Breaking the barrier of 2 for the storage allocation problem

Tobias Mömke, Andreas Wiese

Angaben zur Veröffentlichung / Publication details:

Mömke, Tobias, and Andreas Wiese. 2020. "Breaking the barrier of 2 for the storage allocation problem." In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, edited by Artur Czumaj, Anuj Dawar, and Emanuela Merelli, 86:1–19. Wadern: Schloss Dagstuhl - Leibniz-Zentrum für Informatik GmbH. https://doi.org/10.4230/LIPIcs.ICALP.2020.86.

Breaking the Barrier of 2 for the Storage Allocation Problem

Tobias Mömke
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
moemke@cs.uni-saarland.de

Andreas Wiese
Department of Industrial Engineering, Universidad de Chile, Santiago, Chile
awiese@dii.uchile.cl

Abstract

Packing problems are an important class of optimization problems. The probably most well-known problem if this type is knapsack and many generalizations of it have been studied in the literature like Two-dimensional Geometric Knapsack (2DKP) and Unsplittable Flow on a Path (UFP). For the latter two problems, recently the first polynomial time approximation algorithms with better approximation ratios than 2 were presented [Gálvez et al., FOCS 2017][Grandoni et al., STOC 2018]. In this paper we break the barrier of 2 for the Storage Allocation Problem (SAP), a problem which combines properties of 2DKP and UFP. In SAP, we are given a path with capacitated edges and a set of tasks where each task has a start vertex, an end vertex, a size, and a profit. We seek to select the most profitable set of tasks that we can draw as non-overlapping rectangles underneath the capacity profile of the edges where the height of each rectangle equals the size of the corresponding task.

The problem SAP appears naturally in settings of allocating resources like memory, bandwidth, etc. where each request needs a contiguous portion of the resource. The best known polynomial time approximation algorithm for SAP has an approximation ratio of $2+\varepsilon$ [Mömke and Wiese, ICALP 2015] and no better quasi-polynomial time algorithm is known. We present a polynomial time $(63/32+\varepsilon) < 1.969$-approximation algorithm for the important case of uniform edge capacities and a quasi-polynomial time $(1.997+\varepsilon)$-approximation algorithm for non-uniform quasi-polynomially bounded edge capacities. Key to our results are building blocks consisting of *stair-blocks, jammed tasks,* and *boxes* that we use to construct profitable solutions and which allow us to compute solutions of these types efficiently. Finally, using our techniques we show that under slight resource augmentation we can obtain even approximation ratios of $3/2+\varepsilon$ in polynomial time and $1+\varepsilon$ in quasi-polynomial time, both for arbitrary edge capacities.

2012 ACM Subject Classification Theory of computation → Dynamic programming; Theory of computation → Packing and covering problems; Theory of computation → Rounding techniques

Keywords and phrases Approximation Algorithms, Resource Allocation, Dynamic Programming

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.86

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at [28], `https://arxiv.org/abs/1911.10871`.

Funding *Tobias Mömke*: This work was partially supported by the ERC Advanced Investigators Grant 695614 (POWVER), and by DFG Grant 389792660 as part of TRR 248 (CPEC).
Andreas Wiese: Partially supported by FONDECYT Regular grant 1170223.

1 Introduction

Packing problems play an important role in combinatorial optimization. The most basic packing problem is knapsack where we are given a knapsack of a certain capacity, a set of items with different sizes and profits, and we are looking for a subset of items of maximum profit that fit into the knapsack. Many generalizations of it have been studied. For example,

in the Two-dimensional Geometric Knapsack problem (2DKP) the items are axis-parallel rectangles and we seek to find the most profitable subset of them that fit non-overlappingly into a given rectangular knapsack. Another generalization of the knapsack problem is called Unsplittable Flow on a Path (UFP). We are given a path with capacities on its edges and each item can be interpreted as a commodity of flow that needs to send a given amount of flow from its start vertex to its end vertex in case that we select it. If the path consists of a single edge then UFP is identical to knapsack.

In this paper, we study the Storage Allocation Problem (SAP) which combines properties of 2DKP and UFP: We are given a path (V, E) where each edge $e \in E$ has a capacity $u_e \in \mathbb{N}$, and a set of tasks T where each task $i \in T$ is specified by a size $d_i \in \mathbb{N}$, a profit $w_i \in \mathbb{N}$, a start vertex $s_i \in V$, and an end vertex $t_i \in V$. Let $P(i)$ denote the path between s_i and t_i for each $i \in T$. The goal is to select a subset of tasks $T' \subseteq T$ and define a height level $h(i) \geq 0$ for each task $i \in T'$ such that the resulting rectangle $[s_i, t_i) \times [h(i), h(i) + d_i)$ lies within the profile of the edge capacities, and we require that the rectangles of the tasks in T' are pairwise non-overlapping. Formally, for each task $i \in T'$ we require that $h(i) + d_i \leq u_e$ for each edge $e \in P(i)$ and additionally for any two tasks $i, i' \in T'$ we require that if $P(i) \cap P(i') \neq \emptyset$, then $[h(i), h(i) + d_i) \cap [h(i'), h(i') + d_{i'}) = \emptyset$. Note that since we can choose $h(i)$ we can define the vertical position of the rectangle of each task i but not its horizontal position. Again, if E has only one edge then the problem is identical to knapsack.

The problem SAP is motivated by settings in which tasks need a contiguous portion of an available resource, e.g., a consecutive portion of the computer memory or a frequency bandwidth. Observe that in contrast to UFP, in many applications of SAP the instances have uniform edge capacities, e.g., if the available memory or frequency spectrum does not change over time. From a mathematical perspective, SAP and UFP are closely related. Every feasible SAP-solution T' satisfies $\sum_{i \in T' : e \in P(i)} d_i \leq u_e$ on each edge e. This is exactly the condition when a solution to UFP is feasible (UFP and SAP have the same type of input). In SAP we require additionally that we can represent the tasks in T' as non-overlapping rectangles. Also, if all edges have the same capacity then SAP can be seen as a variant of 2DKP in which the horizontal coordinate of each item i is fixed and we can choose only the vertical coordinate.

For quite some time, the best known polynomial time approximation ratios for 2DKP and UFP had been $2 + \varepsilon$ [24, 2]. Recently, the barrier of 2 was broken for both problems and algorithms with strictly better approximation ratios have been presented [16, 20]. For SAP, the best known approximation ratio is still $2 + \varepsilon$ [27], even for uniform edge capacities and if we allow quasi-polynomial running time. In contrast, for 2DKP and UFP, better quasi-polynomial time algorithms had been known earlier [3, 9, 1].

1.1 Our contribution

In this paper, we break the barrier of 2 for SAP and present a polynomial time $(63/32 + \varepsilon) < 1.969$-approximation algorithm for uniform edge capacities and a quasi-polynomial time $(1.997 + \varepsilon)$-approximation algorithm for non-uniform edge capacities in a quasi-polynomial range. Key to our results is to identify suitable building blocks to construct profitable near-optimal solutions such that we can design algorithms that find profitable solutions of this type. We call a task *small* if its demand is small compared to the capacity of the edges on its path and *large* otherwise. One can show that each edge can be used by only relatively few large tasks which allows for a dynamic program that finds the best solution with large tasks only. However, there can be many small tasks using an edge and hence this approach fails for small tasks. We therefore consider *boxable solutions* in which the tasks

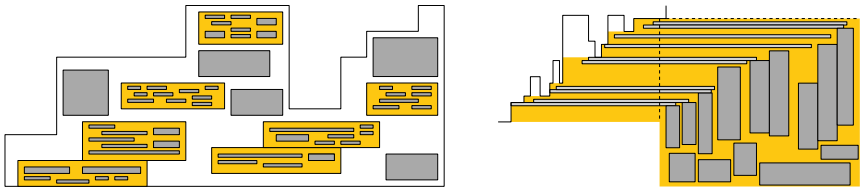

Figure 1 Left: a boxable solution in which the (gray) tasks are assigned into the (orange) boxes. Right: A stair-block into which small tasks (light gray) and large tasks (dark gray) are assigned. All small tasks need to cross the vertical dashed line and all large tasks need to be placed on the right of it underneath the dashed horizontal line. Therefore, the orange area denotes the space that is effectively usable for the tasks that we assign into the stair-block.

are assigned into rectangular boxes such that each edge is used by only $(\log n)^{O(1)}$ of these boxes, see Fig. 1. Using the latter property, we present a quasi-polynomial time algorithm that essentially finds the optimal boxable solution. Furthermore, for many types of instances we prove that there exist boxable solutions with high profit.

There are, however, instances for which it is not clear how to construct boxable solutions that yield a better approximation ratio than 2. This is where our second building block comes into play which are *stair-blocks*. Intuitively, a stair-block is an area into which we assign small and large tasks such that the small tasks are jammed between the large tasks and the capacity profile of the edges, see Fig. 1. We prove the crucial insight that if we fail to construct a good boxable solution then this is because a lot of profit of the optimum is due to small tasks in stair-blocks. We therefore devise a second algorithm that computes solutions for such instances, yielding an approximation ratio better than 2. The algorithm is based on a configuration-LP with a variable for each possible set of large tasks in each stair-block and additionally variables for placing the small tasks in the remaining space. We separate it via the dual LP in which the separation problem turns out to be a variation of SAP with large tasks only. Then we sample the set of large tasks according to the probabilities implied by the LP solution. As a result, there are some small tasks that we cannot pick anymore since they would overlap the sampled large tasks. For some small tasks this will happen with very large probability so most likely we will lose their profit. This is problematic if they represent a large fraction of the profit of the LP. We therefore introduce additional constraints that imply that if the latter happens then we can use another rounding routine for small tasks only that yields enough profit.

