varepsilon W_H - 2w_{\max} N_S - \mu W N_S \geq \mu W N_F$. It holds that

$$\begin{aligned} & 2w_{\max} N_S + \mu W N_S + \mu W N_F \\ & \leq (\varepsilon^7 \delta^2 W / (2c_S))(c_S / \delta^2 \varepsilon^5) + \mu W (c_S / \delta^2 \varepsilon^5 + c_F / \delta^2 \varepsilon^5) \\ & \leq \varepsilon^2 W / 2 + \varepsilon^6 W \leq \varepsilon^2 W. \end{aligned}$$

Thus, if $W_H \geq \varepsilon W$, we know that $\varepsilon W_H - 2w_{\max} N_S - \mu W N_S \geq \mu W N_F$ and we can place all boxes in the designated areas for this case. \blacktriangleleft

In this step, we create at most $2N_S \in \mathcal{O}(1/(\varepsilon^5 \delta^2))$ new boxes for tall items and no new box for vertical items. The boxes for tall items already contain only tall items of the same height. Therefore, we introduce at most $\mathcal{O}(1/(\varepsilon^5 \delta^2))$ boxes for vertical items \mathcal{B}_V such that each box $B \in \mathcal{B}_V$ contains only items with height $h(B)$.

The boxes for small items. The free area inside the boxes from the partition in Lemma 5 for horizontal, tall and vertical items is at least as large as the total area for the small items, because the small items were contained in the optimal packing.

Bounding the packing height. To bound the total height of the packing we summarize which heights we added during the process so far. In the beginning, we had the optimal packing of height OPT . Through our rounding with Lemma 3, we generated a packing of height $(1 + 2\varepsilon)\text{OPT}$. When we extended boxes containing tall and vertical items of height at least $3/4 H$, we added $1/4 H + \varepsilon \text{OPT} \leq 1/4(1 + 2\varepsilon)\text{OPT} + \varepsilon \text{OPT}$ to the packing height. Earlier in this proof, to ensure that all boxes with height greater than $1/2 H$ start and end at multiples of $\varepsilon^2 \text{OPT}$ we shifted some vertical items. This procedure added a further $2(\varepsilon + \varepsilon^2)\text{OPT}$ to the packing height. Thus, in total, we have added at most $1/4(1 + 2\varepsilon)\text{OPT} + 2(\varepsilon + \varepsilon^2)\text{OPT} \leq (1/4 + 3\varepsilon)\text{OPT}$ to the packing height. This assumes sufficiently small, i.e. smaller than $1/2$ values for ε . As such, the structured packing has a height of at most $(5/4 + 5\varepsilon)\text{OPT}$. \blacktriangleleft

We employ a dynamic program to iterate through all possible configurations of these boxes until we find a feasible packing of the items into the boxes.

Step 6

We utilize some techniques developed in [15] to place the small and medium items without exceeding the desired height of the packing.

► Lemma 13. *It is possible to place the small items inside the boxes \mathcal{B}_V^S and \mathcal{B}_H^S and one extra box with width W and height at most $2\varepsilon \delta^6 \text{OPT}$.*

Proof. This is an adaptation of a similar procedure in [15], considering the slicing of items. Recall how we started partitioning the original optimal packing into the boxes in Lemma 5.

Since this optimal packing contained the small items as well as the items we chose as the basis of our partitioning, the total remaining area inside those boxes must be enough to place these small items. In Lemmas 10 and 11 we generate at most $\mathcal{O}(1/(\varepsilon^5 \delta^2))$ empty boxes which we can use to place the small items. We have shown that the total area of those empty boxes is at least as large as the empty space left in the original boxes \mathcal{B}_H and $\mathcal{B}_{T \cup V}$ in Lemma 5. As \mathcal{B}_L only ever contain a single item, there is no empty space in these boxes. Therefore, the total area in those empty boxes is at least as large as the total area of small items.

Let \mathcal{B}_S be the set of boxes for small items and $|\mathcal{B}_S| = c/\delta^2 \varepsilon^5$ for some constant $c \in \mathbb{N}$. We show that we only need one small extra box to place all the small items into \mathcal{B}_S using the Next Fit Decreasing Height (NFDH) algorithm[17].

