A $(5/3 + \varepsilon)$-Approximation for Unsplittable Flow on a Path: Placing Small Tasks into Boxes

Fabrizio Grandoni[*]
IDSIA, USI-SUPSI, Switzerland
fabrizio@idsia.ch

Tobias Mömke[†]
Saarland University, Saarland Informatics Campus, Germany
University of Bremen, Germany
moemke@cs.uni-saarland.de

Andreas Wiese
Department of Industrial Engineering and Center for Mathematical Modeling, Universidad de Chile, Santiago, Chile
awiese@dii.uchile.cl

Hang Zhou[‡]
École Polytechnique, Computer Science Department (LIX), France
hzhou@lix.polytechnique.fr

ABSTRACT

In the unsplittable flow on a path problem (UFP) we are given a path with edge capacities and a collection of tasks. Each task is characterized by a subpath, a profit, and a demand. Our goal is to compute a maximum profit subset of tasks such that, for each edge e, the total demand of selected tasks that use e does not exceed the capacity of e. The current best polynomial-time approximation factor for this problem is $2 + \varepsilon$ for any constant $\varepsilon > 0$ [Anagostopoulos et al.-SODA 2014]. This is the best known factor even in the case of uniform edge capacities [Călinescu et al.-IPCO 2002, TALG 2011]. These results, likewise most prior work, are based on a partition of tasks into large and small depending on their ratio of demand to capacity over their respective edges: these algorithms invoke $(1 + \varepsilon)$-approximations for large and small tasks separately.

The known techniques do not seem to be able to combine a *big* fraction of large and small tasks together (apart from some special cases and quasi-polynomial-time algorithms). The main contribution of this paper is to overcome this critical barrier. Namely, we present a polynomial-time algorithm that obtains roughly all profit from the optimal large tasks plus one third of the profit from the optimal small tasks. In combination with known results, this implies a polynomial-time $(5/3 + \varepsilon)$-approximation algorithm for UFP.

Our algorithm is based on two main ingredients. First, we prove that there exist certain sub-optimal solutions where, roughly speaking, small tasks are packed into boxes. To prove that such solutions can yield high profit we introduce a *horizontal slicing lemma* which yields a novel geometric interpretation of certain solutions. The resulting boxed structure has polynomial complexity, hence cannot be guessed directly. Therefore, our second contribution is a dynamic program that guesses this structure (plus a packing of large and small tasks) *on the fly*, while losing at most one third of the profit of the remaining small tasks.

ACM Reference Format:
Fabrizio Grandoni, Tobias Mömke, Andreas Wiese, and Hang Zhou. 2018. A $(5/3 + \varepsilon)$-Approximation for Unsplittable Flow on a Path: Placing Small Tasks into Boxes. In *Proceedings of 50th Annual ACM SIGACT Symposium on the Theory of Computing (STOC'18).* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3188745.3188894

1 INTRODUCTION

Unsplittable Flow on a Path (UFP) is an important and well-studied problem. We are given an undirected path $G = (V, E)$ with a capacity $u(e) \in \mathbb{N}^+$ for each edge $e \in E$ and a set of n tasks T where each task $i \in T$ is characterized by a subpath $P(i)$ connecting its start (i.e., leftmost) vertex $s(i) \in V$ and its end (i.e., rightmost) vertex $t(i) \in V$, a demand $d(i) \in \mathbb{N}^+$, and a profit (or weight) $w(i) \geq 0$. Let T_e denote the subset of tasks i with $e \in P(i)$. Let also $w(T') := \sum_{i \in T'} w(i)$ and $d(T') := \sum_{i \in T'} d(i)$ for $T' \subseteq T$. The goal is to select a subset of tasks $T' \subseteq T$ of maximum profit $w(T')$ such that, for each edge e, the total demand $d(T' \cap T_e)$ of the selected tasks using e does not exceed $u(e)$.

The problem has many applications in areas such as bandwidth allocation [8, 16, 22], multi-commodity demand flow [15], caching [17], scheduling [4], and resource allocation [7, 11, 18, 23]. For instance, one can interpret the path as a communication channel and the tasks as requests to use a portion of the available bandwidth on some subpath. From a scheduling perspective, the edges on the path can be seen as discrete time slots and the tasks as jobs that request to run during certain time intervals and that need certain portions of a shared time-varying resource during their execution. Also note that if the path E consists of a single edge only, we obtain the knapsack problem. Thus UFP generalizes the latter and hence it is weakly NP-hard; and it is also known to be strongly NP-hard [10,

[*]Partially supported by ERC Starting Grant NEWNET 279352 and SNSF Grant APXNET 200021_159697/1.
[†]Partially supported by Deutsche Forschungsgemeinschaft grant MO 2889/1-1.
[‡]Partially supported by the Lise Meitner Award Fellowship when the author was at the Max Planck Institute for Informatics.

