A $(2+\epsilon)$-Approximation Algorithm for the Storage Allocation Problem

Tobias Mömke[1] and Andreas Wiese[2(✉)]

[1] Saarland University, Saarbrücken, Germany
moemke@cs.uni-saarland.de
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany
awiese@mpi-inf.mpg.de

Abstract. Packing problems are a fundamental class of problems studied in combinatorial optimization. Three particularly important and well-studied questions in this domain are the Unsplittable Flow on a Path problem (UFP), the Maximum Weight Independent Set of Rectangles problem (MWISR), and the 2-dimensional geometric knapsack problem. In this paper, we study the Storage Allocation Problem (SAP) which is a natural combination of those three questions. Given is a path with edge capacities and a set of tasks that are specified by start and end vertices, demands, and profits. The goal is to select a subset of the tasks that can be drawn as non-overlapping rectangles underneath the capacity profile, the height of a rectangles corresponding to the demand of the respective task. This problem arises naturally in settings where a certain available bandwidth has to be allocated contiguously to selected requests.

While for 2D-knapsack and UFP there are polynomial time $(2+\epsilon)$-approximation algorithms known [Jansen and Zhang, SODA 2004] [Anagnostopoulos et al., SODA 2014] the best known approximation factor for SAP is $9+\epsilon$ [Bar-Yehuda, SPAA 2013]. In this paper, we level the understanding of SAP and the other two problems above by presenting a polynomial time $(2+\epsilon)$-approximation algorithm for SAP. A typically difficult special case of UFP and its variations arises if all input tasks are relatively large compared to the capacity of the smallest edge they are using. For that case, we even obtain a pseudopolynomial time *exact* algorithm for SAP.

1 Introduction

Packing problems belong to the most fundamental problems in combinatorial optimization and approximation algorithms. One very prominent packing problem is the well-known Knapsack problem: given is a knapsack with a certain capacity and a set of items I, where each item i is specified by a demand d_i and a profit w_i. The task is to select a subset of the given items $I' \subseteq I$ such that their total demand is bounded by the capacity of the knapsack, the objective being to maximize the obtained profit $w(I') := \sum_{i \in I'} w_i$.

Research partially funded by by the Indo-German Max Planck Center for Computer Science (IMPECS) and Deutsche Forschungsgemeinschaft grant BL511/10-1.

M.M. Halldórsson et al. (Eds.): ICALP 2015, Part I, LNCS 9134, pp. 973–984, 2015.
DOI: 10.1007/978-3-662-47672-7_79

There are several natural generalizations of this basic setting. One is to add a second dimension to the problem such that each item i is represented by an axis-parallel rectangle. The problem is then to select a set of items and place their corresponding rectangles non-overlappingly into a rectangular box. This yields the 2-dimensional geometric knapsack problem.

Another natural extension of knapsack is to add a temporal component such that each item i has additionally a start time $s(i)$ and an end time $t(i)$ which specify when it is active, modelling that it stays in the knapsack only during $[s_i, t_i)$. We call the input items *tasks* in this setting. Typically, one models the time horizon by a path where each edge represents a discrete time point and the values $s(i)$ and $t(i)$ represent vertices of the path. Each edge e is equipped with a capacity u_e, modelling the available knapsack capacity at this time (which can differ from edge to edge). For each edge e we denote by T_e the input tasks whose $s(i)$-$t(i)$-path $P(i)$ uses e. For a computed set T' we then require that $d(T_e \cap T') \leq u_e$ for each edge e. This yields the well-studied Unsplittable Flow on a Path problem (UFP).

In this paper, we study the Storage Allocation Problem (SAP) which is a natural combination of UFP and 2-dimensional knapsack: Given the same input as for UFP, the goal is to select a subset T' of the input set T and we want to compute a vertical position $h(i)$ for each task $i \in T'$ such that we can represent the selected tasks by non-overlapping rectangles underneath the capacity profile, the rectangle for each task $i \in T'$ is drawn at height level $h(i)$ and has a width of d_i (see Figure 1(e)). Formally, we require that (i) $h(i) + d_i \leq u_e$ for each task $i \in T'$ and each edge $e \in P(i)$ and (ii) for any two tasks $i, i' \in T'$ if $P(i) \cap P(i') \neq \emptyset$ then $[h(i), h(i) + d_i) \cap [h(i'), h(i') + d_{i'}) = \emptyset$. Observe that any solution satifying conditions (i) and (ii) also satisfies that $d(T' \cap T_e) \leq u_e$, for each edge e. SAP is particularly motivated by settings where tasks need a contiguous portion of an available resource, i.e., a consecutive portion of the computer memory or a frequency bandwidth.

Seen from a different perspective, SAP is an intermediate problem between 2-dimensional knapsack and the Maximum Weight Independent Set of Rectangles problem (MWISR) in which we are also given a set of items in the form of axis-parallel rectangles that we want to select a non-overlapping subset from, but for each rectangle its placement is predetermined. In 2-dimensional knapsack we are allowed to translate the input rectangles in both dimensions, in SAP we can translate them only up and down, and in MWISR they are completely fixed. Also, SAP is related to the Dynamic Storage Allocation problem (DSA) we are given a set of tasks as above and we want to draw their respective rectangles so that the maximum height $\max_{i \in T} h(i) + d_i$ is minimized.