▶ **Theorem 1.** *There is a quasi-polynomial time* $(1.997 + \varepsilon)$*-approximation algorithm for SAP if the edge capacities are quasi-polynomially bounded integers.*

Recall that in many applications of SAP the instances have uniform edge capacities. For our polynomial time algorithm for this setting the above building blocks are not sufficient since for example in our boxable solutions above an edge can be used by more than constantly many boxes and hence we cannot enumerate all possibilities for those in polynomial time. We therefore identify types of boxable solutions that are more structured and that allow us to find profitable solutions of these types in polynomial time. The first such type are boxable solutions in which each edge is used by only constantly many boxes. A major difficulty is here that for a small task there are possibly several boxes that we can assign it to and if we assign it to the wrong box then it occupies space that we should have used for other tasks instead (in our quasi-polynomial time algorithm above we use a method to address this which inherently needs quasi-polynomial time). We solve this issue by guessing the boxes in a suitable hierarchical order which is *not* the canonical linear order given by their

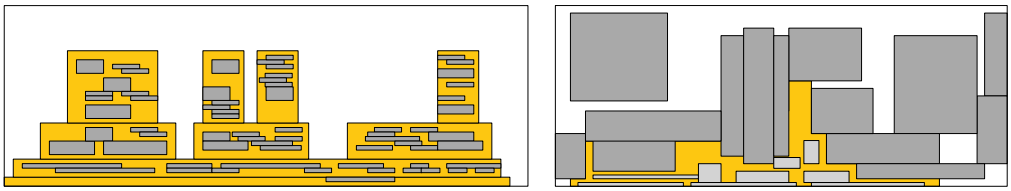

Figure 2 Left: a laminar boxable solution that consists of boxes of geometrically increasing sizes whose paths form a laminar family. Right: a jammed solution in which a set of small tasks (light gray) that are placed underneath some large tasks (dark gray). The small tasks are relatively large compared to the (orange) space underneath the large tasks.

respective leftmost edges and we assign the tasks into the boxes in the guessed order. With a double-counting argument we show that with our strategy we obtain a solution which has essentially at least the profit of the large tasks in the optimal boxable solution of the first type plus half of the profit of the small tasks. Our second special type of boxable solutions is the case in which the paths of the boxes form a laminar family and the sizes of the boxes are geometrically increasing, see Fig. 2. Even though there can be $\Omega(\log n)$ such boxes using an edge, we devise an algorithm with polynomial running time for this kind of solutions. It is a dynamic program inspired by [20] that guesses the boxes in the order given by the laminar family and assigns the tasks into them. Finally, there can be small tasks such that in the optimal solution the large tasks take away so much space that with respect to the remaining space those small tasks actually become relatively large. We say that a solution consisting of such small and large tasks forms a *jammed solution* which is our third type of building block, see Fig. 2. We extend an algorithm in [27] for instances with large tasks only to compute essentially the most profitable jammed solution. Our key technical lemma shows that for any instance there exists a profitable solution that uses only the building blocks above and we provide a polynomial time algorithm that finds such a solution.

▶ **Theorem 2.** *There is a polynomial time* $(63/32 + \varepsilon) < 1.969$*-approximation algorithm for SAP for uniform edge capacities.*

We would like to note that we did not attempt to optimize our approximation ratios up to the third decimal place but instead we focus on a clean exposition of our results (which are already quite complicated). Finally, we study the setting of $(1 + \eta)$-resource augmentation where we can increase the capacity of each edge by a factor of $1 + \eta$ for an arbitrarily small constant $\eta > 0$ while the compared optimal solution cannot do this. In this case we obtain even better approximation ratios and improve the factor of 2 for arbitrary edge capacities even with a polynomial time algorithm. Key for these results is to show that using the resource augmentation we can reduce the general case to the case of a constant range of edge capacities and then establish that there are essentially optimal boxable solutions in which each edge is used by a constant number of boxes. Using our algorithmic tools from above this implies the following theorem.

▶ **Theorem 3.** *In the setting of* $(1 + \eta)$*-resource augmentation there exists a polynomial time* $(3/2 + \varepsilon)$*-approximation algorithm and a quasi-polynomial time* $(1 + \varepsilon)$*-approximation algorithm for SAP with arbitrary edge capacities.*

Due to space constraints, this extended abstract intends to give only an overview of our methodology. For a complete presentation of our results we refer to [28].

1.2 Other related work

Previous to the mentioned $(2+\varepsilon)$-approximation algorithm for SAP [27], Bar-Noy et al. [6] found a 7-approximation algorithm if all edges have the same capacities which was improved by Bar-Yehuda et al. to a $(2+\varepsilon)$-approximation [7]. Bar-Yehuda et al. [8] presented the first constant factor approximation algorithm for SAP for arbitrary capacities, having an approximation ratio of $9+\varepsilon$. A related problem is the dynamic storage allocation problem (DSA) where in the input we are given a set of tasks like in SAP and we all need to pack all of them as non-overlapping rectangles, minimizing the maximum height of a packed item. The best known approximation ratio for DSA is a $(2+\varepsilon)$-approximation which in particular uses a $(1+\varepsilon)$-approximation if all tasks are sufficiently small [11]. This improves earlier results [25, 26, 17, 18].

For 2DKP for squares there is an EPTAS [21] which improves earlier PTASs [22, 23]. For rectangles, there was a $(2+\varepsilon)$-approximation known [24, 23] which was improved to a $(17/9+\varepsilon)$-approximation [16]. There is a PTAS if the profit of each item is proportional to its area [5]. Also, there is a QPTAS for quasi-polynomially bounded input data [1]. For UFP there is a long line of work on the case of uniform edge capacities [29, 6, 12], the no-bottleneck-assumption [13, 15], and the general case [3, 4, 14, 10, 2] which culminated in a QPTAS [3, 9], PTASs for several special cases [19, 9], a $(2+\varepsilon)$- approximation [2], which was improved to a $(5/3+\varepsilon)$-approximation [20].

2 Overview

In this section we present an overview of our methodology for our algorithms. Let $\varepsilon > 0$ and assume that $1/\varepsilon \in \mathbb{N}$. First, we classify tasks into large and small tasks. For each task $i \in T$ let $b(i) := \min_{e \in P(i)} u_e$ denote the *bottleneck capacity* of i. For constants $\mu, \delta > 0$ we define that a task i is *large* if $d_i > \delta \cdot b(i)$ and *small* if $d_i \leq \mu \cdot b(i)$. The constants δ, μ are chosen to be the values δ_{i^*} and μ_{i^*} due to the following lemma, which in particular ensures that the tasks i with $\mu \cdot b(i) < d_i \leq \delta \cdot b(i)$ contribute only a marginal amount to the optimal solution OPT whose weight we denote by opt.

▶ **Lemma 4.** *We can compute a set $(\mu_1, \delta_1), \ldots, (\mu_{1/\varepsilon}, \delta_{1/\varepsilon})$ such that for each tuple (μ_k, δ_k) we have $\varepsilon^{O\left((1/\varepsilon)^{1/\varepsilon}\right)} \leq \mu_k \leq \varepsilon^{10}\delta_k^{1/\varepsilon}$, $\delta_i \leq \varepsilon$ and for one tuple $(\mu_{k^*}, \delta_{k^*})$ it holds that $w(\mathrm{OPT} \cap \{i \in T \mid \mu_{k^*} \cdot b(i) < d_i \leq \delta_{k^*} \cdot b(i)\}) \leq \varepsilon \cdot \mathrm{opt}$.*

Let T_L and T_S denote the sets of large and small input tasks, respectively. For each edge e let $T_e \subseteq T$ denote the set of tasks $i \in T$ for which $e \in P(i)$. We will show later that for many instances there are profitable solutions that are *boxable* which intuitively means that the tasks can be assigned into rectangular boxes such that each edge is used by only few boxes. A *box* B is defined by a start vertex s_B, an end vertex t_B, and a size d_B. We define $P(B)$ to be the path of B which is the path between s_B and t_B. A set of tasks $T' \subseteq T$ *fits* into B if

- for each $i \in T'$ we have that $P(i) \subseteq P(B)$, and
- there is a value $h(i) \in [0, d_B)$ for each $i \in T'$ such that (T', h) is feasible if each edge $e \in P(B)$ has capacity d_B, and
- $|T'| = 1$ or we have $d_i \leq \varepsilon^8 \cdot d_B$ for each $i \in T'$.

We say that a set of boxes $\mathcal{B}$ and a height level assignment $h : \mathcal{B} \to \mathbb{N}$ forms a feasible solution $(\mathcal{B}, h)$ if the boxes in $\mathcal{B}$ interpreted as tasks form a feasible solution with h (see Fig. 1), i.e., if the set $(T(\mathcal{B}), h')$ is feasible where $T(\mathcal{B})$ contains a task $i(B)$ for each $B \in \mathcal{B}$ such that $P(i(B)) = P(B)$, $d_i = d_{i(B)}$ and $h'(i(B)) = h(B)$.

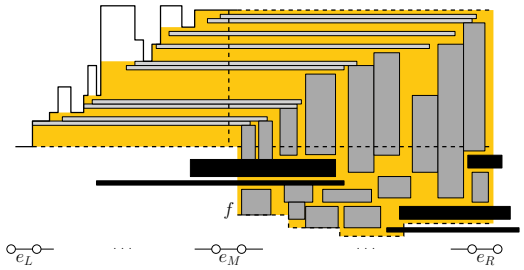

Figure 3 A stair-block $\mathcal{SB} = (e_L, e_M, e_R, f, T'_L, h')$. The black tasks are the tasks in T'_L. The orange area denotes the area that is effectively usable for the tasks that we assign to $\mathcal{SB}$. The light and dark gray tasks are small and large tasks, respectively, that together fit into $\mathcal{SB}$.

▶ **Definition 5.** *A solution (T', h') is a* β-boxable solution *if there exists a set of boxes $\mathcal{B} = \{B_1, \dots, B_{|\mathcal{B}|}\}$ and a partition $T' = T'_1 \dot\cup \dots \dot\cup T'_{|\mathcal{B}|}$ such that*

- *for each $j \in [|\mathcal{B}|]$, T'_j fits into the box B_j and if $T'_j \cap T_L \neq \emptyset$ then $|T'_j| = 1$,*
- *each edge $e \in E$ is used by the paths of at most β boxes in $\mathcal{B}$,*
- *there is a height level $h'(B)$ for each box $B \in \mathcal{B}$ such that $(\mathcal{B}, h')$ is feasible.*

In the following lemma we present an algorithm that essentially computes the optimal β-boxable solution. We will use it later with $\beta = (\log n)^{O(1)}$. Assume in the sequel that we are given a SAP-instance where $u_e \leq n^{(\log n)^c}$ for some $c \in \mathbb{N}$ for each $e \in E$.

▶ **Lemma 6.** *Let $\beta \in \mathbb{N}$ and let $(T_{\text{box}}, h_{\text{box}})$ be a β-boxable solution. There is an algorithm with running time $n^{(\beta \log n/aw\varepsilon)^{O(c)}}$ that computes a β-boxable solution (T', h') with $w(T') \geq w(T_{\text{box}})/(1+\varepsilon)$.*

Our second type of solutions are composed by *stair-blocks* (see Fig. 1 and Fig. 3). Intuitively, a stair-block is an area underneath the capacity profile defined by a function $f\colon E \to \mathbb{N}_0$ and three edges e_L, e_M, e_R, where e_M lies between e_L and e_R. The corresponding area contains all points above each edge between e_L and e_M whose y-coordinate is at least u_{e_L} and all points above each edge e between e_M and e_R whose y-coordinate is in $[f_e, u_{e_M})$. Additionally, there are some tasks $T'_L \subseteq T_L \cap (T_{e_M} \cup T_{e_R})$ and a function $h' : T'_L \to \mathbb{N}_0$ that assigns height levels to them where the intuition is that those tasks are given in advance and fixed. We require that each of them intersects the mentioned area below u_{e_L}, i.e., for each $i \in T'_L$ we have that $h'(i) + d_i \leq u_{e_L}$ and there is an edge $e \in P(i) \cap P_{e_M,e_R} \setminus \{e_M\}$ such that $h'(i) + d_i > f_e$ where P_{e_M,e_R} is the path that starts with e_M and ends with e_R. Also, we require that $f(e) = u_{e_L}$ for $e = e_M$ and each edge e on the left of e_M.