STOC'18, June 25–29, 2018, Los Angeles, CA, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5559-9/18/06…$15.00
https://doi.org/10.1145/3188745.3188894

17]. The best known polynomial-time approximation algorithm for UFP has a ratio of $2 + \varepsilon$ [3]. This result, likewise most prior work [10, 11, 13, 15], is based on a classification of tasks as large and small. Intuitively, a small task i has a demand $d(i)$ which is at most an ε^2-fraction of the capacity $u(e)$ of any edge $e \in P(i)$. For the small tasks there is a $(1 + O(\varepsilon))$-approximation algorithm via LP-rounding [15], and for the large tasks there is a $(1 + \varepsilon)$-approximation algorithm via dynamic programming [3]. Hence by taking the best of the two solutions, one obtains a $(2 + O(\varepsilon))$-approximation. However, there are no non-trivial polynomial time algorithms known that compute solutions with (a big number of both) large *and* small tasks together. On the other hand, there is a quasi-polynomial time approximation scheme (QPTAS) [5, 9], i.e., a $(1 + \varepsilon)$-approximation algorithm with a running time of $n^{\text{poly}(\log(n))}$ for any constant $\varepsilon > 0$. This gives hope to achieve better approximation ratios in polynomial time, possibly even a PTAS. The latter is considered an important open problem.

1.1 Our results and techniques

A major obstacle for constructing a PTAS for UFP is that one needs to compute solutions that combine large and small tasks with high profit together. In this paper, we overcome this obstacle and demonstrate how to compute such solutions efficiently. As a consequence we break for the first time the approximation barrier of 2 for polynomial time algorithms and general instances, achieving the following result.

Theorem 1. *There is a deterministic polynomial-time* $(5/3 + \varepsilon)$-*approximation algorithm for UFP.*

We next describe our approach in more detail. Our starting point is a different classification of tasks into large and small, which is based on prior work by the authors of this paper [21] plus some additional ideas. Let opt_L and opt_S denote the profit of large and small tasks, resp., in the optimal solution. The LP-rounding approach of Chekuri, Mydlarz, and Shepherd [15] allows us to compute a solution of value at least $(1-O(\varepsilon))\text{opt}_S$ using small tasks only. The main contribution of this paper is a polynomial-time dynamic program (DP) that computes a solution whose profit is essentially at least $\text{opt}_L + \text{opt}_S/3$. It then follows easily that the claimed approximation ratio can be obtained by combining our DP with the algorithm in [15].

One of our key ideas is to define *boxed solutions* in which the small tasks are packed in a very structured way into *boxes*, see Fig. 1. Suppose we want to compute a fractional solution where large tasks are taken integrally. In particular, for a small task i, we are allowed to pack a fraction $x_i \in [0, 1]$ of i using capacity $x_i \cdot d(i)$ on each edge of $P(i)$ and providing a profit $x_i \cdot w(i)$. It is not a problem that the small tasks are taken fractionally since we can round them at the end with a negligible loss.

We define a collection of boxes, where a box B is specified by a subpath $P(B)$ and a capacity $u(B)$. Box paths induce a laminar family. There exists a geometric packing of boxes inside the capacity profile, where the horizontal position of each box B is determined by $P(B)$, and its vertical position is induced by the laminar family: the roots of the laminar family are placed at the bottom of the capacity profile, and the remaining boxes on top of them as low as possible in a recursive way. The space left free by the boxed packing is sufficient to accommodate (non-geometrically) all large tasks, hence providing a profit of opt_L.

Small tasks are fractionally packed inside boxes, so that the fractional tasks T' assigned to box B define a feasible solution on path $P(B)$ with uniform capacity $u(B)$ (in particular, for each $i \in T'$, $P(i) \subseteq P(B)$). We show that there exists a fractional packing of this form of profit roughly $\text{opt}_S/2$. The key idea to show the existence of such a boxed packing is a novel *horizontal slicing lemma* (see also Fig. 2). A common geometric visualization of a feasible UFP solution T' is as follows. Imagine each task $i \in T'$ as a rectangle $R(i)$ with base sitting on $P(i)$ and with height given by $d(i)$. Suppose that you are allowed to slice vertically each $R(i)$ into pieces and then translate vertically each piece. Then there exists a feasible *geometric packing* of the vertical slices inside the capacity profile, which follows trivially from the definition of feasibility. Instead, we exploit an alternative geometric viewpoint: imagine that each $R(i)$ can only be sliced horizontally, and then horizontal slices can be translated vertically. We show that a geometric packing of horizontal slices exists, *provided* that T' uses *at most one half* of the capacity on each edge, fractionally or integrally.