A lot of progress has been made on the packing problems listed above. Specifically, we now have polynomial time $(2 + \epsilon)$-approximation algorithms for 2-dimensional knapsack [28], UFP [4], and DSA [15] and quasi-polynomial time $(1 + \epsilon)$-approximation algorithms for UFP [6], MWISR [1], and 2-dimensional knapsack [2]. The state of the art for SAP is a $(9 + \epsilon)$-approximation in polynomial time [10] which is a best-of-three algorithm. It classifies a task i to be

δ-small if $d_i \leq \delta \cdot b(i)$ where $b(i)$ denotes the *bottleneck capacity* of i which is the minimum capacity of an edge used by i. Similarly, a task i is δ-large if $d_i > \delta \cdot b(i)$. Intuitively, the value δ denote the relative size of the tasks. The mentioned algorithm provides a $(4+\epsilon)$-approximation for δ-small tasks (for some small value δ depending on ϵ), a 3-approximation for $\frac{1}{2}$-large tasks and finally a $(2+\epsilon)$-approximation for the remaining tasks.

1.1 Our Contribution

In this paper, we level our understanding of SAP and the other packing problems mentioned above in terms of polynomial time approximation algorithms. We present a $(2+\epsilon)$-approximation algorithm for SAP whose ratio matches the factors of the respective best known polynomial time algorithms for UFP, DSA, and 2D-knapsack. It is a best-of-two algorithm which improves all components of the so far best known $(9+\epsilon)$-approximation algorithm [10]. First, we show that if tasks are sufficiently small, we can get a $(1+\epsilon)$-approximation by rounding a suitably defined new LP-relaxation. While such a result is known for UFP [21, Corollary 3.4], it is not clear how to transfer it to SAP, in particular, since the optimal value of the canonical LP for UFP can differ from the best SAP-solution for a given instance by up to a factor 2, even if all input tasks are arbitrarily small. Our key technical contribution here is that we present a way to reduce the overall problem to assigning tasks to rectangular strips underneath the capacity profile. Since tasks are small, at negligible loss we we can ignore the aspect that they are supposed to be drawn as non-overlapping rectangles and we ensure only that the load of each strip is bounded by its capacity. This yields our new LP-formulation for the problem. With a suitable rounding method, we prove that for any $\epsilon > 0$ there exists a $\delta > 0$ such that for SAP-instances with only δ-small tasks we obtain a $(1+\epsilon)$-approximation algorithm.

Then we study the converse setting where we assume that we are given a constant $\delta > 0$ and a SAP-instance with only δ-large tasks. In the related UFP problem, this is a rather difficult setting and the known PTAS for it [4] is very complex and involved. In this paper we present a very clean and elegant dynamic program for this setting for SAP. In particular, rather than computing an approximation, we solve the problem even *exactly* in pseudopolynomial time. In our DP we guess the tasks in the optimal solution step by step, ordered by their vertical positions in OPT. We prove that by using this order we need to remember only few information from the previous guesses. The pseudopolynomial running time stems from the fact that there are a pseudopolynomial number of possible vertical positions for each task and there are densely packed optimal solutions in which it is not sufficient to allow only fewer values e.g., only powers of $1+\epsilon$. However, using a result by Knipe [32] about trimming graphs with bounded treewidth, together with an argument by Erlebach et al. [23], we show that there are $(1+\epsilon)$-approximative solutions in which the task positions come only from a polynomial size subset. This yields a PTAS for δ-large tasks.

We round up our results by showing that any feasible solution for UFP, (i.e., any set of tasks satisfying the edge capacities) can be partitioned into $O(1)$

subsets such that each of them is a feasible solution to SAP. In a sense, this bounds the "price of contiguousness". Moreover, we can also show that if we increase the capacity of each edge by a constant factor, any UFP-solution also yields a SAP solution. This connects well with a result by Gergov [26] which proves an upper bound of 3 for the special case for uniform edge capacities, improving on several earlier results [25,30,31].

1.2 Related Work

For the special case of SAP that all edges have the same capacities, a local ratio 7-approximation algorithm is presented by Bar-Noy et al. [8], using an algorithm by Gergov for DSA [26]. This is improved by Bar-Yehuda to a randomized $(2+\epsilon)$-approximation algorithm and a deterministic $\frac{2e-1}{e-1}+\epsilon \approx 2.582$-approximation for the same special case [9]. In fact, there is a close connection between unsplittable flow and dynamic storage allocation in the case of uniform edge capacities and if all tasks are sufficiently small. A result by Buchsbaum et al. [15] implies that then, if a set of tasks is feasible for UFP then a $(1-\epsilon)$-fraction of it yields a feasible solution for SAP. However, this connection breaks if edges have different capacities. Chen et al. [22] provide an exact dynamic programming algorithm running in time $O(n(Kn)^K)$ assuming that all demands are integral multiples of $1/K$ and an $(\frac{e}{e-1}+\epsilon)$-approximation if all demands have size $O(1/K)$. As mentioned above, Bar-Yehuda, Beder, and Rawitz [10] present a $(9+\epsilon)$-approximation algorithm for general SAP with arbitrary edge capacities which is the best known result for this case. For UFP, after a long line of work on the special cases of uniform edge capacities [8,16,34], the no-bottleneck-assumption [17,21] and the general case [4,6,14,20**?**] the best known results are now are quasi-PTASs due to Bansal et al. [6] and Batra et al. [11] and a polynomial time $(2+\epsilon)$-approximation algorithm by Anagnostopoulos et al. [4]. Recently, Batra et al. [11] presented PTASs for two special cases.