Given a stair-block, we will assign tasks T'' into the mentioned area such that we require that all small tasks in T'' use e_M and for each large tasks $i \in T''$ we require that $P(i) \subseteq P_{e_M,e_R}$. Due to the former condition, not all points with x-coordinate between e_L and e_M are actually usable for tasks assigned to $\mathcal{SB}$ and the usable ones form a staircase shape (see Fig. 1). Formally, we say that a solution (T'', h'') *fits into a stair-block* $\mathcal{SB} = (e_L, e_M, e_R, f, T'_L, h')$ if $P(i) \subseteq P_{e_M,e_R}$ and $f_e \leq h''(i) \leq u_{e_M} - d_i$ for each $i \in T'' \cap T_L$ and each $e \in P(i)$, $h''(i') \geq u_{e_L}$ and $i' \in T_{e_M}$ for each $i' \in T'' \cap T_S$, and additionally $(T'_L \cup T'', h' \cup h'')$ forms a feasible solution. Also, we require that $h''(i) < d_i$ for each $i \in T'' \cap T_L$ which is a technical condition that we need later in order to be able to compute a profitable stair solution efficiently. A set of tasks T'' fits into a stair-block $\mathcal{SB}$, if there is a function h'' such that the solution (T'', h'') fits into $\mathcal{SB}$. We will need later that the function f is simple and to this end we say that a stair-block $\mathcal{SB} = (e_L, e_M, e_R, f, T'_L, h')$ is a γ-*stair-block* if f is a a step-function with

at most γ steps. Note that it can happen that e_R lies on the left of e_L and then we define P_{e_M,e_R} to be the path that starts with e_R and ends with e_M (one may imagine that Fig. 1 is mirrored).

We seek solutions that consist of stair-blocks and large tasks that are compatible with each other. To this end, for a stair-block $\mathcal{SB} = (e_L, e_M, e_R, f, T'_L, h')$ we define $P(\mathcal{SB})$ to be the path starting with the edge on the right of e_L and ending with e_R. A large task i with height $h(i)$ is *compatible with* $\mathcal{SB}$ if $i \notin T'_L$ and intuitively i does not intersect the area of the stair-block, i.e., if $h(i) \geq u_{e_M}$ or $h(i) + d_i \leq f_e$ for each $e \in P(i) \cap P(\mathcal{SB})$. We say that a task $i \in T_L$ with height $h(i)$ is *part of* $\mathcal{SB}$ if $i \in T'_L$ and $h(i) = h'(i)$. We say that stair-blocks $\mathcal{SB} = (e_L, e_M, e_R, f, T'_L, h')$ and $\overline{\mathcal{SB}} = (\bar{e}_L, \bar{e}_M, \bar{e}_R, \bar{f}, \bar{T}'_L, \bar{h}')$ are *compatible* if for each task $i \in \bar{T}'_L \cap T'_L$ we have $h'(i) = \bar{h}'(i)$, each task $i \in \bar{T}'_L \setminus T'_L$ is compatible with $\mathcal{SB}$, each task $i \in T'_L \setminus \bar{T}'_L$ is compatible with $\overline{\mathcal{SB}}$, and there is no task $i \in T$ that fits into both $\mathcal{SB}$ and $\overline{\mathcal{SB}}$ (for suitable heights $h''(i)$ and $\bar{h}''(i)$). Intuitively, a stair-solution consists of a set of stair-blocks and a set of large tasks T^0_L that are all compatible with each other.

▶ **Definition 7.** *A solution (T'', h'') is a* γ-stair-solution *if there exists a set of γ-stair-blocks $\{\mathcal{SB}_1, \ldots, \mathcal{SB}_k\}$ and partitions $T'' \cap T_L = T^0_L \dot\cup T^1_L \dot\cup \ldots \dot\cup T^k_L$ and $T'' \cap T_S = T^1_S \dot\cup \ldots \dot\cup T^k_S$ such that for each $j \in [k]$, the tasks $T^j_L \cup T^j_S$ fit into $\mathcal{SB}_j$, for any $j, j' \in [|\mathcal{SB}|]$ the stair-blocks SB_j and $SB_{j'}$ are compatible, for each stair-block $\mathcal{SB}_j$ and each task $i \in T^0_L$ with height $h''(i)$, the task i is compatible with $\mathcal{SB}_j$ or part of $\mathcal{SB}_j$, and each edge is contained in the path $P(i)$ of at most γ tasks $i \in T^0_L$ and in the path $P(\mathcal{SB}_j)$ of at most γ stair-blocks $\mathcal{SB}_j$.*

Our main structural lemma is that there exists a boxable solution or a stair solution whose profit is large enough so that we can get an approximation ratio better than 2.

▶ **Lemma 8** (Structural lemma)**.** *There exists a $(\log n/\delta^2)^{O(c+1)}$-boxable solution T_{box} such that $w(T_{\text{box}}) \geq \text{opt}/(1.997 + \varepsilon)$ or there exists a $(\log n/O(\delta))^{O(c+1)}$-stair-solution T_{stair} with $w(T_S \cap T_{\text{stair}}) \geq \frac{1}{\alpha} w(T_L \cap T_{\text{stair}})$ for some value $\alpha \geq 1$ such that $w(T_{\text{stair}} \cap T_L) + \frac{1}{8(\alpha+1)} w(T_{\text{stair}} \cap T_S) \geq \text{opt}/(1.997 + \varepsilon)$.*

If the first case of Lemma 8 applies then the algorithm due to Lemma 6 yields a $(1.997+\varepsilon)$-approximation. In the second case the following algorithm yields a $(1.997+\varepsilon)$-approximation which completes the proof of Theorem 1.

▶ **Lemma 9.** *Let $(T_{\text{stair}}, h_{\text{stair}})$ be a γ-stair solution with $w(T_S \cap T_{\text{stair}}) \geq \frac{1}{\alpha} w(T_L \cap T_{\text{stair}})$ for some value $\alpha \geq 1$. There is an algorithm with running time $(n \cdot \max_e u_e)^{O_\delta(\gamma^2 \log(\max_e u_e))}$ that computes a stair solution (T', h') with $w(T') \geq (1 - O(\varepsilon))(w(T_{\text{stair}} \cap T_L) + \frac{1}{8(\alpha+1)} w(T_{\text{stair}} \cap T_S))$.*

2.1 Uniform edge capacities

Assume now that all edge capacities are identical, i.e., that there exists a value U such that $u_e = U$ for each edge $e \in E$ but that not necessarily $U \leq n^{(\log n)^c}$. For this case we want to design a *polynomial* time $(63/32 + \varepsilon)$-approximation algorithm. The above building blocks are not sufficient since the corresponding algorithms need quasi-polynomial time. Therefore, first we consider special cases of boxable solutions for which we design polynomial time algorithms. We begin with such an algorithm for β-boxable solutions for constant β that intuitively collects all the profit from the large tasks in the optimal β-boxable solution and half of the profit of its small tasks.

▶ **Lemma 10.** *Let $\beta \in \mathbb{N}$ and let $(T_{\text{box}}, h_{\text{box}})$ be a β-boxable solution. There is an algorithm with running time $n^{O(\beta^3/\varepsilon)}$ that computes a solution (T', h') with $w(T') \geq w(T_{\text{box}} \cap T_L) + (1/2 - \varepsilon) w(T_{\text{box}} \cap T_S)$.*

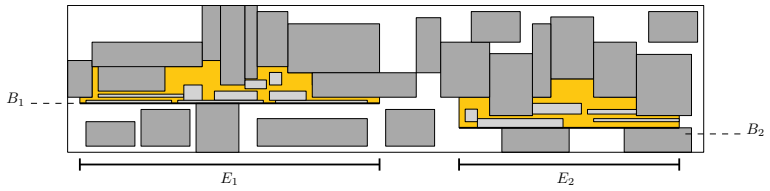

Figure 4 A jammed solution with two subpaths E_1, E_2 corresponding horizontal line segments $E_1 \times B_1$ and $E_1 \times B_2$, and small tasks (light gray) that are jammed in the respective orange areas between the large tasks (dark gray).

Next, we define laminar boxable solutions which are boxable solutions in which the paths of the boxes form a laminar family and the sizes of the boxes are geometrically increasing through the levels (see Fig. 2). A set of boxes $\mathcal{B} = \{B_1, \dots, B_{|\mathcal{B}|}\}$ with a height assignment $h : \mathcal{B} \to \mathbb{N}$ is a *laminar set of boxes* if

- the paths of the boxes form a laminar family, i.e., for any two boxes $B_k, B_{k'}$ we have that $P(B_k) \subseteq P(B_{k'})$, $P(B_{k'}) \subseteq P(B_k)$, or $P(B_k) \cap P(B_{k'}) = \emptyset$,
- there is a box $B^* \in \mathcal{B}$ with $P(B) \subseteq P(B^*)$ for each $B \in \mathcal{B}$,
- for each box $B \in \mathcal{B}$ we have that $d_B = (1+\varepsilon)^k$ for some $k \in \mathbb{N}_0$,
- for each box $B \in \mathcal{B}$ with $d_B = (1+\varepsilon)^k$ for some integer $k \geq 1$ there is a box $B' \in \mathcal{B}$ with $P(B) \subseteq P(B')$, $d_{B'} = (1+\varepsilon)^{k-1}$, and $h(B_k) = h(B_{k-1}) + d_{B_{k-1}}$.

We define $P(\mathcal{B}) := P(B^*)$. A *β-laminar boxable solution* is now a boxable solution whose boxes can be partitioned into sets $\mathcal{B} = \{\mathcal{B}_0, \mathcal{B}_1, \dots, \mathcal{B}_{|\mathcal{B}|-1}\}$ such that the boxes in the sets $\mathcal{B}_1, \dots, \mathcal{B}_{|\mathcal{B}|-1}$ are laminar sets of boxes whose respective paths $P(\mathcal{B}_j)$ are pairwise disjoint and each edge is used by at most β boxes from $\mathcal{B}_0$. Also, each box in $\mathcal{B}_0$ contains exactly one large task and each box in $\mathcal{B}_1, \dots, \mathcal{B}_{|\mathcal{B}|-1}$ contains only small tasks. We design a polynomial time algorithm for finding profitable laminar boxable solutions.

▶ **Lemma 11.** *Let $(T_{\text{lam}}, h_{\text{lam}})$ be a β-laminar boxable solution. There is an algorithm with a running time of $n^{O(\beta+1/\varepsilon^2)}$ that computes a β-laminar boxable solution (T', h') with $w(T') \geq w(T_{\text{lam}} \cap T_L) + w(T_{\text{lam}} \cap T_S)/(2+\varepsilon)$.*

The next class of solutions are pile boxable solutions. A set of boxes $\mathcal{B} = \{B_1, \dots, B_{|\mathcal{B}|}\}$ with a height assignment $h : \mathcal{B} \to \mathbb{N}$ is called a *β-pile of boxes* if $|\mathcal{B}| \leq \beta$, $P(B_k) \supseteq P(B_{k+1})$, $h(B_k) = (k-1)U/|\mathcal{B}|$ and $d_{B_k} = U/|\mathcal{B}|$ for each k. We define $P(\mathcal{B}) := P(B_1)$. A *β-pile boxable solution* is, similarly as above, a boxable solution whose boxes can be partitioned into sets of boxes $\mathcal{B} = \{\mathcal{B}_0, \mathcal{B}_1, \dots, \mathcal{B}_{|\mathcal{B}|-1}\}$ such that the boxes in the sets $\mathcal{B}_1, \dots, \mathcal{B}_{|\mathcal{B}|-1}$ are β-piles of boxes whose respective paths $P(\mathcal{B}_j)$ are pairwise disjoint and each edge is used by at most β boxes in $\mathcal{B}_0$. For β-pile boxable solutions we design a polynomial time algorithm that finds essentially the optimal solution of this type.