Using the above lemma, we proceed roughly as follows. We select all small tasks from OPT (fractionally) to an extent of $1/2$ each, losing a factor 2 in the profit of small tasks. Thus they use only one half of the capacity of each edge e, or more precisely, at most $f(e)/2$ where $f(e)$ is the capacity used by the small tasks of OPT on e. Hence, we can invoke the horizontal slicing lemma to find a fractional geometric packing within the profile given by f. With an additional rounding step we can assume that each $f(e)$ is zero or a power of $1+\varepsilon$. The boxes are then obtained by horizontally slicing the capacity profile f according to horizontal lines at heights $(1 + \varepsilon)^i$, for all $i \in \mathbb{N}$ (that implies a further slicing of slices at the boundary between boxes). The slices inside each box naturally induce a fractional solution.

Let OPT′ be the fractional boxed solution derived from the above discussion, and let opt'_L and opt'_S be the corresponding profit associated with large and small tasks, resp. It remains to compute a boxed solution of sufficiently high value compared with OPT′: note that this is a non-trivial task since the boxed structure is unknown and guessing it in one step would take super-polynomial time. The rough idea is as follows. Suppose that, given a box B, we are able to guess the boxes $B_1, \ldots, B_q$ right above B. Suppose also that we are able to guess all the large tasks $L(B)$ whose path is contained in $P(B)$ but not in any $P(B_i)$. It is not hard to define a dynamic program that scans the edges of $P(B)$ from left to right and, in some sense, performs this guessing for us.

Then we might proceed as follows. We start with any root box B, and compute an optimal fractional solution $x(B)$ for B over small tasks with path in $P(B)$. Then we solve recursively the subproblems induced by each box B_j above B. In each such subproblem we need to remember the tasks of $L(B)$ that use the leftmost or rightmost edge of B_j (*boundary* large tasks). Furthermore, we have to remember that each small task i can be used only up to an amount of $1 - x_i(B)$. Clearly, in the following recursive steps on some box B we will have to remember all the *ancestor* boxes $\mathcal{B}$ whose path contains $P(B)$, with the associated fractional solutions, plus all the boundary large tasks induced by those boxes. This high-level approach has three main difficulties that we need to address:

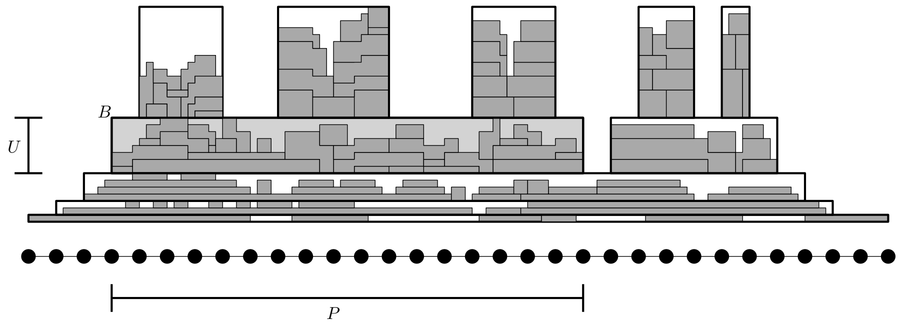

Figure 1: The boxes and small tasks of a boxed solution. The light gray box B has capacity U and uses the path P.

- We might select small tasks in a sub-optimal way w.r.t. OPT′ since we compute the fractional solutions for the boxes step by step in the order given by the laminar family, instead of computing a (global) solution for all boxes simultaneously in one step. Using a simple but non-trivial charging argument, we show that despite this issue our DP obtains a profit of at least $\frac{2}{3}\text{opt}'_S$ from the small tasks.
- The laminar family can contain chains (i.e., sequences of boxes one on top of the next one) of unbounded length. This implies that, for a given box B, we are not able to remember all the boundary large tasks induced by ancestor boxes $\mathcal{B}$. We show that it is possible to create some slack capacity on each edge so that from time to time we are allowed to *forget* boundary large tasks of sufficiently small demand. This way, we need to remember only $O_\varepsilon(1)$ boundary large tasks for each box.
- For the same reason as above, we are also not able to remember all the fractional solutions associated with ancestor boxes $\mathcal{B}$ of B. Also in this case, we are allowed to forget the fractional solution of ancestor boxes that appear *much below* B: this leads to a slight overcounting of the profit of small tasks that, however, we are able to keep under control within our charging scheme.

The definition of boxes and small tasks guarantees that we can round small tasks at the end of the process with a small loss in the profit using ideas in [15], and thus obtain an integral solution.

1.2 Other related work

Prior to the mentioned $(2+\varepsilon)$-approximation algorithm for UFP [3] there were algorithms known for the problem with this approximation guarantee for the special cases of uniform edge capacities [11] and under the no-bottleneck-assumption [15] (which requires that $\max_{i\in T} d(i) \le \min_{e\in E} u(e)$), as well as algorithms with approximation ratios of $O(\log n)$ [6] and $(7+\varepsilon)$ [10] for the general case. Somewhat interestingly, our approach does not seem to benefit (in terms of a better approximation factor) from uniform edge capacities.