The two-dimensional geometric knapsack problem admits a $(2+\epsilon)$-approximation algorithm due to Jansen and Zhang [28]. PTASs are known if the size of the knapsack can be increased by a factor $(1+\epsilon)$ in both dimensions [24] or even only in one of them [27] while the compared optimum has to use the original knapsack. Also, there is a PTAS if the profit of each item equals its area [5]. For MWISR, there are many polynomial time $O(\log n)$-approximations algorithms known [3,12,29,33], and the best known result is a $O(\log n/\log\log n)$-approximation by Chan and Har-Peled [19]. For the unweighted case, there is also a $O(\log\log n)$-approximation by Chalermsook and Chuzhoy [18]. Recently, a quasi-PTAS (for the weighted case) was found [1]. As mentioned above, a result by Buchsbaum et al. [15] for DSA states that if all tasks are sufficiently small then they can be drawn within a height of at most $(1+\epsilon)\cdot L$ where L denotes the maximum total demand of tasks crossing any edge. Combined with a DP for the other tasks, this yields a $(2+\epsilon)$-approximation algorithm. For bounding the needed height as a function of L the best known bound is from Gergov [26] who shows an upper bound of $3\cdot L$, improving on previous results [25,30,31].

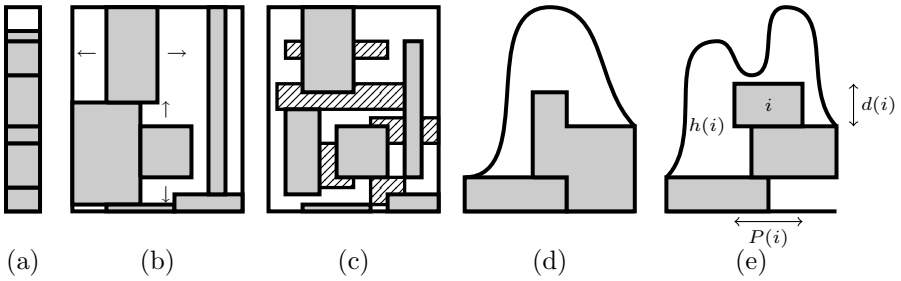

Fig. 1. (a) Knapsack problem, (b) two-dimensional knapsack problem, (c) independent set of rectangles, (d) unsplittable flow on a path (UFP), (e) storage allocation problem (SAP)

2 Approximating Small Tasks up to a Factor $1+\epsilon$

In this section, we prove that for any $\epsilon > 0$ there is a $\delta > 0$ such that for δ-small tasks we can construct a $(1+\epsilon)$-approximation algorithm. First, we show this result for the special case that the edge capacities are in a constant range. Then, we show how to reduce the general case to this special case.

2.1 Edge Capacities in Constant Range

Let $\epsilon > 0$. Assume that the edge capacities lie in a constant range, i. e., assume that there is a constant U such that $\max_e u_e \leq U \cdot \min_e u_e$. Assume for simplicity that $U \in \mathbb{N}$ and $1/\epsilon \in \mathbb{N}$. We draw a set of strips in the area underneath the capacity profile. A strip is specified by a tuple (k, v_ℓ, v_r) which intuitively represents the rectangle $[v_\ell, v_r] \times \{k \cdot \epsilon \cdot \min_e u_e, (k+1) \cdot \epsilon \cdot \min_e u_e\}$ where the vertices of the graph are interpreted as integers. Let $\mathcal{S}$ denote the set of all maximally long strips whose respective rectangles fit underneath the capacity profile. Formally, a strip (k, v_ℓ, v_r) is contained in $\mathcal{S}$ if each edge between v_ℓ and v_r has a capacity of at least $(k+1) \cdot \epsilon \cdot \min_e u_e$ and the edges on the left of v_ℓ and on the right of v_r have a capacity of less than $(k+1) \cdot \epsilon \cdot \min_e u_e$ or do not exist because v_ℓ / v_r are the left/right-most vertices of the path. For a strip $S = (k, v_\ell, v_r)$ denote by $P(S)$ the set of edges between v_ℓ and v_r.

Instead of aiming directly at selecting a set of tasks T' and finding a non-overlapping drawing of them, we compute a set $T' \subseteq T$ and an assignment $f : T' \to \mathcal{S}$ of them to the strips. We require that for each strip $S \in \mathcal{S}$ that (i) each task $i \in f^{-1}(S)$ fits into S, meaning that $P(i) \subseteq P(S)$, and (ii) the total demand of the tasks in each strip S does not exceed the capacity of S on any edge $e \in P(S)$, i. e., $d(f^{-1}(S) \cap T_e) \leq \epsilon \cdot \min_e u_e$ for each edge $e \in P(S)$.