▶ **Lemma 12.** *Let $(T_{\text{pile}}, h_{\text{pile}})$ be a β-pile boxable solution. There is an algorithm with a running time of $n^{O(\beta+1/\delta)}$ that computes a β-pile boxable solution (T', h') with $w(T') \geq w(T_{\text{pile}})/(1+\varepsilon)$.*

Finally, we define jammed solutions (which are *not* defined via boxes). Intuitively, they consist of large and small tasks such that the small tasks are placed in areas between some horizontal line segments and the large tasks such that the small tasks are relatively large compared to the free space on each edge in these areas (see Fig. 2 and Fig. 4). Formally, given a solution (T', h') where we define $T'_L := T' \cap T_L$, let $E' \subseteq E$ be a subpath, and let $B \geq 0$ such that intuitively no task $i \in T'_L$ crosses the line segment $E' \times B$, i.e., for each

task $i \in T'_L$ we have that $E' \cap P(i) = \emptyset$ or $h'(i) \geq B$ or $h'(i) + d_i \leq B$. The reader may imagine that we draw the line segment $E' \times B$ in the solution given by the large tasks T'_L and that we are interested in small tasks that are drawn above $E' \times B$. For each edge $e \in E'$ let $u'_e := \min_{i \in T'_L : e \in P(i) \wedge h(i) \geq B} h(i) - B$ and define $u'_e := U - B$ if there is no task $i \in T'_L$ with $e \in P(i)$ and $h(i) \geq B$. A task $i \in T' \cap T_S$ is a δ'-*jammed tasks for* (T'_L, E', B, h') if $P(i) \subseteq E'$, $B \leq h'(i) \leq h'(i) + d_i \leq u'_e$ for each edge $e \in P(i)$, and there exists an edge $e' \in P(i)$ such that $d_i > \delta' u'_{e'}$, i.e., intuitively i is relatively large for the edge capacities u'.

▶ **Definition 13.** *A solution solution* (T', h') *is a* δ'-jammed-solution *if there are pairwise disjoint subpaths* $E_1, \ldots, E_k \subseteq E$*, values* $B_1, \ldots, B_k$*, and a partition* $T'_S := T' \cap T_S = T'_{S,1} \dot\cup \ldots \dot\cup T'_{S,k}$ *such that* $T'_{S,\ell}$ *is a set of* δ'*-jammed tasks for* $(T'_L, E_\ell, B_\ell, h')$ *for each* $\ell \in [k]$ *with* $T'_L := T' \cap T_L$*.*

▶ **Lemma 14.** *Let* $(T_{\text{jam}}, h_{\text{jam}})$ *be a* δ'*-jammed solution. There is an algorithm with a running time of* $n^{O_\varepsilon(1/(\delta \cdot \delta')^3)}$ *that computes a* $O(\delta')$*-jammed solution* (T', h') *with* $w(T') \geq w(T_{\text{lam}})/(1+\varepsilon)$*.*

Our key structural lemma for the case of uniform edge capacities shows that for each instance there exists a solution of one of the above types for which the respective algorithm finds a solution of profit at least $\text{opt}/(63/32 + \varepsilon)$. Then Theorem 2 follows from combining Lemmas 10, 11, 12, 14, and 15.

▶ **Lemma 15** (Structural lemma, uniform capacities)**.** *Given a SAP-instance* (T, E) *where* $u_e = U$ *for each edge* $e \in E$ *and some value* U*. There exists at least one of the following solutions*

- *a* $O_\varepsilon(1)$*-boxable solution* $(T_{\text{box}}, h_{\text{box}})$ *such that* $w(T_{\text{box}} \cap T_L) + w(T_{\text{box}} \cap T_S)/2 \geq \text{OPT}/(63/32 + \varepsilon)$
- *a laminar boxable solution* $(T_{\text{lam}}, h_{\text{lam}})$ *with* $w(T_{\text{lam}} \cap T_L) + w(T_{\text{lam}} \cap T_S)/2 \geq \text{OPT}/(63/32 + \varepsilon)$
- *a* $O_\varepsilon(1)$*-pile boxable solution* $(T_{\text{pile}}, h_{\text{pile}})$ *with* $w(T_{\text{pile}}) \geq \text{OPT}/(63/32 + \varepsilon)$
- *a jammed-solution* $(T_{\text{jam}}, h_{\text{jam}})$ *with* $w(T_{\text{jam}}) \geq \text{OPT}/(63/32 + \varepsilon)$*.*

2.2 Resource augmentation

We consider now again the case of arbitrary edge capacities but under $(1+\eta)$-resource augmentation. First, we show that due to the latter we can reduce the general case to the case of a constant range of edge capacities.

▶ **Lemma 16.** *If there is an* α*-approximation algorithm with a running time of* $n^{O(f(\eta, M))}$ *for the case of* $(1+\eta)$*-resource augmentation where* $\eta < 1$ *and* $u_e \leq M u_{e'}$ *for any two edges* e, e' *then there is an* $\alpha(1+\varepsilon)$*-approximation algorithm with a running time of* $n^{O(f(\eta, 1/(\varepsilon\eta)))}$ *for the case of* $(1 + O(\eta))$*-resource augmentation.*

Next, we show that if we are given an instance with a constant range of edge capacities, under $(1+\eta)$-resource augmentation we can guarantee that there is an $(1+\varepsilon)$-approximative $O_{\varepsilon,\eta}(1)$-boxable solution. Then Theorem 3 follows by combining Lemmas 6, 10, 16, and 17 with the $(1+\varepsilon)$-approximation algorithm for sufficiently small tasks in [27].

▶ **Lemma 17.** *Given an instance where* $u_e \leq M u_{e'}$ *for any two edges* e, e' *with optimal solution* (T^*, h^*)*. If we increase the edge capacities by a factor of* $1+\eta$*, there is a* $O_{\varepsilon,\eta}(1)$*-boxable solution* (T', h') *such that* $w(T') \geq w(T^*)/(1+\varepsilon)$*.*

3 Structural lemma for uniform capacities

In the remaining part of this subsection, we sketch the proof of Lemma 15. Consider an optimal solution (OPT, h). We define $\mathrm{OPT}_L := \mathrm{OPT} \cap T_L$ and $\mathrm{OPT}_S := \mathrm{OPT} \cap T_S$.

Recall that our goal is to improve the approximation ratio of 2. Observe that OPT_L alone is a $1/\delta$-boxable solution and hence if $w(\mathrm{OPT}_L) \geq \frac{1}{(63/32+\varepsilon)}\mathrm{opt}$ then we obtain a $1/\delta$-boxable solution with the desired properties. Similarly, the set OPT_S yields a pile boxable solution with exactly 1 box B with $P(B) = E$ and $d_B = U$. Therefore, if $w(\mathrm{OPT}_S) \geq \frac{1}{(63/32+\varepsilon)}\mathrm{opt}$ then we are done. The reader may imagine that $w(\mathrm{OPT}_S) = w(\mathrm{OPT}_L) = \frac{1}{2}\mathrm{opt}$. We split the small tasks in OPT_S into three sets $\mathrm{OPT}_{S,\mathrm{top}}, \mathrm{OPT}_{S,\mathrm{mid}}, \mathrm{OPT}_{S,\mathrm{bottom}}$. Intuitively, we draw a strip of height δU at the bottom of the capacity profile and a strip of height δU at the top of the capacity profile and assign to $\mathrm{OPT}_{S,\mathrm{top}}$ all tasks in OPT_S whose top edge lies in the top strip, we assign to $\mathrm{OPT}_{S,\mathrm{bottom}}$ all tasks in OPT_S whose bottom edge lies in the bottom strip, and we assign to $\mathrm{OPT}_{S,\mathrm{mid}}$ all other small tasks. Formally, we define $\mathrm{OPT}_{S,\mathrm{top}} := \{i \in \mathrm{OPT}_S \mid h(i) + d_i > (1-\delta)U\}$, $\mathrm{OPT}_{S,\mathrm{bottom}} := \{i \in \mathrm{OPT}_S \mid h(i) < \delta U\}$, and $\mathrm{OPT}_{S,\mathrm{mid}} := \mathrm{OPT}_S \setminus (\mathrm{OPT}_{S,\mathrm{top}} \cup \mathrm{OPT}_{S,\mathrm{bottom}})$.

Small tasks at bottom and large tasks. We define solutions that consists of OPT_L and subsets of $\mathrm{OPT}_{S,\mathrm{bottom}}$. For each edge e let $u'_e := \min_{i \in T'_L : e \in P(i)} h(i)$ and define $u'_e := U$ if there is no task $i \in T'_L$ with $e \in P(i)$. One may think of u' as a pseudo-capacity profile for which $\mathrm{OPT}_{S,\mathrm{bottom}}$ is a feasible solution. We the following lemma and obtain sets $\mathrm{OPT}_{S,\mathrm{bottom},L}$, $\mathrm{OPT}_{S,\mathrm{bottom},S}$ with $w(\mathrm{OPT}_{S,\mathrm{bottom},L} \cup \mathrm{OPT}_{S,\mathrm{bottom},S}) \geq (1-\varepsilon)w(\mathrm{OPT}_{S,\mathrm{bottom}})$.

▶ **Lemma 18.** *Given a solution (T', h') for a SAP instance where $\max_e u_e \leq n^{(\log n)^c}$ for some constant c. Then there are sets $T'_L \subseteq T'$ and $T'_S \subseteq T'$ with $w(T'_L \cup T'_S) \geq (1-O(\varepsilon))w(T')$ and there is an $\eta = O_{\varepsilon,\delta}(1)$ such that*

1. *for each edge e it holds that $|T'_L \cap T_e| \leq (\log n)^{O_{\varepsilon,\delta}(c)}$, and*
2. *for each task $i \in T'_L$ there is an edge $e \in P(i)$ with $d_i \geq \eta u_e$,*
3. *there is a boxable solution for T'_S in which each edge e is used by at most $(\log n)^{O_\varepsilon(c)}$ boxes,*
4. *these boxes form groups of laminar sets of boxes.*

We have that for each task $i \in \mathrm{OPT}_{S,\mathrm{bottom},L}$ there is an edge $e \in P(i)$ with $d_i > \delta u'_e$, therefore $\mathrm{OPT}_{S,\mathrm{bottom},L}$ is a set of δ-jammed tasks for $(\mathrm{OPT}_L, E, 0, h)$ and hence $\mathrm{OPT}_L \cup \mathrm{OPT}_{S,\mathrm{bottom},L}$ forms a δ-jammed-solution. Also, $\mathrm{OPT}_L \cup \mathrm{OPT}_{S,\mathrm{bottom},S}$ forms a laminar boxable solution. Hence, if $w(\mathrm{OPT}_{S,\mathrm{bottom},S})$ or $w(\mathrm{OPT}_{S,\mathrm{bottom},L})$ is sufficiently large then we are done. Therefore, the reader may imagine that $w(\mathrm{OPT}_{S,\mathrm{bottom},S}) = w(\mathrm{OPT}_{S,\mathrm{bottom},L}) = 0$.