There are PTASes known for the special cases when the demand of each task is proportional to its weight [9]; when one can slightly shorten the path of each input task [9]; when there is an edge used by all input tasks [21]; when the profit of each task i is proportional to its "area" $|P(i)| \cdot d(i)$ [21]; and when one can select each task an arbitrary number of times [21]. The integrality gap of LP relaxations for UFP is studied, among others, in [2, 14].

For *Unsplittable Flow on a Tree*, i.e., the natural generalization of UFP where the input graph is a tree, the best known result is an $O(k \cdot \log n)$-approximation [1] where k denotes the path-width of the tree (which is bounded by $O(\log n)$). This result holds even for submodular objectives. Thus the result refines and generalizes the previously known $O(\log^2 n)$-approximation [14] for linear objectives. Under the no-bottleneck-assumption there is a 48-approximation [15]. Unlike for paths, the problem on trees is known to be APX-hard even for unit demands and trees of depth 3 whose edges have capacities of either 1 or 2 [19].

Another interesting generalization of UFP is *bag UFP*, where tasks are partitioned into subsets (*bags*), and at most one task per bag can be selected. This captures situations where a task can be scheduled at different times [1, 12, 20].

2 OVERVIEW AND TASK CLASSIFICATION

Our starting point is a classification of tasks which is partially inspired by prior work of the authors of this paper [21]. We sketch the basic ideas and formally define the properties that we will use in the rest of the paper. The technical details (which do not involve substantially new ideas) are given in Appendix A.

In [21] it is shown how to define an almost optimal solution OPT_{slack} that leaves on each edge e a certain amount of slack $\delta_e \ge 0$ such that there are only few tasks in $\text{OPT}_{slack} \cap T_e$ that are relatively large compared to δ_e. Such tasks are defined to be the *large* tasks. All other tasks in $\text{OPT}_{slack} \cap T_e$ have a total capacity of at most $O(\delta_e/\varepsilon^3)$ and are defined to be the *small* tasks[1]. Starting with this classification and applying some shifting arguments, we can prove the following lemma, where, roughly speaking, $f(e)$ is the capacity used by the small tasks in $\text{OPT}_{slack} \cap T_e$.

Lemma 2. *Let $\varepsilon > 0$ be an arbitrary constant. For a UFP instance with optimal value* opt, *there exists disjoint subsets of tasks* $\text{OPT}_L \subseteq T$ *and* $\text{OPT}_S \subseteq T$, *a value $f(e) \ge 0$ for each edge e, and two constants $\mu_1, \mu_2 \in (0, \varepsilon^4)$ with $\mu_1 < \mu_2/(1+\varepsilon)^{1/\varepsilon^3}$ such that the following holds. Let*

$$T_L = \{i \in T : d(i) \ge \mu_2 \cdot f(e) \text{ for some edge } e \in P(i)\} \quad \text{(large tasks)}$$

$$T_S = \{i \in T : d(i) < \mu_1 \cdot f(e) \text{ for every edge } e \in P(i)\} \quad \text{(small tasks)}$$

[1]Throughout this paper, whenever needed and w.l.o.g., we assume that ε is sufficiently small and that $1/\varepsilon$ is integral.

Then:

(1) $\mathrm{OPT}_L \subseteq T_L$ *and* $\mathrm{OPT}_S \subseteq T_S$*;*
(2) $w(\mathrm{OPT}_L \cup \mathrm{OPT}_S) \geq (1 - O(\varepsilon))\mathrm{opt}$*;*
(3) μ_1 *and* μ_2 *are contained in a set of size* $O_\varepsilon(1)$*;*

Furthermore, for every edge e, the following properties hold:

(4) $f(e) \in \{(1+\varepsilon)^i : i \in \mathbb{N}_0\} \cup \{0\}$*;*
(5) *There are at most* $1/(\mu_2 \cdot \varepsilon^4)$ *tasks* $i \in T_e \cap \mathrm{OPT}_L$ *such that* $d(i) \geq \mu_2 \cdot f(e)$*;*
(6) *The total demand of tasks* $T_e \cap \mathrm{OPT}_S$ *is at most* $f(e)$*.*
(7) $f(e) + d(T_e \cap \mathrm{OPT}_L) \leq u(e) - \varepsilon^4 \cdot f(e)$*.*

Note that some of the tasks are neither small nor large. However, by choosing μ_1 and μ_2 appropriately, we can ensure that they can be discarded with a small loss of the profit.[2]

Let $\mathrm{opt}_L := w(\mathrm{OPT}_L)$ and $\mathrm{opt}_S = w(\mathrm{OPT}_S)$. In order to obtain our $(5/3+\varepsilon)$-approximation algorithm, we use two subroutines, yielding different approximation guarantees with respect to opt_L and opt_S. For the small tasks, we use a result in [15] to achieve the following.