We call a pair (T', f) satisfying the above a *strip assignment*. It is not true that for any feasible solution (T', h) we can find a strip assignment (T', f) with the same set of tasks T', already because the total capacity of the strips using

some edge e might be smaller than u_e. The converse statement is also false, since the second property above is only a relaxation of the requirement that tasks should be drawn as non-overlapping rectangles. Nevertheless, we can show that if tasks are very small compared to the capacity of each strip, then the two notions are equivalent up to a factor $1+\epsilon$. Key to show this is that the unavailable edge capacity is small compared to the total capacity and we assume all tasks to be very small. Also, based on a result by Buchsbaum et al. [15], Bar-Yehuda et al. [9] showed that if the two properties are true for some strip S, then a $(1-\epsilon)$-fraction of the tasks in $f^{-1}(S)$ can be in fact drawn as non-overlapping rectangles. Using this intuition, we can prove the following lemma.

Lemma 1. *For any $\epsilon > 0$ there is a $\delta_1 > 0$ with the following property. Assume we are given an instance in which every task is δ_1/U-small. Then for any feasible solution (T', h) there is a strip assignment (T'', f') with $w(T'') \geq (1-O(\epsilon))w(T')$. Conversely, for any strip assignment (T'', f') there is a feasible solution (T', h) with $w(T') \geq (1-O(\epsilon))w(T'')$.*

Knowing that it is sufficient to compute a good strip assignment, we present now an LP-rounding algorithm for the latter goal. We formulate the problem as an integer program whose LP-relaxation (STRIP-LP) is given below.

$$\begin{array}{lll} \max \sum_{i,S} w_i \cdot x_{i,S} & & \\ \text{s.t.} \quad \sum_{i \in T_e} x_{i,S} \cdot d_i \leq \epsilon \cdot \min_e u_e & \forall S \in \mathcal{S}, & (2.1) \\ & \forall e \in P(S) & \\ \sum_S x_{i,S} \leq 1 & \forall i \in T & \\ x_{i,S} \geq 0 & \forall i \in T\, \forall S \in \mathcal{S} & \\ & s.t.\, P(i) \subseteq P(S) & \end{array}$$

We compute the optimal feasible solution to the above LP. By losing only a factor $(1+\epsilon)$, we round it to a strip assignment via randomized rounding with alteration as introduced by Calinescu et al. [16]. Important for this to work is that the demand of each input task is sufficiently small compared to the capacity of each strip. In the rounding, we first sample a preliminary integral solution y such that $\Pr[y_{i,S} = 1] = (1-\epsilon)x_{i,S}$ and $\Pr[y_{i,S} = 1 \wedge y_{i,S'} = 1] = 0$ for any task i and for any two strips $S, S' \in \mathcal{S}$. Such a distribution can easily be obtained via dependant rounding similar to Bertsimas et al. [13]. Then, intuitively, in an alteration phase for each strip, we consider the tasks in the order of their start vertices and drop a task if it causes a capacity constraint (2.1) to be violated. We can show that the probability that a task is dropped in this alteration phase is bounded by $O(\epsilon)$. In contrast to the method of Calinescu et al. [16] we work with dependent rounding. However, for any pair of tasks the outcomes of the random experiment are still independent and thus the argumentation from [16] still works.

Lemma 2. *For any $\epsilon > 0$ there is a $\delta_2 > 0$ such that given any instance with only δ_2/U-small tasks and a solution x^* to (STRIP-LP), there is a polynomial*

time algorithm computing a strip assignment (T', f) *with* $w(T') \geq (1-\epsilon)\sum_{i,S} w_i \cdot x^*_{i,S}$.

Together with Lemma 1, we thus obtain a $(1 + O(\epsilon))$-approximation for instances with only δ/U-small tasks where $\delta := \min\{\delta_1, \delta_2\}$. Next, we give a reduction of the general case to this special case.

2.2 Arbitrary Edge Capacities

The key idea is to use some shifting arguments to remove tasks with small total cost from the optimal solution and move some other tasks up into the resulting empty space. As a result, afterwards each task i is drawn at a position $h(i) \in \Omega_\epsilon(b(i))$, i.e., not too far below its bottleneck edge. As a result, we can split the problem into independent subproblems, each having a bounded range of edge capacities.

Lemma 3. *Let $\epsilon > 0$ such that $1/\epsilon$ is an integer. There is a $\delta > 0$ such that for any instance I with only δ-small tasks there is an integer value ℓ with $0 \leq \ell < 1/\epsilon$ and a solution $(\bar{T}, \bar{h})$ to I where*

1. *$w(\bar{T}) \geq (1-2\epsilon)\mathrm{OPT}(I)$,*
2. *for each task $i \in \bar{T}$ with $b(i) \geq 2^{\ell+1+r/\epsilon}$ it holds that $\bar{h}(i) \geq 2^{\ell+r/\epsilon}$, and*
3. *for each task $i \in \bar{T}$ with $b(i) < 2^{\ell+1+r/\epsilon}$ it holds that $\bar{h}(i) + d_i \leq 2^{\ell+r/\epsilon}$*

for any $r \in \mathbb{N}_0$.