Small tasks at top and large tasks. We mirror OPT along the y-axis and do a symmetric construction with $\mathrm{OPT}_{S,\mathrm{top}}$: we apply Lemma 18 which yields sets $\mathrm{OPT}_{S,\mathrm{top},L}, \mathrm{OPT}_{S,\mathrm{top},S}$ and a δ-jammed solution $\mathrm{OPT}_L \cup \mathrm{OPT}_{S,\mathrm{top},L}$ and a laminar boxable solution $\mathrm{OPT}_L \cup \mathrm{OPT}_{S,\mathrm{top},S}$. Like before, the reader may imagine that $w(\mathrm{OPT}_{S,\mathrm{top},S}) = w(\mathrm{OPT}_{S,\mathrm{top},L}) = 0$ and hence $w(\mathrm{OPT}_{S,\mathrm{mid}}) = w(\mathrm{OPT}_S) = \frac{1}{2}\mathrm{opt}$.

Small and large tasks in the middle. Next, we split the large tasks OPT_L into three sets $\mathrm{OPT}_{L,\mathrm{top}}$, $\mathrm{OPT}_{L,\mathrm{mid}}$, and $\mathrm{OPT}_{L,\mathrm{bottom}}$. Intuitively, $\mathrm{OPT}_{L,\mathrm{top}}$ consists of all tasks in OPT_L whose top edge lies in the strip of height δU at the top of the capacity profile, $\mathrm{OPT}_{L,\mathrm{bottom}}$ consists of all tasks in OPT_L whose bottom edge lies in the strip of height δU at the bottom

of the capacity profile, and $\mathrm{OPT}_{L,\mathrm{mid}}$ contains all remaining tasks in OPT_L. Formally, $\mathrm{OPT}_{L,\mathrm{top}} := \{i \in \mathrm{OPT}_L | h(i) + d_i > (1-\delta)U\}$, $\mathrm{OPT}_{L,\mathrm{bottom}} := \{i \in \mathrm{OPT}_L | h(i) < \delta U\}$, and $\mathrm{OPT}_{L,\mathrm{mid}} := \mathrm{OPT}_L \setminus (\mathrm{OPT}_{L,\mathrm{top}} \cup \mathrm{OPT}_{L,\mathrm{bottom}})$.

Observe that no task in $\mathrm{OPT} \setminus (\mathrm{OPT}_{L,\mathrm{bottom}} \cup \mathrm{OPT}_{S,\mathrm{bottom}})$ touches the rectangle $E \times [0, \delta U)$. Using this, we define a boxable solution T_{box} that will consist essentially of all tasks in the set $\mathrm{OPT} \setminus (\mathrm{OPT}_{L,\mathrm{bottom}} \cup \mathrm{OPT}_{S,\mathrm{bottom}})$. Intuitively, we will use the free space $E \times [0, \delta U)$ in order to untangle the interaction between the large and small tasks. More precisely, T_{box} will be a $O_\delta(1)$-boxable solution in which all small tasks are assigned into boxes of height $\Theta_\delta(U)$ each. More formally, we apply the following lemma to $\mathrm{OPT} \setminus (\mathrm{OPT}_{L,\mathrm{bottom}} \cup \mathrm{OPT}_{S,\mathrm{bottom}})$ and denote by $(T'_{\mathrm{box}}, h'_{\mathrm{box}})$ the resulting solution.

▶ **Lemma 19.** *Given a solution (T', h') such that no task touches the rectangle $E \times [0, \delta U)$. Then there exists a $O_\delta(1)$-boxable solution (T', h_{box}).*

We apply Lemma 19 to the solution obtained by taking $\mathrm{OPT} \setminus (\mathrm{OPT}_{L,\mathrm{top}} \cup \mathrm{OPT}_{S,\mathrm{top}})$ and shifting each task up by δU units. Let $(T''_{\mathrm{box}}, h''_{\mathrm{box}})$ denote the resulting solution. Assume w.l.o.g. that $w(\mathrm{OPT}_{L,\mathrm{top}}) \leq w(\mathrm{OPT}_{L,\mathrm{bottom}})$. Observe that if $w(\mathrm{OPT}_{L,\mathrm{mid}}) \geq \gamma \mathrm{opt}$ then

$$\begin{aligned} w(T' \cap T_L) &\geq w(\mathrm{OPT}_{L,\mathrm{bottom}}) + w(\mathrm{OPT}_{L,\mathrm{mid}}) \\ &\geq \frac{1}{2} w(\mathrm{OPT}_{L,\mathrm{top}} \cup \mathrm{OPT}_{L,\mathrm{bottom}}) + w(\mathrm{OPT}_{L,\mathrm{mid}}) \\ &\geq \frac{1+\gamma}{2} w(\mathrm{OPT}_L) \end{aligned}$$

and $w(T' \cap T_S) \geq w(\mathrm{OPT}_S \setminus \mathrm{OPT}_{S,\mathrm{bottom}})$. Therefore, the reader may imagine now that $w(\mathrm{OPT}_{L,\mathrm{mid}}) = 0$.

Types of points in the middle

We distinguish points in the rectangle $E \times [0, U]$ into different types and identify each task $i \in \mathrm{OPT}$ with a rectangle $R_i := P(i) \times [h(i), h(i) + d_i]$. For each point p let ℓ_p denote the maximally long horizontal line segment in $E \times [0, U]$ that contains p and that does not touch the relative interior of a task in $\mathrm{OPT}_{L,\mathrm{top}} \cup \mathrm{OPT}_{L,\mathrm{bottom}}$. We say that p is a *top-point* if no endpoint of ℓ_p touches a task in $\mathrm{OPT}_{L,\mathrm{bottom}}$, p is a *sandwich-point* if one of the end-points of ℓ_p touches a task in $\mathrm{OPT}_{L,\mathrm{top}}$ and the other end-point touches task in $\mathrm{OPT}_{L,\mathrm{bottom}}$, and p is a *bottom-point* if no endpoint of ℓ_p touches a task in $\mathrm{OPT}_{L,\mathrm{top}}$ and at least one endpoint of ℓ_p touches a task in $\mathrm{OPT}_{L,\mathrm{bottom}}$. Note that here we do not define stair-points since the edges have uniform capacities. Let $\mathcal{C}_{\mathrm{top}}, \mathcal{C}_{\mathrm{sw}}, \mathcal{C}_{\mathrm{bottom}}$ denote the set of connected components of top- sandwich-, and bottom-points, respectively.

Each edge is used by at most three connected components in $\mathcal{C}_{\mathrm{top}} \cup \mathcal{C}_{\mathrm{sw}} \cup \mathcal{C}_{\mathrm{bottom}}$.

▶ **Lemma 20.** *Each edge e can be used by at most one connected component of top-points, by at most one connected component of sandwich-points, and by at most one connected component of bottom-points.*

Let $\mathrm{OPT}_{S,\mathrm{cross}} \subseteq \mathrm{OPT}_{S,\mathrm{mid}}$ denote the tasks in $\mathrm{OPT}_{S,\mathrm{mid}}$ that intersect at least two different connected components, e.g., a connected component of top-points and a connected component of sandwich-points.

▶ **Lemma 21.** *Each edge is used by at most 2 different tasks in $\mathrm{OPT}_{S,\mathrm{cross}}$.*

In particular, $\mathrm{OPT}_L \cup \mathrm{OPT}_{S,\mathrm{cross}}$ forms a $O(1/\delta)$-boxable solution. If $w(\mathrm{OPT}_{S,\mathrm{cross}})$ is sufficiently large we are therefore done. The reader may imagine that $w(\mathrm{OPT}_{S,\mathrm{cross}}) = 0$.

Top points. Consider a connected component C of top-points. Let $\mathrm{OPT}_{S,\mathrm{mid},\mathrm{top}}(C) \subseteq \mathrm{OPT}_{S,\mathrm{mid}}$ denote the tasks in $\mathrm{OPT}_{S,\mathrm{mid}}$ contained in C. We apply Lemma 18 and obtain the sets $\mathrm{OPT}_{S,\mathrm{mid},\mathrm{top},S}(C)$ and $\mathrm{OPT}_{S,\mathrm{mid},\mathrm{top},L}(C)$. Let

$$\mathrm{OPT}_{S,\mathrm{mid},\mathrm{top},S} := \bigcup_{C \in \mathcal{C}_{\mathrm{top}}} \mathrm{OPT}_{S,\mathrm{mid},\mathrm{top},S}(C)$$

and

$$\mathrm{OPT}_{S,\mathrm{mid},\mathrm{top},L} := \bigcup_{C \in \mathcal{C}_{\mathrm{top}}} \mathrm{OPT}_{S,\mathrm{mid},\mathrm{top},L}(C).$$

We obtain that $\mathrm{OPT}_{L,\mathrm{top}} \cup \mathrm{OPT}_{L,\mathrm{bottom}} \cup \mathrm{OPT}_{S,\mathrm{mid},\mathrm{top},L}$ is a δ-jammed solution and $\mathrm{OPT}_{L,\mathrm{top}} \cup \mathrm{OPT}_{L,\mathrm{bottom}} \cup \mathrm{OPT}_{S,\mathrm{mid},\mathrm{top},S}$ is a laminar boxable solution. Intuitively, if $w(\mathrm{OPT}_{S,\mathrm{mid},\mathrm{top},L})$ or $w(\mathrm{OPT}_{S,\mathrm{mid},\mathrm{top},S})$ is sufficiently large then we are done. The reader may therefore imagine that both quantities are zero.

Bottom points. We do a symmetric operation for all connected components C of bottom-points, obtaining respective sets $\mathrm{OPT}_{S,\mathrm{mid},\mathrm{bottom},S}, \mathrm{OPT}_{S,\mathrm{mid},\mathrm{bottom},L}$, a δ-jammed solution

$$\mathrm{OPT}_{L,\mathrm{top}} \cup \mathrm{OPT}_{L,\mathrm{bottom}} \cup \mathrm{OPT}_{S,\mathrm{mid},\mathrm{bottom},L},$$

and a laminar boxable solution

$$\mathrm{OPT}_{L,\mathrm{top}} \cup \mathrm{OPT}_{L,\mathrm{bottom}} \cup \mathrm{OPT}_{S,\mathrm{mid},\mathrm{bottom},S}.$$

Like before, the reader may imagine that $w(\mathrm{OPT}_{S,\mathrm{mid},\mathrm{bottom},S}) = w(\mathrm{OPT}_{S,\mathrm{mid},\mathrm{bottom},L}) = 0$.