Lemma 3. *There is a deterministic polynomial-time algorithm that computes a solution with profit at least* $(1 - O(\varepsilon)) \cdot \mathrm{opt}_S$*.*

Proof. In [15] the authors define $\mathcal{P}(A, W)$ to be the class of problems of the form $\max\{wx : Ax \leq b, x \in [0,1]^n\}$ for some $\{0,1\}$-matrix A, a vector $w \in W$ and a vector $b \in \mathbb{Z}^m$. Moreover, they define $\mathcal{P}^{dem}(A, W)$ to be the class of problems of the form $\max\{wx : A[d]x \leq b, x \in [0,1]^n\}$ for some $w \in W$ and vectors $b \in \mathbb{Z}^m$, $d \in \mathbb{Z}_+^n$ and $d_{\max} \leq b_{\min}$ with $d_{\max} = \max_{i\in[n]} d_i$ and $b_{\min} = \min_{i\in[m]} b_i$ and where $A[d]$ is obtained from taking a $\{0,1\}$-matrix and multiplying the entry in each column i by d_i. Additionally, they define $\mathcal{P}^{\varepsilon dem}(A, W)$ to be the class of problems where additionally $d_{\max} \leq \varepsilon b_{\min}$. A collection of vectors $W \subseteq \mathbb{Z}^n$ is defined to be *closed* if for any vector $w \in W$ the vector w' obtained by setting some $w_j = 0$ is also in W. Based on these definitions, they prove the following lemma.

Lemma 4 (Corollary 3.4 in [15]). *Let A be a* $\{0,1\}$ *matrix and W be a closed collection of vectors. If the integrality gap for the collection of problems* $\mathcal{P}(A, W)$ *is at most* Γ*, then the integrality gap for the collection of problems* $\mathcal{P}^{\varepsilon dem}(A, W)$ *for* $\varepsilon < (3-\sqrt{5})/2$ *is at most* $\frac{1+\sqrt{\varepsilon}}{1-\sqrt{\varepsilon}-\varepsilon}\Gamma$*. This holds even under the weaker condition that* $\max_j A_{ij}d_j \leq \varepsilon b_i$ *for each* $i = 1, 2, \ldots, m$*.*

For UFP, the respective classes of problems $\mathcal{P}(A, W)$ represent UFP-instances in which all tasks have unit demand. Using flow-arguments one can show that the LP-relaxation for such instances is exact and hence $\Gamma = 1$ in our case. The condition that $\max_j A_{ij}d_j \leq \varepsilon b_i$ for each $i = 1, 2, \ldots, m$ replaces the conditions that $d_{\max} \leq \varepsilon b_{\min}$ and $d_{\max} \leq b_{\min}$ and hence the integrality gap for UFP-instances in which for each task i it holds that $d(i) \leq \varepsilon u(e)$ for each $e \in P(i)$ is bounded by $\frac{1+\sqrt{\varepsilon}}{1-\sqrt{\varepsilon}-\varepsilon}$. Hence, if for each task i it even holds that $d(i) \leq \varepsilon^2 u(e)$ for each $e \in P(i)$ then the integrality gap is even bounded by $\frac{1+\varepsilon}{1-\varepsilon-\varepsilon^2} \leq 1 + 3\varepsilon$ for a sufficiently small ε, in particular, it is necessary that $\varepsilon^2 < (3-\sqrt{5})/2$.

[2]The parameters μ_1 and μ_2 are unknown and depend on the optimal solution. However, in our algorithm, we can enumerate all possibilities of μ_1 and μ_2, and there are only $O_\varepsilon(1)$ many options according to Lemma 2.

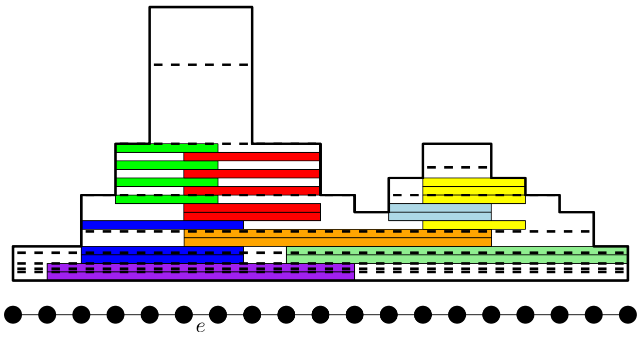

Figure 2: The rectangles represent slices of tasks that are drawn non-overlappingly within the capacity profile. Note that the solution uses only half of the capacity of *e*, however, if the tasks in the figure using *e* had more slices, we could not draw their slices as non-overlapping rectangles any more. This is the reason why we require that the tasks use only half of the available capacity on each edge. The horizontal lines represent the boundaries of the boxes.