Proof sketch. Given the optimal solution (T^*, h^*) of the δ-small tasks, we assign the tasks into groups according to the position at which they are drawn. Denote by $T^*_k \subseteq T^*$ all tasks from T^* whose rectangles have non-empty intersection with the horizontal strip $[0, |V|] \times [2^k, 2^{k+1})$.

Formally, we define $T^*_k := \{i \in T^* | [h^*(i), h^*(i) + d_i) \cap [2^k, 2^{k+1}) \neq \emptyset\}$. Note that a task might appear in several of these sets. Let $\bar{k}$ denote the largest index k such that $T^*_k \neq \emptyset$ and consider the sets $T^*_{\bar{k}-1/\epsilon+1}, T^*_{\bar{k}-1/\epsilon+2}, ..., T^*_{\bar{k}}$. If δ is sufficiently small then each task appears in at most two of these groups. We select one set $T^*_{k'} \in \{T^*_{\bar{k}-1/\epsilon}, T^*_{\bar{k}-1/\epsilon+1}, ..., T^*_{\bar{k}}\}$ uniformly at random and remove all its tasks. In expectation we lose at most an 2ϵ-fraction of $w(\bigcup_{k=\bar{k}-1/\epsilon+1}^{\bar{k}} T^*_k)$. Then, we take all tasks $i \in T^*$ with $b(i) \geq 2^{k'+1}$ and $h(i) + d_i \leq 2^{k'}$ and move them up by $2^{k'}$ units, i.e., we define $\bar{h}^*(i) := h^*(i) + 2^{k'}$ for them. As a result, we obtain the property that for each task $i \in T^* \setminus T^*_{k'}$ with $b(i) \geq 2^{k'+1}$ it holds that $\bar{h}^*(i) \geq 2^{k'}$ and for each task $i' \in T^* \setminus T^*_{k'}$ with $b(i') < 2^{k'+1}$ it holds that $\bar{h}^*(i) + d_i < 2^{k'}$. Iterating the above argument over multiple levels then completes the proof. □

Observe that Lemma 3 decouples the instance into separate subinstances. For each $r \in \mathbb{N}_0$ we have one subinstance consisting of the tasks $T^r := \{i \in T : 2^{\ell+1+r/\epsilon} \leq b(i) < 2^{\ell+1+(r+1)/\epsilon}\}$ for which we are looking for a solution with $2^{\ell+r/\epsilon} \leq \bar{h}(i)$ and $\bar{h}(i) + d_i \leq 2^{\ell+(r+1)/\epsilon}$ for each selected task $i \in T^r$. Thus, we

can treat each group T^r independently as an instance where the edge capacities are in a range of $[2^{\ell+r/\epsilon}, 2^{\ell+1+(r+1)/\epsilon} - 2^{\ell+r/\epsilon})$.

We define a constant $\delta' \in O_\epsilon(1)$ such that the algorithm from the previous section gives us a $(1+\epsilon)$-approximation for the δ'-small tasks in each group T^r. This yields a $(1+\epsilon)$-approximation algorithm for instances with only $\min\{\delta, \delta'\}$-small tasks where δ is the constant due to Lemma 3.

Theorem 1. *For each $\epsilon > 0$ there is a $\delta > 0$ such that there is a $(1+\epsilon)$-approximation algorithm for the storage allocation problem if the input consists of δ-small tasks only.*

3 Large Tasks

In this section we present a pseudo-polynomial time exact algorithm for the storage allocation problem for δ-large tasks, for any $\delta > 0$. Subsequently, we show how to turn it into a polynomial time $(1+\epsilon)$-approximation algorithm. Together with Theorem 1 this yields a polynomial time $(2+\epsilon)$-approximation algorithm for SAP.

Since all input data are integers we can assume w.l.o.g. that all position values $h(i)$ in the optimal solution are integers: for any optimal solution without this property we can apply "gravity", i. e., decrease the vertical position of all tasks as much as we can. In the resulting solution each arising position of a task is either zero or the sum of the demands of some other tasks and thus an integer.

Our algorithm is a dynamic program. Denote by (OPT, h^*) the optimal solution. In the first step, we guess the task $i_0 \in OPT$ with smallest position $h^*(i_0)$ and all tasks from OPT using its bottleneck edge $e(i_0)$. Denote them by OPT_0. Since all tasks are δ-large, there can be only $1/\delta$ of them. More precisely, we guess these tasks as well as their positions according to h^*. The whole problem splits then into two disjoint subproblems given by the subpath on the left of $e(i_0)$ and the subpath on the right of $e(i_0)$. We recurse on both sides. Consider the left side and let $OPT_L \subseteq OPT$ denote the tasks from OPT whose path is completely contained in the subpath on the left of $e(i_0)$. Note that $OPT_L \cap OPT_0 = \emptyset$. We guess the task $i_1 \in OPT_L$ with smallest position $h^*(i_1)$. Naively, one would like to guess all tasks using $e(i_1)$ and then recurse again. The problem is that $e(i_1)$ might be used by all up to $1/\delta$ tasks in OPT_0 and another $1/\delta$ tasks from OPT_L. Thus, in each recursive step the number of tasks to be remembered would increase by $1/\delta$ while the recursion depth could be even linear. Thus, we could not bound the number of DP-cells by a polynomial. Instead, we first show that the number of tasks $i \in OPT$ (not only OPT_L!) using $e(i_1)$ with $h(i) \geq h^*(i_1)$ is bounded by $1/\delta^2$ as the following lemma implies.