Sandwich points. Finally, let $\mathrm{OPT}_{S,\mathrm{mid},\mathrm{sw}}$ denote all points in $\mathrm{OPT}_{S,\mathrm{mid}}$ that are contained in a connected component in $\mathcal{C}_{\mathrm{sw}}$. We assume first that $w(\mathrm{OPT}_{L,\mathrm{top}}) \geq w(\mathrm{OPT}_{L,\mathrm{bottom}})$. We define a solution consisting of some tasks in $\mathrm{OPT}_{S,\mathrm{mid},\mathrm{sw}}$ and additionally all tasks in $\mathrm{OPT}_{L,\mathrm{top}}$. Let $C \in \mathcal{C}_{\mathrm{sw}}$ and denote by $\mathrm{OPT}_{S,\mathrm{mid}}(C)$ the set of tasks in $\mathrm{OPT}_{S,\mathrm{mid}}$ that are contained in C. Let E' denote the maximally long subpath between two vertices v, v' such that for each $x \in [v, v']$ the vertical line through x, i.e., $\{x\} \times \mathbb{R}$, has non-empty intersection with C. Note that the rectangle $E' \times [0, \delta U]$ has empty intersection with each tasks in $\mathrm{OPT}_{L,\mathrm{top}} \cup \mathrm{OPT}_{S,\mathrm{mid}}$. Intuitively, we use this free space in order to push all tasks in $\mathrm{OPT}_{S,\mathrm{mid}}(C)$ down by δU units. Then they all fit into $O_\varepsilon(1)$ boxes that have non-empty intersection with the tasks in $\mathrm{OPT}_{L,\mathrm{top}}$.

▶ **Lemma 22.** *Given $C \in \mathcal{C}_{\mathrm{sw}}$. There is a pile of boxes $\mathcal{B} = \{B_1, \ldots, B_{1/\delta}\}$ (with a height assignment $h\colon \mathcal{B} \to \mathbb{N}_0$) such that $P(B) \subseteq E'$ and there are pairwise disjoint sets $T_1, \ldots, T_{|\mathcal{B}|} \subseteq \mathrm{OPT}_{S,\mathrm{mid}}(C)$ such that for each j, the tasks in T_j fit into B_j. The weight of tasks in the boxes is at least $\sum_j w(T_j) \geq (1-\varepsilon) w(\mathrm{OPT}_{S,\mathrm{mid}}(C))$.*

Let $\mathrm{OPT}_{S,\mathrm{mid},\mathrm{sw}} := \bigcup_{C \in \mathcal{C}_{\mathrm{sw}}} \mathrm{OPT}_{S,\mathrm{mid},S}(C)$. We apply Lemma 22 to each component $C \in \mathcal{C}_{\mathrm{sw}}$ and hence we obtain a pile boxable solution whose profit is at least

$$(1-\varepsilon) w\left(\mathrm{OPT}_{L,\mathrm{top}} \cup \mathrm{OPT}_{S,\mathrm{mid},\mathrm{sw}}\right).$$

In a similar way we can construct a pile boxable solution of profit at least

$$(1-\varepsilon) w\left(\mathrm{OPT}_{L,\mathrm{bottom}} \cup \mathrm{OPT}_{S,\mathrm{mid},\mathrm{sw}}\right).$$

▶ **Lemma 23.** *There is a pile boxable solution* $(T'_{\text{sw}}, h'_{\text{sw}})$ *with profit at least*

$$(1-\varepsilon)w\left(\text{OPT}_{L,\text{top}} \cup \text{OPT}_{S,\text{mid,sw}} \cup \text{OPT}_{S,\text{cross}}\right)$$

and a pile boxable solution $(T''_{\text{sw}}, h''_{\text{sw}})$ *with profit at least*

$$(1-\varepsilon)w\left(\text{OPT}_{L,\text{bottom}} \cup \text{OPT}_{S,\text{mid,sw}} \cup \text{OPT}_{S,\text{cross}}\right).$$

Intuitively, since $w(\text{OPT}_{L,\text{mid}}) = 0$ we have $w(\text{OPT}_{L,\text{top}}) \geq \text{opt}/4$ or $w(\text{OPT}_{L,\text{bottom}}) \geq \text{opt}/4$. Also, since $w(\text{OPT}_{S,\text{mid,top}}) = w(\text{OPT}_{S,\text{mid,bottom}}) = w(\text{OPT}_{S,\text{cross}}) = 0$ and $w(\text{OPT}_{S,\text{mid}}) = \text{opt}/2$ we have that $w(\text{OPT}_{S,\text{mid,sw}}) = \text{opt}/2$. Hence, (T'_{sw}, h') or (T''_{sw}, h'') satisfies the claim of the lemma. Formally, our candidate solutions are

$$\begin{aligned}
\text{OPT}^{(1)} &:= \text{OPT}_S\,,\\
\text{OPT}^{(2)} &:= \text{OPT}_L \cup \text{OPT}_{S,\text{bottom},S}\,,\\
\text{OPT}^{(3)} &:= \text{OPT}_L \cup \text{OPT}_{S,\text{bottom},L}\,,\\
\text{OPT}^{(4)} &:= \text{OPT}_L \cup \text{OPT}_{S,\text{top},S}\\
\text{OPT}^{(5)} &:= \text{OPT}_L \cup \text{OPT}_{S,\text{top},L}\,,\\
\text{OPT}^{(6)} &:= T'_{\text{box}}\,,\\
\text{OPT}^{(7)} &:= T''_{\text{box}}\,,\\
\text{OPT}^{(8)} &:= \text{OPT}_L \cup \text{OPT}_{S,\text{cross}}\,,\\
\text{OPT}^{(7)} &:= \text{OPT}_{L,\text{top}} \cup \text{OPT}_{L,\text{bottom}} \cup \text{OPT}_{S,\text{mid,top},S}\,,\\
\text{OPT}^{(8)} &:= \text{OPT}_{L,\text{top}} \cup \text{OPT}_{L,\text{bottom}} \cup \text{OPT}_{S,\text{mid,top},L}\,,\\
\text{OPT}^{(9)} &:= \text{OPT}_{L,\text{top}} \cup \text{OPT}_{L,\text{bottom}} \cup \text{OPT}_{S,\text{mid,bottom},S}\,,\\
\text{OPT}^{(10)} &:= \text{OPT}_{L,\text{top}} \cup \text{OPT}_{L,\text{bottom}} \cup \text{OPT}_{S,\text{mid,bottom},L}\,,\\
\text{OPT}^{(11)} &:= T'_{\text{sw}}\,, \text{ and}\\
\text{OPT}^{(12)} &:= T''_{\text{sw}}\,.
\end{aligned}$$

The sets $\text{OPT}^{(1)}, \text{OPT}^{(9)}$, and $\text{OPT}^{(10)}$ are pile boxable solutions, $\text{OPT}^{(2)}, \text{OPT}^{(4)}, \text{OPT}^{(7)}$, and $\text{OPT}^{(9)}$ are laminar boxable solutions, $\text{OPT}^{(3)}, \text{OPT}^{(5)}, \text{OPT}^{(8)}$, and $\text{OPT}^{(10)}$ are δ-jammed solutions, and finally $\text{OPT}^{(7)}$ and $\text{OPT}^{(8)}$ are $O_\delta(1)$-boxable solutions.

Our insights above determine constraints of a linear program which provides an upper bound on the approximation ratio. Using LP duality, we are able to prove that the obtained value is at most $63/32$, which completes the proof of Lemma 15.

4 Structural lemma for arbitrary capacities

In this section we prove Lemma 8. We first limit the number of large tasks per edge that can appear in a feasible solution.

▶ **Lemma 24.** *For each edge* e *and each feasible solution* (T_{SOL}, h_{SOL}) *it holds that* $|T_{SOL} \cap T_L \cap T_e| \leq (\log u_e)/\delta^2$.

Consider an optimal solution (OPT, h). Define $\text{OPT}_L := \text{OPT} \cap T_L$ and $\text{OPT}_S := \text{OPT} \cap T_S$. Since we assumed the maximum edge capacity to be quasi-polynomially bounded, Lemma 24 shows that each edge can be used by at most $1/\delta^2 (\log n)^{O(1)}$ large tasks in OPT.

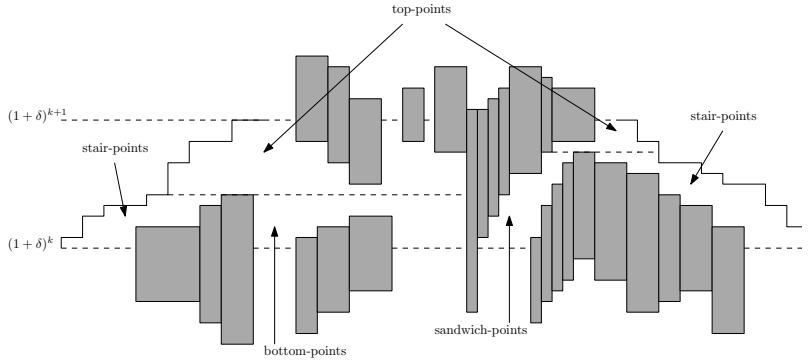

Figure 5 The different types of points within a corridor C_k. Note that the small tasks are not shown in the figure.

Therefore, the tasks in OPT_L alone form a boxable solution. Since our goal is to improve the approximation ratio of 2 in [27], we are done if $w(\mathrm{OPT}_L) \geq (\frac{1}{2}+\gamma)\mathrm{OPT}$ for some $\gamma > 0$. The reader may therefore imagine that $w(\mathrm{OPT}_L) \leq \mathrm{OPT}/2$ and hence that $w(\mathrm{OPT}_S) \geq \mathrm{OPT}/2$.

We partition the large tasks OPT_L into two groups. We define $\mathrm{OPT}_{L,\downarrow} := \{i \in \mathrm{OPT}_L \mid h(i) < d_i\}$ and $\mathrm{OPT}_{L,\uparrow} := \{i \in \mathrm{OPT}_L \mid h(i) \geq d_i\}$. In the next lemma we show that there is a boxable solution that contains essentially all tasks in $\mathrm{OPT}_S \cup \mathrm{OPT}_{L,\uparrow}$.

▶ **Lemma 25.** *For an arbitrary $0 < \varepsilon \leq 1/3$ there exists a boxable solution with profit at least $(1-\varepsilon)w(\mathrm{OPT}_S \cup \mathrm{OPT}_{L,\uparrow})$.*

Intuitively, if now $w(\mathrm{OPT}_{L,\uparrow}) \geq \gamma\mathrm{OPT}$ for some $\gamma > 0$ then $w(\mathrm{OPT}_S \cup \mathrm{OPT}_{L,\uparrow}) \geq (\frac{1}{2}+\gamma)\mathrm{OPT}$ and we are done, due to Lemma 6 and Lemma 25. The reader therefore may imagine that $w(\mathrm{OPT}_{L,\uparrow}) = 0$ and $w(\mathrm{OPT}_S) = \mathrm{OPT}/2$ and hence also $w(\mathrm{OPT}_{L,\downarrow}) = \mathrm{OPT}/2$.

Next, we define solutions that either consist of $\mathrm{OPT}_{L,\downarrow}$ or of a subset of $\mathrm{OPT}_{L,\downarrow}$ and additionally some small tasks. We will prove that one of the constructed sets or $\mathrm{OPT}_S \cup \mathrm{OPT}_{L,\uparrow}$ has large profit. In the sequel, we will identify the vertices $\{v_1, \ldots, v_{|V|}\}$ of (V, E) with the coordinates $1, \ldots, |V|$ and a path P between vertices $v_i, v_{i'}$ with the closed interval $[i, i']$. For each task $i \in \mathrm{OPT}$ define its rectangle $R_i := P(i) \times [h(i), h(i) + d_i]$. We will identify a task i with its rectangle R_i.