Note that in [14] a statement for larger values for ε is shown: the authors prove that for any fixed $\delta > 0$ the integrality gap is at most $O(\log(1/\delta)/\delta^3)$ if for each task i we have that $d(i) \leq (1-\delta)u(e)$ for each $e \in P(i)$. Hence, this statement allows that $1-\delta$ is arbitrarily close to 1 and guarantees a constant integrality gap for any fixed δ. Note however that this statement does not immediately imply that the integrality gap is $1 + O(\varepsilon)$ if $d(i) \leq \varepsilon^2 u(e)$ for each $e \in P(i)$. □

If $\mathrm{opt}_S \geq (3/5)\cdot\mathrm{opt}$ then Lemma 3 yields a $(5/3+\varepsilon)$-approximation. For instances where $\mathrm{opt}_S < (3/5)\cdot\mathrm{opt}$ and hence $\mathrm{opt}_L \geq (2/5)\cdot\mathrm{opt}$ we obtain a $(5/3+\varepsilon)$-approximation from the following lemma. By taking the best of the two solutions we obtain Theorem 1.

Lemma 5. *There is a deterministic polynomial-time algorithm that computes a solution with profit at least* $\mathrm{opt}_L + (1/3 - O(\varepsilon))\mathrm{opt}_S$*.*

We devote the rest of the paper to prove Lemma 5. In Section 3, we show that there exists a structured solution that we call a *boxed solution* which yields a profit of at least $\mathrm{opt}_L + (1/2 - O(\varepsilon)) \cdot \mathrm{opt}_S$. In Section 4, we present a DP that computes a boxed solution with a (smaller, but still large enough) profit of at least $\mathrm{opt}_L + (1/3 - O(\varepsilon)) \cdot \mathrm{opt}_S$.

3 BOXED SOLUTIONS

In this section we transform OPT into a (fractional) solution OPT′ with profit at least $\mathrm{opt}_L + (1/2 - O(\varepsilon)) \cdot \mathrm{opt}_S$. In addition, OPT′ is a *boxed solution* as defined later.

Before defining OPT′, we present a technical lemma that might be useful also for related problems. Suppose we are given a fractional solution x defined via a value $x_i \in [0,1] \cap \mathbb{Q}$ for each task i. Consider a rectangle $R(i)$ for each task i with height $x_i d(i)$ and with horizontal coordinates $s(i)$ and $t(i)$ where we assume that the vertices of G correspond to the coordinates $1, 2, \ldots, |V|$. We will define vertical coordinates later. We slice these rectangles horizontally into uniform *slices*. Formally, we cut each rectangle $R(i)$ into a set

$\mathcal{R}(i)$ of $x_i d(i)/\Delta$ smaller rectangles, each with height Δ and with the same horizontal coordinates as $R(i)$, where $\Delta \geq 0$ is the largest value such that $x_i d(i)/\Delta \in \mathbb{N}$ for each i. For each task i, we want to assign a vertical coordinate h_{ij} to each slice $R_j \in \mathcal{R}(i)$ such that $h_{ij} + \Delta \leq u(e)$ for each $e \in P(i)$ and such that all slices of all tasks are pairwise non-overlapping, apart from their boundaries possibly, see Figure 2. So intuitively, all slices are packed within the capacity profile. We call such a height-assignment $\left\{h_{ij}\right\}_{i\in T_S, R_j\in\mathcal{R}(i)}$ a *horizontally-sliced geometric packing of* x. If all edges of G have the same capacity, the existence of such a packing follows easily from the fact that interval graphs are perfect graphs and hence the clique and the coloring numbers coincide [24]. If the edges of G have different capacities, then such an assignment might not exist, even if x is a feasible fractional solution, see Figure 2. However, we prove that it exists if the fractional solution x uses at most half of the capacity of each edge.

Lemma 6 (Horizontal slicing lemma). *Let x be a fractional solution for a given UFP instance that satisfies $\sum_{i\in T_e} x_i \cdot d(i) \leq u(e)/2$ for each edge e. Then there exists a horizontally-sliced geometric packing of x.*

Proof. Assume w.l.o.g. that $\Delta = 1$ which can be achieved by scaling all edge capacities and task demands by $1/\Delta$. We will show how to pack all slices (which then have unit height). Formally, we show how to solve inductively the following problem. Consider any subpath $E' \subseteq E$ with leftmost edge e_L and rightmost edge e_R such that $u(e) \geq \max\{u(e_L), u(e_R)\}$ for every edge $e \in E'$ strictly between e_L and e_R. Let S' be the set of slices that intersect E'. Then there is an assignment h of an integer height $h_j \geq 0$ to each slice $j \in S'$ such that: (1) for any slice j, $h_j + 1 \leq \min_{e\in E'\cap P(j)} u(e)$ and (2) for any two slices $j, j' \in S'$ with $h_j = h_{j'}$ it holds that $P(j) \cap P(j') = \emptyset$.

The lemma statement follows from the claim when E' equals the entire path E. (By appending two dummy edges e_L and e_R with capacity 0 at the endpoints of E, we can ensure that $u(e) \geq \max\{u(e_L), u(e_R)\}$ for every edge $e \in E$ strictly between e_L and e_R.)