First, we discard any box that is too small in either dimension, i.e. has a height of less than μOPT or a width of μW . Clearly, the total area of discarded boxes is at most $\mu W \text{OPT}$. Consider a box B with height and width greater than μOPT and μW respectively. The last item we want to place on each shelf during the NFDH-Algorithm might not fit from its width. As such, we may have an empty space of at most μW on each shelf, as all small items have at most that width. Furthermore, the final shelf has a distance of at most μOPT to the top border of B . Again, this area might remain free if no suitable items are left. Finally, the free area between two shelves used in the algorithm has a size of at most $\mu \text{OPT} \cdot w(B)$, again because all small items have at most that size. As a result, the total free area inside the box B is at most $\mu W \cdot h(B) + 2\mu \text{OPT} \cdot w(B) \leq 3\mu W \text{OPT}$. We have this free area for every box $B \in \mathcal{B}_S$. Therefore, the total area of items that could not be placed inside boxes is at most $3\mu W \text{OPT} \cdot c/\delta^2 \varepsilon^5$. Since we have defined $\mu \leq \varepsilon^{11} \delta^2/k$ for some suitable constant k through Lemma 2 it holds that $3\mu W \text{OPT} \cdot c/\delta^2 \varepsilon^5 \leq \varepsilon^6 W \text{OPT}$ when choosing $k \geq c$.

These remaining items can then be placed into a box of width W and height $2\varepsilon^6 \text{OPT}$ using Steinbergs algorithm [27]. The relevant conditions to apply this algorithm are met because each item has a height of at most μOPT . ◀

Having placed the small items, there is only one set of items left to place before we have successfully shown that our partition into boxes allows us to generate a structured packing while only adding a height of $(1/4)\text{OPT} + \mathcal{O}(\varepsilon \text{OPT})$ to the top of the packing. Thus, we now show that the medium items, of which there are few, can be placed in a structured manner.

► **Lemma 14.** [15] *It is possible to place the medium items M into a box of width W and height at most $2\varepsilon \text{OPT}$.*

Proof. We know, from Lemma 2, that the total area of medium items is small. Thus, we use a well defined algorithm to place these items at an approximately greater loss. As we did with the small items, we use the NFDH algorithm for this purpose [17]. Before we apply the algorithm, we sort all items by their width, in descending manner. We know that $a(M)$ is bounded by $\varepsilon^{13} W \text{OPT}$ and the widest item in M has a width of at most εOPT . Therefore, we can place these items in a box of width W and height of at most $2\varepsilon^{13} \text{OPT} + \varepsilon \text{OPT} \leq 2\varepsilon \text{OPT}$. ◀

Step 7

The algorithm iterates over the previous steps until the minimal value for H' has been found and returns the associated packing. Because we have shown that any optimal packing of height OPT can be transformed into a structured packing of height $(5/4 + \varepsilon)\text{OPT}$, we know that $H' \leq \text{OPT}$ must hold. Therefore, our algorithm generates a solution with height at most the desired approximation ratio.

Running time:

Scaling the instance can be done in $\mathcal{O}(n \log(n))$. The binary search in step 2 runs logarithmically in its step-length. Due to our rounding, this length is n/ε . Therefore, the search can be computed in $\mathcal{O}(\log(n/\varepsilon))$. For every value of the search, we have to guess the starting points for horizontal items. As there are $\mathcal{O}_\varepsilon(1)$ of these boxes, we can guess these in $W^{\mathcal{O}_\varepsilon(1)}$ time. Then, we guess all possible configurations of boxes inside the packing. Again, there are $\mathcal{O}_\varepsilon(1)$ many of those, yielding a running time of $W^{\mathcal{O}_\varepsilon(1)}$ for this step. As a result, the running time can be expressed as $\mathcal{O}(n \log(n)) + \mathcal{O}(\log(n/\varepsilon)) \cdot W^{\mathcal{O}_\varepsilon(1)} = \mathcal{O}(n \log(n)) \cdot W^{\mathcal{O}_\varepsilon(1)}$. This completes the proof of Theorem 5.