Lemma 4. *Consider any solution (T', h') and a task $i \in T'$. There are at most $1/\delta^2$ tasks $i' \in T'$ such that $e(i) \in P(i')$ and $h(i') \geq h(i)$.*

Proof. For any task i' with $h(i') \geq h(i)$ and $e(i) \in P(i')$, we have that $d_i \leq h(i') \leq b(i') - d_{i'}$. Thus, using that i and i' are δ-large, $d_{i'}/\delta \geq b(i') \geq \delta \cdot b(i) + d_{i'}$ and thus $d_{i'} \geq \delta^2 \cdot b(i)/(1-\delta) > \delta^2 \cdot b(i)$. □

When recursing on the subpath on the left of $e(i_1)$, we specify the subproblem by its subpath, by the at most $1/\delta^2$ tasks $i \in OPT$ using $e(i_1)$ with $h(i) \geq h^*(i_1)$, and the information that each task in the desired solution to this subproblem has to have a height of at least $h^*(i_1)$.

When continuing with this recursion, each arising subproblem can be characterized by two edges e_L, e_R, by at most $1/\delta^2$ tasks T_L using e_L together with a placement $h(i)$ for each task $i \in T_L$, at most $1/\delta^2$ tasks T_R using e_R together with a placement $h(i)$ for each task $i \in T_R$, and an integer $h_{\min}$. For each such combination we introduce a DP-cell $(e_L, e_R, T_L, T_R, h, h_{\min})$. Formally, it models the following subproblem: assume that we committed to selecting tasks T_L and T_R and assigning heights to them as given by the function h. Now we ask for the maximum profit we can obtain by selecting additional tasks whose paths are contained in the path strictly between e_L and e_R (so excluding e_L and e_R) and assigning heights to them such that $h(i) \geq h_{\min}$ for each selected task i.

For a given DP-cell $C = (e_L, e_R, T_L, T_R, h, h_{\min})$ we denote by (OPT_C, h^*_C) its optimal solution. Observe that $OPT_C \cap T_L = \emptyset = OPT_C \cap T_R$. Let $\hat{i} \in OPT_C$ be the task in OPT_C with minimum height $h^*_C(\hat{i})$. To compute (OPT_C, h^*_C) we guess $\hat{i} \in OPT_C$ and $h^*_C(\hat{i})$. According to Lemma 4 there can be at most $1/\delta^2$ tasks $i \in OPT_C \cup T_L \cup T_R$ using $e(\hat{i})$ such that $h^*_C(i) \geq h^*_C(\hat{i})$. We also guess all those tasks (the tasks from T_L and T_R among them are of course already given) together with their respective heights according to h^*_C. Denote them by $\bar{T}$. We can then show that OPT_C consists of the tasks in $\bar{T}$ together with the tasks in the optimal solutions to the DP-cells $C' := (e_L, e(\hat{i}), T_L, \bar{T}, h', h^*_C(\hat{i}))$ and $C'' := (e(\hat{i}), e_R, \bar{T}, T_R, h'', h^*_C(\hat{i}))$ where the assignments h' and h'' are obtained by inheriting from h the values for the tasks in T_L and T_R, respectively, and taking the guessed values for the tasks in $\bar{T}$. Conversely, we can easily show that we obtain a feasible solution to the original cell C if we combine two arbitrary feasible solutions for C' and C'', respectively, with $\bar{T}$. This proves the correctness of our DP. Since the total number of DP-cells is bounded by $(n \cdot \max_i d_i)^{O(1/\delta^2)}$ we obtain an exact pseudopolynomial time algorithm.

Theorem 2. *Let $\delta > 0$. There is an exact algorithm for instances of SAP with only δ-large tasks whose running time is $(n \cdot \max_i d_i)^{O(1/\delta^2)}$ where n denotes the number of tasks.*

In order to still obtain a PTAS, our strategy is to bound the number of candidate values for the height $h(i)$ of each task i. We will show that there is a set of such candidates of polynomial size such that there is a $(1+\epsilon)$-approximative solution in which each selected task is assigned a height from this set. By allowing only these heights in the above DP computation we then obtain a PTAS.

Our first step is to introduce a polynomial number of heights which we call *anchors lines*. Those are the capacities of the input edges and all powers of $1+\delta$ between 1 and $\max_e u_e$. Denote by H_0 this set of values. W.l.o.g. from now on we restrictly ourselves to solutions in which for each task the height of its top edge equals the height of an anchor line or its top edge touches the bottom edge of some other task. We call such solutions *top-aligned* solutions. In a given solution,

we say that a task is in level 1 if the height of its top edge equals an anchor line. Recursively, a task i is in level $\ell+1$ if its top edge touches the bottom edge of a task in level ℓ and i is not in level any level $\ell' < \ell$. Our goal is to show that there is a $(1+\epsilon)$-approximative solution in which each task has a level of at most $c(\epsilon)$ for some constant $c(\epsilon)$ that holds universally for any input instance. Observe that for the heights of the tasks in level ℓ there are only $|H_0| \cdot n^\ell$ possible values that are obtained by recursively defining $H_{k+1} := H_k \cup \{h - d_i | h \in H_k, i \in T\}$ for each k.