The proof of the claim is by induction. For the base case of the induction, E' consists of two edges e_L and e_R (the claim is clearly true if E' has only one edge). We consider each slice j that contains both e_L and e_R (*middle slice*) and slice it vertically at the middle point v_{middle} between the two edges, hence getting two slices j_{left} and j_{right}. We denote by *pieces* the resulting set of slices (consisting of the two parts of middle slices plus the remaining slices). We now sort arbitrarily the pieces to the left of v_{middle}, and we assign the (odd) height $2j-1$ to the j-th piece in this order, $j \geq 1$. We remark that these heights are upper-bounded by $u(e_L)-1$ since by assumption the total height of slices using e_L is at most $u(e_L)/2$. Similarly, we sort arbitrarily the pieces to the right of v_{middle} and we assign the (even) height $2j-2$ to the j-th piece in this order, $j \geq 1$. Note that we might assign different heights to the two pieces of the same middle slice: we resolve this conflict by re-assigning to both those pieces the minimum of the two mentioned heights. Note that the odd (resp., even) heights over edge e_R (resp., e_L) correspond to free regions. Thus property (2) is satisfied. Since the original heights satisfy property (1) and we only move slices down, property (1) is also satisfied at the end of the process.

Consider next the inductive step (with more than 2 edges). Let $e_M = (a, a+1)$ denote an edge with smallest capacity strictly between e_L and e_R. Denote by S_M the slices containing e_M that we call the *middle slices*. We split each $s \in S_M$ vertically at the middle coordinate $v_{\text{middle}} = a + 1/2$ into two slices j_{left} and j_{right} similarly to the base case. We again denote by *pieces* the resulting set of slices. We consider separately the pieces to the left and right of v_{middle}, and use the inductive hypothesis to derive an assignment of heights h^{left} and h^{right} to the left and right pieces, respectively. Also in this case, we need to resolve conflicts due to different heights of pieces of the same middle slice. This fixing is slightly more involved than in the base case. For each height h' of any left piece we define one left *block* as the minimum-width unit-height slice that contains all the left pieces at height h'. We similarly define right blocks.

Observe that each piece is fully contained in some block: we will modify the heights of the blocks so that blocks containing the two pieces of the same middle slice have the same height and blocks are non-overlapping: this naturally induces a packing of the slices that satisfies (2). We say that a block is critical if it contains a piece of a middle slice. We temporarily remove all non-critical blocks, that will be re-packed later. We assign odd heights to the critical left blocks similarly to the base case, however considering first the critical left blocks B_L^{crit} spanning edge e_L, and then the remaining critical left blocks $\overline{B}_L^{\text{crit}}$. Note that the blocks B_L^{crit} will have a height of at most $u(e_L)-1$ since they occupy at most half of the capacity $u(e_L)$, and the blocks $\overline{B}_L^{\text{crit}}$ will have a height of at most $u(e_M)-1$ since all critical blocks together occupy at most half of the capacity $u(e_M)$. We symmetrically pack the critical right blocks. Next, similarly to the base case, whenever two critical blocks containing pieces of the same middle slice have different heights, we assign to both blocks the minimum of those two heights. By the same argument as in the base case the geometric packing of critical blocks remains feasible and the pieces corresponding to critical blocks satisfy (1).

We can now pack independently the left and right non-critical blocks: we focus on the left such blocks, the other case being symmetric. Let us further partition left non-critical blocks into the *top* ones, whose pieces have height at least $u(e_M)$ in h^{left}, and the remaining *bottom* ones. We assign the corresponding height in h^{left} to the top left non-critical blocks. The pieces corresponding to these blocks satisfy (1) by inductive hypothesis.

The bottom left non-critical blocks are packed from bottom to top as low as possible, considering first the blocks $B_L^{\text{n-crit}}$ spanning e_L, and then the remaining blocks $\overline{B}_L^{\text{n-crit}}$. Note that $|B_L^{\text{crit}} \cup B_L^{\text{n-crit}}| \leq u(e_L)/2$ by construction and assumption: hence the heights between 0 and $u(e_L)-1$ not assigned to B_L^{crit} are more than sufficient to accommodate $B_L^{\text{n-crit}}$. Similarly, $|B_L^{\text{crit}} \cup B_L^{\text{n-crit}} \cup \overline{B}_L^{\text{crit}} \cup \overline{B}_L^{\text{n-crit}}| \leq u(e_M)$ by construction, hence the heights between 0 and $u(e_M)-1$ not assigned to $B_L^{\text{crit}} \cup B_L^{\text{n-crit}} \cup \overline{B}_L^{\text{crit}}$ are sufficient to accommodate $\overline{B}_L^{\text{n-crit}}$. Thus the pieces of bottom left non-critical blocks satisfy (1). Furthermore, since the geometric packing of blocks is globally feasible, all pieces satisfy (2). □

Let us remark that the factor 2 in the above lemma is in some sense tight: to see that, consider three consecutive edges with capacities 1, 2, and 1 from left to right, and a feasible integral solution consisting of 2 tasks of demand 1 and spanning the leftmost two edges and rightmost two edges, resp.