4 Conclusion

We showed that the DEMAND STRIP PACKING problem remains hard to solve with an approximation ratio lower than $(5/4) \cdot \text{OPT}$, even in pseudo-polynomial time. In the process of this proof, we developed a transformation algorithm showing that we can treat DSP and PTS as somewhat dual problems of one another. We provided a framework using known results to solve either of these problems optimally when admitting some resource augmentation. These results are valuable in applications where penalty costs for exceeding makespan or power demand are large.

Finally, we have developed an approximation algorithm for DSP that runs in pseudo-polynomial time. The adaptations made bridge the known integrality gap between SP and DSP. Furthermore, its approximation ratio of $(5/4 + \varepsilon) \cdot \text{OPT}$ almost matches the proven hardness except for a negligibly small ε . As a contrast, in polynomial time, the gap between proven hardness $(3/2)$ [31] and best-known approximation $(5/3 + \varepsilon)$ [3, 6] remains quite large.

Similar results for the SP algorithm suggest that the presented algorithm can be adapted to admit 90° item rotations or moldable jobs. Both of these settings are interesting in the context of smart grids. Rotations can model different modes of power consumption, i.e. fast versus slow charging. Moldable jobs can model the parallel operation of multiple machines for the same task. Each variation on the job/item represents a set number of machines assigned to that task.

Further open questions extend to the relation of DSP to other packing problems. A family of instances exists where the height of optimal solutions for SP and DSP differs by $5/4$ [2]. Furthermore, due to Steinberg's algorithm, this difference is bounded by 2. Closing this gap might imply better bounds on optimal solutions for either problem.

On the other hand, the discovered link between DSP and PTS may suggest similar links between DSP and other resource allocation problems. Amongst these, the well-known STORAGE ALLOCATION PROBLEM (SAP) [20] and UNSPLITTABLE FLOW ON A PATH (UFP) [9] problems are promising candidates. So far, a PTAS is known only for the latter [8]. Techniques used in the presented algorithm may be helpful in adapting existing algorithms for SAP to UFP, or vice versa. Furthermore, similar resource augmentation arguments as we made for PTS and DSP may be made for either problem should a link between DSP and UFP or SAP exist. Finally, techniques used in the presented approximation scheme may be useful when developing an approximation scheme for SAP.

References

- 1 Brenda S. Baker, Edward G. Coffman Jr., and Ronald L. Rivest. Orthogonal packings in two dimensions. *SIAM J. Comput.*, 9(4):846–855, 1980.