To this end, consider an optimal solution (T^*, h^*). We construct the following directed graph $D(T^*, h^*)$. For each task $i \in T^*$ we introduce a vertex in $D(T^*, h^*)$. There is an edge from the vertex for i to the vertex for i' if and only if the following three conditions are satisfied: $P(i) \cap P(i') \neq \emptyset$; $h^*(i) + d_i \leq h^*(i')$; there is no anchor line strictly between $h^*(i) + d_i$ and $h^*(i')$. Using that the tasks are δ-large, we can show that each tasks rectangle is crossed by some anchor line which implies that $D(T^*, h^*)$ is planar. Moreover, the second condition implies that $D(T^*, h^*)$ is acyclic.

Proposition 1. *If the length of the longest chain in $D(T^*, h^*)$ is bounded by some value ℓ, then $h^*(i) \in H_\ell$ for each $i \in T^*$.*

Unfortunately, the length of the longest chain in $D(T^*, h^*)$ might be $\Omega(n)$, e.g., when all tasks are tightly stacked on top of each other. To construct a solution where the latter is bounded, we apply the following theorem to $D(T^*, h^*)$. It can be proven by combining a result from Knipe [32] on trimming weighted graphs with bounded treewidth with the argumentation used in Corollary 2.3 in [23] (see also the discussion in Section 4 in the latter paper).

Theorem 3 ([23,32]). *Let $\epsilon > 0$. There exists a constant $c(\epsilon) \in \mathbb{N}$ such that for any planar graph $G = (V, E)$ with vertex weights given by a function $w : V \to \mathbb{R}$ there is a set of vertices $V' \subseteq V$ such that $w(V') \geq (1-\epsilon)w(V)$ and the length of the longest simple path in $G[V']$ is bounded from above by $c(\epsilon)$.*

The above theorem yields a subset of the vertices in $D(T^*, h^*)$ and thus a set of tasks $\bar{T}^* \subseteq T^*$ with $w(\bar{T}^*) \geq (1-\epsilon)w(T^*)$. Starting from their heights according to the function h^* we construct a top-aligned solution $(T^* \setminus T', h')$ by pushing up each task until the height of its top edge either equals the height of an anchor line or the height of the bottom edge of some other task. Since the length of the longest chain in $D(T^* \setminus T', h')$ is bounded by $c(\epsilon)$ the same upper bound holds for the maximum level of a task. This shows that there is a $(1-\epsilon)$-approximative solution $(T^* \setminus T', h')$ in which $h'(i) \in H_{c(\epsilon)}$ for each $i \in T^* \setminus T'$. By restricting our DP from the previous section to use only heights in $H_{c(\epsilon)}$ for the tasks we obtain our theorem below.

Theorem 4. *Let $\epsilon > 0$ and $\delta > 0$. There is a polynomial time $(1+\epsilon)$-approximation algorithm for instances of SAP with only δ-large tasks.*

In order to obtain our overall $(2+\epsilon)$-approximation algorithm, for any given $\epsilon > 0$ we first choose $\delta > 0$ according to Theorem 1. Then we compute a $(1+\epsilon)$-approximations for the δ-small and the δ-large tasks using Theorems 1 and 4. Selecting the best of these two solutions yields a $(2+\epsilon)$-approximation overall.

Theorem 5. *Let $\epsilon > 0$. There is a polynomial time $(2+\epsilon)$-approximation algorithm for SAP.*

Acknowledgments. We would like to thank Naveen Garg, Amit Kumar, and Jatin Batra for helpful discussions.