We define $(\log n)^{O_\delta(1)}$ corridors. We draw a horizontal line $\ell^{(k)}$ with y-coordinate $y = (1+\delta)^k$ for each $k \in \mathbb{N}$. For each $k \in \mathbb{N}$ we define the area $\mathbb{R} \times [\ell^{(k)}, \ell^{(k+1)})$ to be the *corridor* C_k. Consider a task $i \in \mathrm{OPT}_L$. Observe that for each task $i \in \mathrm{OPT}_L$ the rectangle R_i has to be intersected by at least one line $\ell^{(k)}$ since $d_i > \delta \cdot b(i)$. Also, observe that for each edge e and each corridor C_k there can be at most two tasks $i, i' \in \mathrm{OPT}_L$ whose respective paths $P(i), P(i')$ use e and whose respective rectangles $R_i, R_{i'}$ intersect C_k. If there are two such task i, i' then for one of them its rectangle must intersect $\ell^{(k)}$ and for the other its rectangle must intersect $\ell^{(k+1)}$.

Let C_k be a corridor. For a task $i \in \mathrm{OPT}_L$ with $R_i \cap C_k \neq \emptyset$ we say that i is a *top-large-task for C_k* if $h(i) \in [(1+\delta)^k, (1+\delta)^{k+1})$, a *bottom-large-task for C_k* if $h(i) + d_i \in [(1+\delta)^k, (1+\delta)^{k+1})$, and a *cross-large-task for C_k* if $h(i) < (1+\delta)^k$ and $h(i) + d_i \geq (1+\delta)^{k+1}$. We partition the area of C_k that is not used by large tasks into connected components of points. For each point p, let ℓ_p denote the maximally long horizontal line segment that contains p and that neither crosses a large task nor the capacity profile. We say that p is a *top-point* if each endpoint of ℓ_p touches a top-large-task or the capacity profile; p is a *sandwich-point* if one of the end-points of ℓ_p touches a top-large-task and the other end-point

touches a bottom-large-task; p is a *stair-point* if one end-point of ℓ_p touches a bottom-large-task and the other end-point touches the capacity profile; and p is a *bottom-point*, if both end-points of ℓ_p touch a bottom-large-task, see Fig. 5. In each corridor C_k this yields connected components of top-, bottom-, sandwich-, and stair-points. In the remaining proof (see [28]) we show that for each of these types there exists a $(\log n/\delta)^{O_\varepsilon(c)}$-boxable solution or a $(\log n/\delta)^{O_\varepsilon(c)}$-stair solution that contains all tasks in $\mathrm{OPT}_{L,\downarrow}$ and a constant fraction of the small tasks in all connected components of the respective type, or of the small tasks that overlap more than one type of points. Hence, we obtain an approximation ratio strictly better than 2, unless essentially all small tasks lie in connected components of sandwich points. For this case, intuitively we prove that there is a $(\log n/\delta)^{O_\varepsilon(c)}$-boxable solution that contains all profit from the small tasks and a 1/4-fraction of the profit of the tasks in $\mathrm{OPT}_{L,\downarrow}$. We show that the best of our solutions yields an approximation ratio of $1.997+\varepsilon$ which proves Lemma 8.

5 Compute stair solution

In this section we prove Lemma 9. To this end, we first show how to compute a solution for a single stair-block $\mathcal{SB}$. Then we devise a dynamic program that intuitively sweeps the path from left to right, guesses the stair-blocks and the other large tasks, and then uses the subroutine for a single stair-block to assign tasks into each stair-block. Suppose that we are given a γ-stair-block $\mathcal{SB} = (e_L, e_M, e_R, f, T'_L, h')$. Let $\bar{T}$ denote the set of tasks $i \in T$ such that $\{i\}$ fits into $\mathcal{SB}$. In the sequel, we prove the following lemma.

▶ **Lemma 26.** *Let $T^* \subseteq \bar{T}$ be an unknown set of tasks that fits into $\mathcal{SB}$ such that $w(T_S \cap T^*) \geq \frac{1}{\alpha} w(T_L \cap T^*)$ for some value $\alpha \geq 1$. There is a $(n \cdot \max_e u_e)^{O_\delta(\log(\max_e u_e))}$ time algorithm that computes a set $\hat{T}$ that fits into $\mathcal{SB}$ such that*

$$w(\hat{T}) \geq (1 - O(\varepsilon))) \left(w(T_L \cap T^*) + \frac{1}{8(\alpha+1)} \cdot w(T_S \cap T^*) \right).$$

First, we guess $w(T_L \cap T^*)$ up to a factor $1+\varepsilon$, i.e., we guess a value W such that $w(T_L \cap T^*) \in [W, (1+\varepsilon)W)$. One can show that $(n/\varepsilon)^{O(1)}$ many guesses for W suffice. Our algorithm is based on a linear program that uses configurations for the sets of large tasks that fit into $\mathcal{SB}$. Formally, we define a pair $C = (\bar{T}', \bar{h}')$ to be a *configuration* if $\bar{T}' \subseteq \bar{T}$, $w(\bar{T}') \in [W, (1+\varepsilon)W)$, $\bar{h}'$ is a function $\bar{h}' : \bar{T}' \to \mathbb{N}$ such that $\bar{h}'(i) < d_i$ for each task $i \in \bar{T}'$, and $(\bar{T}', \bar{h}')$ fits into $\mathcal{SB}$. Let $\mathcal{C}$ denote the set of all configurations. We introduce a variable y_C for each configuration $C \in \mathcal{C}$. Intuitively, $y_C = 1$ indicates that the computed solution contains exactly the set of large tasks in C, each of them drawn at the height level determined by C. For each small task $j \in \bar{T}_S := \bar{T} \cap T_S$ and each $t \in \{0, \ldots, b(j) - d_j\}$ we introduce a variable $x_{j,t}$ indicating whether j is contained in the solution and drawn at height t. Note that we do not need variables $x_{j,t}$ for $t > b(j) - d_j$ since the upper edge of j has to have a height of at most $b(j)$.

We add constraints that ensure that the rectangles corresponding to the selected tasks do not overlap. To this end, for each small tasks $j \in \bar{T}_S$ and each possible height $t \in \{0, \ldots, b(j) - d_j\}$ we define a "rectangle" $p(j,t) = \{(e,t') \mid e \in P(j) \text{ and } t \leq t' < t + d_j\}$. For a pair (e,t) the reader may imagine that it represents the point whose x-coordinate is the mid-point of the edge e and whose y-coordinate is t. Similarly, for a configuration $C = (\bar{T}', \bar{h}') \in \mathcal{C}$ we define the "points" covered by C to be $p(C) := \{(e,t') \mid \exists i \in \bar{T}' : e \in P(i) \text{ and } \bar{h}'(i) \leq t' < \bar{h}'(i) + d_i\}$ and $w_C := w(\bar{T}')$.

Denote by $\mathrm{LP}_{\mathcal{SB}}$ the linear program below where for convenience we assume that all non-existing variables are set to zero.

$$\max \sum_{C\in\mathcal{C}} y_C w_C + \sum_{j\in\bar{T}_S,t} x_{j,t} w_j \tag{1}$$

$$\text{s.t.} \quad \sum_{C\colon (e,t)\in p(C)} y_C \tag{2}$$

$$+ \sum_{j\in\bar{T}_S,t'\colon (e,t)\in p(j,t')} x_{j,t'} \leq 1 \quad \text{for all } e\in P_{e_M,e_R}, t\geq 0 \tag{3}$$

$$\sum_{C\colon (e,t)\in p(C)} y_C + \sum_{t'\colon t'\leq t} x_{j,t'} \leq 1 \quad \text{for all } e\in P_{e_M,e_R}, t\geq 0, j\in\bar{T}_S\cap T_e \tag{4}$$

$$\sum_{C\in\mathcal{C}} y_C = 1 \tag{5}$$

$$\sum_{t\geq 0} x_{j,t} \leq 1 \quad \text{for all } j\in\bar{T}_S \tag{6}$$

$$x_{j,t}, y_C \geq 0 \quad \text{for all } j\in\bar{T}_S, t\in\mathbb{N}, t\leq b(j)-d_j, C\in\mathcal{C}$$

The first set of constraints (3) expresses intuitively that no two rectangles overlap. Then (4) strengthens this condition by stating that if a configuration C covers a point (e,t) then no small task j using e can be selected such that it covers a point (e,t') with $t'\leq t$ (note that if C covers (e,t) then it also covers each point (e,t') with $t'\leq t$). Constraint (5) ensures that we select exactly one configuration. A task j still cannot be drawn at two positions simultaneously, which we ensure with (6). In the LP above, we introduce constraints (3) and (4) for each $t\geq 0$, however, it is sufficient to state those for each t such that $t\in\mathbb{N}$ since $d_i\in\mathbb{N}$ for each task $i\in T$ and we introduce the variables $x_{j,t}$ only for values t with $t\in\mathbb{N}$. This yields an equivalent formulation with only $(n\max_e u_e)^{O(1)}$ constraints (apart from the non-negativity constraints). The number of variables in $\mathrm{LP}_{\mathcal{SB}}$ is exponential. However, we can solve it in polynomial time via a suitable separation oracle for the dual.

▶ **Lemma 27.** *There is an algorithm with running time $(n\max_e u_e)^{O_\delta(\log(\max_e u_e))}$ that computes an optimal solution to $LP_{\mathcal{SB}}$.*

5.1 The rounding algorithm

Let (x^*,y^*) denote the optimal solution to $\mathrm{LP}_{\mathcal{SB}}$. We round (x^*,y^*) via randomized rounding. First, we sample a configuration $\hat{C}$ using the distribution determined by y^*, i.e., for each configuration $C\in\mathcal{C}$, we obtain $\hat{C}=C$ with probability y_C. Define $\bar{y}_{\hat{C}}:=1$ and $\bar{y}_C:=0$ for each $C\in\mathcal{C}\setminus\{\hat{C}\}$. Then we construct a new solution x for the small tasks where intuitively we remove all pairs (j,t) that overlap with $\hat{C}$, i.e., such that $p(j,t)\cap p(\hat{C})\neq\emptyset$. For each pair of the latter type we define $x_{j,t}:=0$ and we define $x_{j,t}:=x^*_{j,t}$ for all other pairs (j,t). Observe that x is a solution to the LP that is obtained by taking $\mathrm{LP}_{\mathcal{SB}}$ and removing all variables y_C and constraint (5). Denote by $\mathrm{LP}'_{\mathcal{SB}}$ the resulting LP. We can round it via randomized rounding with alteration, using that for two pairs $(j,t),(j',t')$ with $j,j'\in T_S$ the corresponding rectangles $p(j,t),p(j',t')$ overlap if and only if they overlap on a "vertical line segment above e_M", i.e., on $\cup_{t''\in[t,t+d_j)\cap[t',t'+d_{j'})}(e_M,t'')$.