- 2 Iwo Blazek, Maciej Drozdowski, Frédéric Guinand, and Xavier Schepler. On contiguous and non-contiguous parallel task scheduling. *J. Sched.*, 18(5):487–495, 2015.
- 3 Max A. Deppert, Klaus Jansen, Arindam Khan, Malin Rau, and Malte Tutas. Peak demand minimization via sliced strip packing. In *APPROX-RANDOM*, volume 207 of *LIPIcs*, pages 21:1–21:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- 4 G. Dileep. A survey on smart grid technologies and applications. *Renewable Energy*, 146:2589–2625, 2020.
- 5 Jianzhong Du and Joseph Y.-T. Leung. Complexity of scheduling parallel task systems. *SIAM J. Discret. Math.*, 2(4):473–487, 1989.
- 6 Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, and Kamyar Khodamoradi. Approximation algorithms for demand strip packing. *CoRR*, abs/2105.08577, 2021.
- 7 Waldo Gálvez, Fabrizio Grandoni, Salvatore Ingala, and Arindam Khan. Improved pseudo-polynomial-time approximation for strip packing. In *FSTTCS*, volume 65 of *LIPIcs*, pages 9:1–9:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.
- 8 Fabrizio Grandoni, Tobias Mömke, and Andreas Wiese. A PTAS for unsplittable flow on a path. In *STOC*, pages 289–302. ACM, 2022.
- 9 Fabrizio Grandoni, Tobias Mömke, and Andreas Wiese. Unsplittable flow on a path: The game! In *SODA*, pages 906–926. SIAM, 2022.
- 10 Rolf Harren, Klaus Jansen, Lars Prädell, and Rob van Stee. A $(5/3 + \epsilon)$ -approximation for strip packing. *Comput. Geom.*, 47(2):248–267, 2014.
- 11 Rolf Harren and Rob van Stee. Improved absolute approximation ratios for two-dimensional packing problems. In *APPROX-RANDOM*, volume 5687 of *Lecture Notes in Computer Science*, pages 177–189. Springer, 2009.
- 12 Sören Henning, Klaus Jansen, Malin Rau, and Lars Schmarje. Complexity and inapproximability results for parallel task scheduling and strip packing. In *CSR*, volume 10846 of *Lecture Notes in Computer Science*, pages 169–180. Springer, 2018.
- 13 Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *J. ACM*, 34(1):144–162, 1987.
- 14 Klaus Jansen and Malin Rau. Improved approximation for two dimensional strip packing with polynomial bounded width. In *WALCOM*, volume 10167 of *Lecture Notes in Computer Science*, pages 409–420. Springer, 2017.
- 15 Klaus Jansen and Malin Rau. Closing the gap for pseudo-polynomial strip packing. In *ESA*, volume 144 of *LIPIcs*, pages 62:1–62:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
- 16 Klaus Jansen and Ralf Thöle. Approximation algorithms for scheduling parallel jobs. *SIAM J. Comput.*, 39(8):3571–3615, 2010.
- 17 Edward G. Coffman Jr., M. R. Garey, David S. Johnson, and Robert Endre Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM J. Comput.*, 9(4):808–826, 1980.
- 18 Landis D. Kannberg, David P. Chassin, John G. DeStee, Steve G. Hauser, Michael C. Kintner-Meyer, Robert G. Pratt, Lawrence A. Schienbein, and W. Michael Warwick. Gridwisetm: The benefits of a transformed energy system. Technical report, Pacific Northwest National Laboratory, 2003.
- 19 Mohammad M. Karbasioun, Gennady Shaikhov, Ioannis Lambadaris, and Evangelos Kranakis. Asymptotically optimal scheduling of random malleable demands in smart grid. *Discret. Math. Algorithms Appl.*, 10(2):1850025:1–1850025:25, 2018.
- 20 Tobias Mömke and Andreas Wiese. Breaking the barrier of 2 for the storage allocation problem. In *ICALP*, volume 168 of *LIPIcs*, pages 86:1–86:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- 21 Giorgi Nadiradze and Andreas Wiese. On approximating strip packing with a better ratio than $3/2$. In *SODA*, pages 1491–1510. SIAM, 2016.

- 22 Anshu Ranjan, Pramod Khargonekar, and Sartaj Sahni. Offline preemptive scheduling of power demands to minimize peak power in smart grids. In *2014 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2014.
- 23 Anshu Ranjan, Pramod Khargonekar, and Sartaj Sahni. Offline first fit scheduling in smart grids. In *2015 IEEE Symposium on Computers and Communication (ISCC)*, pages 758–763. IEEE, 2015.
- 24 Ingo Schiermeyer. Reverse-fit: A 2-optimal algorithm for packing rectangles. In *ESA*, volume 855 of *Lecture Notes in Computer Science*, pages 290–299. Springer, 1994.
- 25 Pierluigi Siano. Demand response and smart grids—a survey. *Renewable and sustainable energy reviews*, 30:461–478, 2014.
- 26 Daniel Dominic Sleator. A 2.5 times optimal algorithm for packing in two dimensions. *Inf. Process. Lett.*, 10(1):37–40, 1980.
- 27 A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM J. Comput.*, 26(2):401–409, 1997.
- 28 Shaojie Tang, Qiuyuan Huang, Xiang-Yang Li, and Dapeng Wu. Smoothing the energy consumption: Peak demand reduction in smart grid. In *32nd IEEE International Conference on Computer Communications (INFOCOM)*, pages 1133–1141. IEEE, 2013. doi:10.1109/INFOCOM.2013.6566904.
- 29 Shaojie Tang, Qiuyuan Huang, Xiang-Yang Li, and Dapeng Wu. Smoothing the energy consumption: Peak demand reduction in smart grid. In *2013 Proceedings IEEE INFOCOM*, 2013.
- 30 Jacopo Torriti. *Peak Energy Demand and Demand Side Response*. Routledge, 2016.
- 31 Sean Yaw, Brendan Mumey, Erin McDonald, and Jennifer Lemke. Peak demand scheduling in the smart grid. In *2014 IEEE international conference on smart grid communications (SmartGridComm)*, pages 770–775. IEEE, 2014.