References

1. Adamaszek, A., Wiese, A.: Approximation schemes for maximum weight independent set of rectangles. In: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS), pp. 400–409. IEEE (2013)
2. Adamaszek, A., Wiese, A.: A quasi-PTAS for the two-dimensional geometric knapsack problem. In: Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, pp. 1491–1505 (2015)
3. Agarwal, P.K., van Kreveld, M., Suri, S.: Label placement by maximum independent set in rectangles. Computational Geometry **11**, 209–218 (1998)
4. Anagnostopoulos, A., Grandoni, F., Leonardi, S., Wiese, A.: A mazing 2+ϵ approximation for unsplittable flow on a path. In: Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014) (2014)
5. Bansal, N., Caprara, A., Jansen, K., Prädel, L., Sviridenko, M.: A structural lemma in 2-dimensional packing, and its implications on approximability. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 77–86. Springer, Heidelberg (2009)
6. Bansal, N., Chakrabarti, A., Epstein, A., Schieber, B.: A quasi-PTAS for unsplittable flow on line graphs. In: STOC, pp. 721–729. ACM (2006)
7. Bansal, N., Friggstad, Z., Khandekar, R., Salavatipour, R.: A logarithmic approximation for unsplittable flow on line graphs. In: SODA, pp. 702–709 (2009)
8. Bar-Noy, A., Bar-Yehuda, R., Freund, A., Naor, J., Schieber, B.: A unified approach to approximating resource allocation and scheduling. Journal of the ACM (JACM) **48**(5), 1069–1090 (2001)
9. Bar-Yehuda, R., Beder, M., Cohen, Y., Rawitz, D.: Resource allocation in bounded degree trees. Algorithmica **54**(1), 89–106 (2009)
10. Bar-Yehuda, R., Beder, M., Rawitz, D.: A constant factor approximation algorithm for the storage allocation problem. In: Proceedings of the 25th ACM symposium on Parallelism in Algorithms and Architectures, pp. 204–213. ACM (2013)
11. Batra, J., Garg, N., Kumar, A., Mömke, T., Wiese, A.: New approximation schemes for unsplittable flow on a path. In: SODA, pp. 47–58 (2015)
12. Berman, P., DasGupta, B., Muthukrishnan, S., Ramaswami, S.: Improved approximation algorithms for rectangle tiling and packing. In: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 427–436. Society for Industrial and Applied Mathematics (2001)
13. Bertsimas, D., Teo, C.-P., Vohra, R.: On dependent randomized rounding algorithms. Oper. Res. Lett. **24**(3), 105–114 (1999)
14. Bonsma, P., Schulz, J., Wiese, A.: A constant-factor approximation algorithm for unsplittable flow on paths. SIAM Journal on Computing **43**, 767–799 (2014)
15. Buchsbaum, A.L., Karloff, H., Kenyon, C., Reingold, N., Thorup, M.: Opt versus load in dynamic storage allocation. SIAM Journal on Computing **33**(3), 632–646 (2004)

16. Călinescu, G., Chakrabarti, A., Karloff, H.J., Rabani, Y.: An improved approximation algorithm for resource allocation. ACM Transactions on Algorithms **7**, 48:1–48:7 (2011)
17. Chakrabarti, A., Chekuri, C., Gupta, A., Kumar, A.: Approximation algorithms for the unsplittable flow problem. Algorithmica **47**, 53–78 (2007)
18. Chalermsook, P., Chuzhoy, J.: Maximum independent set of rectangles. In: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2009), pp. 892–901. SIAM (2009)
19. Chan, T.M., Har-Peled, S.: Approximation algorithms for maximum independent set of pseudo-disks. Discrete & Computational Geometry **48**(2), 373–392 (2012)
20. Chekuri, C., Ene, A., Korula, N.: Unsplittable flow in paths and trees and column-restricted packing integer programs. In: APPROX-RANDOM, pp. 42–55 (2009)
21. Chekuri, C., Mydlarz, M., Shepherd, F.: Multicommodity demand flow in a tree and packing integer programs. ACM Transactions on Algorithms **3**, (2007)
22. Chen, B., Hassin, R., Tzur, M.: Allocation of bandwidth and storage. IIE Transactions **34**(5), 501–507 (2002)
23. Erlebach, T., Hagerup, T., Jansen, K., Minzlaff, M., Wolff, A.: Trimming of graphs, with application to point labeling. Theory of Computing Systems **47**(3), 613–636 (2010)
24. Fishkin, A.V., Gerber, O., Jansen, K., Solis-Oba, R.: Packing weighted rectangles into a square. In: Jedrzejowicz, J., Szepietowski, A. (eds.) MFCS 2005. LNCS, vol. 3618, pp. 352–363. Springer, Heidelberg (2005)
25. Gergov, J.: Approximation algorithms for dynamic storage allocation. In: Díaz, J. (ed.) ESA 1996. LNCS, vol. 1136, pp. 52–61. Springer, Heidelberg (1996)
26. Gergov, J.: Algorithms for compile-time memory optimization. In: Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 907–908. Society for Industrial and Applied Mathematics (1999)
27. Jansen, K., Solis-Oba, R.: New approximability results for 2-dimensional packing problems. In: Kučera, L., Kučera, A. (eds.) MFCS 2007. LNCS, vol. 4708, pp. 103–114. Springer, Heidelberg (2007)
28. Jansen, K., Zhang, G.: On rectangle packing: maximizing benefits. In: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 204–213. Society for Industrial and Applied Mathematics (2004)
29. Khanna, S., Muthukrishnan, S., Paterson, M.: On approximating rectangle tiling and packing. In: Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1998), pp. 384–393. SIAM (1998)
30. Kierstead, H.A.: The linearity of first-fit coloring of interval graphs. SIAM Journal on Discrete Mathematics **1**(4), 526–530 (1988)
31. Kierstead, H.A.: A polynomial time approximation algorithm for dynamic storage allocation. Discrete Mathematics **88**(2), 231–237 (1991)
32. Knipe, D.: Trimming weighted graphs of bounded treewidth. Discrete Applied Mathematics **160**(6), 902–912 (2012)
33. Nielsen, F.: Fast stabbing of boxes in high dimensions. Theor. Comp. Sc. **246**, 53–72 (2000)
34. Phillips, C.A., Uma, R.N., Wein, J.: Off-line admission control for general scheduling problems. In: Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2000), pp. 879–888. ACM (2000)