▶ **Lemma 28.** *Given a solution x to $LP'_{\mathcal{SB}}$. In polynomial time we can compute an integral solution $\bar{x}$ to $LP'_{\mathcal{SB}}$ with expected value $\sum_{j\in\bar{T}_S,t}\bar{x}_{j,t}w_j \geq \frac{1}{4}\sum_{j\in\bar{T}_S,t} x_{j,t}w_j$ such that the support of $\bar{x}$ is contained in the support of x.*

Let $(\bar{x},\bar{y})$ denote the resulting solution. Secondly, we compute the optimal solution to $\mathrm{LP}'_{\mathcal{SB}}$ (hence ignoring the configurations of large tasks) and round it via Lemma 28, let $(\bar{x}',\bar{y}')$ denote the resulting solution. In the sequel we prove that the most profitable solution among $(\bar{x},\bar{y})$ and $(\bar{x}',\bar{y}')$ satisfies the claim of Lemma 9. Since we sampled $\hat{C}$ according to

the distribution given by y^*, we have that $\mathbb{E}[w_{\hat{C}}] = \sum_{C\in\mathcal{C}} y^*_C w_C$. Recall that we discarded all pairs (j,t) such that $p(j,t)$ overlaps with $p(\hat{C})$. Hence, there are some pairs (j,t) that are discarded with very high probability. We call such a pair problematic where formally we say that a pair (j,t) with $j \in \bar{T}_S$ and $t \in \mathbb{N}$ is *problematic* if $\sum_{C\in\mathcal{C}:p(C)\cap p(j,t)\neq\emptyset} y^*_C > 1-\eta$ for some value $\eta > 0$ to be defined later. Let $T_{S!}$ denote the set of all problematic pairs. In the following lemma we prove that their contribution to the profit of (x^*, y^*) is only small and hence we can afford to ignore them, unless $(\bar{x}', \bar{y}')$ already has enough profit. Here we crucially need constraint (4). We define $\mathrm{opt}_{LP} := \sum_{C\in\mathcal{C}} y^*_C w_C + \sum_{j\in\bar{T}_S,t} x^*_{j,t} w_j$.

▶ **Lemma 29.** *We have that $\sum_{(j,t)\in T_{S!}} x^*_{j,t} w_j \leq 4\eta \mathrm{opt}_{LP}$ or the profit of $(\bar{x}', \bar{y}')$ is at least $\mathrm{opt}_{LP} \geq w(T_L \cap T^*) + w(T_S \cap T^*)$.*

Assume now that the first case of Lemma 29 applies. We argue that then the problematic pairs contribute at most half of the profit of all pairs (for all small tasks) and hence we can ignore the problematic pairs.

▶ **Lemma 30.** *Assume that $\eta \leq \frac{1}{8}\frac{1/\alpha - O(\varepsilon)}{1+1/\alpha-O(\varepsilon)}$. Then $\sum_{(j,t)\notin T_{S!}} x^*_{j,t} w_j \geq \frac{1}{2}\sum_{j\in\bar{T}_S,t} x^*_{j,t} w_j$.*

Proof sketch. Let us pretend that $\sum_{C\in\mathcal{C}} y^*_C w_C = w(T_L \cap T^*)$. Then $\sum_{j\in\bar{T}_S,t} x^*_{j,t} w_j \geq w(T_S \cap T^*)$ since (x^*, y^*) is the optimal fractional solution. Therefore, $\sum_{j\in\bar{T}_S,t} x^*_{j,t} w_j \geq \frac{1}{\alpha}\sum_{C\in\mathcal{C}} y^*_C w_C$ and $(1+\frac{1}{\alpha})\sum_{j\in\bar{T}_S,t} x^*_{j,t} w_j \geq \frac{1}{\alpha}\left(\sum_{C\in\mathcal{C}} y^*_C w_C + \sum_{j\in\bar{T}_S,t} x^*_{j,t} w_j\right) = \frac{1}{\alpha}\mathrm{opt}_{LP}$. This implies that $\sum_{j\in\bar{T}_S,t} x^*_{j,t} w_j \geq \mathrm{opt}_{LP}/(\alpha+1)$. Since $\sum_{(j,t)\in T_{S!}} x^*_{j,t} w_j \leq 4\eta\mathrm{opt}_{LP} \leq \frac{\mathrm{opt}_{LP}}{2(\alpha+1)} \leq \frac{1}{2}\sum_{j\in\bar{T}_S,t} x^*_{j,t} w_j$ the claim follows. ◀

Each non-problematic pair is discarded only with probability at most $1-\eta$. Therefore, the expected profit of the auxiliary solution x is at least an η-fraction of the profit of the non-problematic pairs. Due to Lemma 30 we can neglect the profit due to problematic pairs. For $\eta := \frac{1-O(\varepsilon)}{8(\alpha+1)}$ the claim of Lemma 26 follows from some simple calculation.

Arbitrary stair-solutions

In order to compute a profitable γ-stair-solution we device a DP that intuitively sweeps the path from left to right and guesses the stair-blocks in the optimal stair-solution. For each stair-block we invoke the algorithm above. Since each edge can be used by at most γ stair-blocks and large tasks we obtain a running time of $n^{(\gamma\log n)^{O(c/\delta)}}$. Since we require the stair-blocks to be compatible, there can be no task that can be assigned to more than one stair-block, even if the subproblems for each stair-block is solved independently. Recall that for a large task $i \in T_L$ we required that $h(i) < d_i$ for its computed height $h(i)$ and hence we cannot assign it into two stair-blocks, even if it would fit into the respective areas of the stair-blocks. We defer the details to the full version of the paper [28].

References

1. Anna Adamaszek and Andreas Wiese. A quasi-PTAS for the two-dimensional geometric knapsack problem. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015)*, pages 1491–1505. SIAM, 2015.
2. Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Andreas Wiese. A mazing $2+\varepsilon$ approximation for unsplittable flow on a path. In *SODA*, 2014.
3. N. Bansal, A. Chakrabarti, A. Epstein, and B. Schieber. A quasi-PTAS for unsplittable flow on line graphs. In *STOC*, pages 721–729. ACM, 2006.
4. N. Bansal, Z. Friggstad, R. Khandekar, and R. Salavatipour. A logarithmic approximation for unsplittable flow on line graphs. In *SODA*, pages 702–709, 2009.

5 Nikhil Bansal, Alberto Caprara, Klaus Jansen, Lars Prädel, and Maxim Sviridenko. A structural lemma in 2-dimensional packing, and its implications on approximability. In *Algorithms and Computation*, pages 77–86. Springer, 2009.

6 Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM (JACM)*, 48(5):1069–1090, 2001.

7 Reuven Bar-Yehuda, Michael Beder, Yuval Cohen, and Dror Rawitz. Resource allocation in bounded degree trees. *Algorithmica*, 54(1):89–106, 2009.

8 Reuven Bar-Yehuda, Michael Beder, and Dror Rawitz. A constant factor approximation algorithm for the storage allocation problem. *Algorithmica*, 77(4):1105–1127, 2017.

9 Jatin Batra, Naveen Garg, Amit Kumar, Tobias Mömke, and Andreas Wiese. New approximation schemes for unsplittable flow on a path. In *SODA*, pages 47–58, 2015. doi:10.1137/1.9781611973730.5.

10 Paul Bonsma, Jens Schulz, and Andreas Wiese. A constant-factor approximation algorithm for unsplittable flow on paths. *SIAM Journal on Computing*, 43:767–799, 2014.

11 Adam L Buchsbaum, Howard Karloff, Claire Kenyon, Nick Reingold, and Mikkel Thorup. Opt versus load in dynamic storage allocation. *SIAM Journal on Computing*, 33(3):632–646, 2004.

12 Gruia Călinescu, Amit Chakrabarti, Howard J. Karloff, and Yuval Rabani. An improved approximation algorithm for resource allocation. *ACM Transactions on Algorithms*, 7:48:1–48:7, 2011. doi:10.1145/2000807.2000816.

13 A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar. Approximation algorithms for the unsplittable flow problem. *Algorithmica*, 47:53–78, 2007.

14 C. Chekuri, A. Ene, and N. Korula. Unsplittable flow in paths and trees and column-restricted packing integer programs. In *APPROX-RANDOM*, pages 42–55, 2009.

15 C. Chekuri, M. Mydlarz, and F. Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Transactions on Algorithms*, 3, 2007.

16 Waldo Gálvez, Fabrizio Grandoni, Sandy Heydrich, Salvatore Ingala, Arindam Khan, and Andreas Wiese. Approximating geometric knapsack via l-packings. In *58th Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 260–271. IEEE, 2017.

17 Jordan Gergov. Approximation algorithms for dynamic storage allocation. In *Algorithms–ESA'96*, pages 52–61. Springer, 1996.

18 Jordan Gergov. Algorithms for compile-time memory optimization. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 907–908. Society for Industrial and Applied Mathematics, 1999.

19 Fabrizio Grandoni, Tobias Mömke, Andreas Wiese, and Hang Zhou. To augment or not to augment: Solving unsplittable flow on a path by creating slack. In *SODA*, pages 2411–2422, 2017.

20 Fabrizio Grandoni, Tobias Mömke, Andreas Wiese, and Hang Zhou. A $(5/3 + \varepsilon)$-approximation for unsplittable flow on a path: placing small tasks into boxes. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 607–619, 2018. doi:10.1145/3188745.3188894.

21 Sandy Heydrich and Andreas Wiese. Faster approximation schemes for the two-dimensional knapsack problem. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 79–98, 2017. doi:10.1137/1.9781611974782.6.

22 Klaus Jansen and Roberto Solis-Oba. A polynomial time approximation scheme for the square packing problem. In Andrea Lodi, Alessandro Panconesi, and Giovanni Rinaldi, editors, *Integer Programming and Combinatorial Optimization*, pages 184–198, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

23 Klaus Jansen and Guochuan Zhang. Maximizing the number of packed rectangles. In *Scandinavian Workshop on Algorithm Theory*, pages 362–371. Springer, 2004.

24 Klaus Jansen and Guochuan Zhang. Maximizing the total profit of rectangles packed into a rectangle. *Algorithmica*, 47(3):323–342, 2007.

25 Hal A Kierstead. The linearity of first-fit coloring of interval graphs. *SIAM Journal on Discrete Mathematics*, 1(4):526–530, 1988.

26 Hal A Kierstead. A polynomial time approximation algorithm for dynamic storage allocation. *Discrete Mathematics*, 88(2):231–237, 1991.

27 Tobias Mömke and Andreas Wiese. A $(2+\varepsilon)$-approximation algorithm for the storage allocation problem. In *Proceedings of the 42nd Annual International Colloquium on Automata, Languages and Programming (ICALP 2015)*, volume 9134 of *Lecture Notes in Computer Science*, pages 973–984. Springer, 2015.

28 Tobias Mömke and Andreas Wiese. Breaking the barrier of 2 for the storage allocation problem. *CoRR*, abs/1911.10871, 2019. `arXiv:1911.10871`.

29 C. A. Phillips, R. N. Uma, and J. Wein. Off-line admission control for general scheduling problems. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '00)*, pages 879–888. ACM, 2000.