Based on OPT_S, OPT_L, and f we construct now a solution in which the small tasks are assigned fractionally into *boxes*, as defined below.

Definition 7 (Types). *An edge e is of* type $j \in \mathbb{N}_0$ *if $f(e) = (1+\varepsilon)^j$ and we write* $\mathrm{type}(e) = j$. *If $f(e) = 0$ then we define* $\mathrm{type}(e) = -1$. *The* type *of a task $i \in T$ is the minimum edge type among all edges on $P(i)$, i.e.,* $\mathrm{type}(i) = \min_{e \in P(i)} \mathrm{type}(e)$.

Given an assignment of a type to each edge, we say that a type-j task $i \in T$ is *small* (resp. *large*) if $d(i) < \mu_1 \cdot (1+\varepsilon)^j$ (resp. $d(i) \geq \mu_2 \cdot (1+\varepsilon)^j$), and denote the resulting set by T_S (resp. T_L). This definition is equivalent to the definition in Lemma 2.

Definition 8 (Boxes). *A* box *B is a pair $(P(B), \mathrm{type}(B))$ consisting of a subpath $P(B)$ of the graph G and a type $\mathrm{type}(B) \in \mathbb{N}_0$. It has a capacity of $u(B)$, which is defined as $u(B) = 0$ if $\mathrm{type}(B) = -1$, $u(B) = 1$ if $\mathrm{type}(B) = 0$, and $u(B) = (1+\varepsilon)^{\mathrm{type}(B)} - (1+\varepsilon)^{\mathrm{type}(B)-1}$ otherwise.*

One can naturally define a notion of feasibility for a fractional solution x w.r.t. a box B by interpreting B as a UFP instance on path $P(B)$ and with uniform edge capacity $u(B)$.

To define the boxes for our instance, the reader may imagine that we take the profile given by f (with values that are powers of $1+\varepsilon$) and draw horizontal lines with y-coordinates $1, 1+\varepsilon, (1+\varepsilon)^2, \ldots$ into it. Formally, for each $j \in \mathbb{N}_0 \cup \{-1\}$ and for every maximally large subpath P of G in which each edge is of type j or larger, we introduce a box $B = (P, j)$. As a result, an edge e of type $j \geq 0$ (with thus $f(e) = (1+\varepsilon)^j$) is used by one box of each type $-1, \ldots, j$ and by no box of type greater than j. The total capacity of these boxes is $(1+\varepsilon)^j$. Observe that the paths of the boxes form a laminar family, i.e., for any two boxes $(P_1, j_1), (P_2, j_2)$ we have $P_1 \subseteq P_2$ or $P_2 \subseteq P_1$ or $P_1 \cap P_2 = \emptyset$. Given an arbitrary assignment of a type to each edge, we say that a set of boxes and the sets of large and small tasks T_L and T_S are *induced* by the edge types when defined according to our definitions above.

Definition 9 (Boxed solution). *A* boxed solution *consists of an assignment* $\mathrm{type}\colon E \to \mathbb{N}_0 \cup \{-1\}$ *of types to edges, a set of boxes $\mathcal{B}$ and sets T_L and T_S of large and small tasks induced by these edge types, a subset $S_L \subseteq T_L$ of large tasks, and for each box B a feasible fractional solution $\{x_i(B)\}_{i \in T_S}$ for the small tasks such that:*

1. *for each edge e, the total demand of the large tasks from $S_L \cap T_e$ plus the total capacity of the boxes from $\mathcal{B}$ using e is at most $u(e)$;*
2. *each edge of type j is used by at most $1/(\mu_2 \varepsilon^4)$ large tasks $i \in S_L$ such that $d(i) \geq \mu_2 (1+\varepsilon)^j$;*
3. *for every box $B = (P, j) \in \mathcal{B}$, $d(i) < \mu_2 (1+\varepsilon)^j \leq \varepsilon^2 u(B)$ for each $i \in T_S$ if $x_i(B) > 0$;*
4. *For every small task $i \in T_S$, $\sum_{B \in \mathcal{B}, i \in T_S} x_i(B) \leq 1$.*

Lemma 10. *There exists a boxed solution OPT' with profit $\mathrm{opt}' \geq \mathrm{opt}_L + (1/2 - O(\varepsilon)) \cdot \mathrm{opt}_S$ such that for each edge e of type $j \geq 0$, there is an amount of $\varepsilon^4 \cdot (1+\varepsilon)^j$ unused capacity, i.e., the total demand of the large tasks from $S_L \cap T_e$ plus the total capacity of the boxes using e is at most $u(e) - \varepsilon^4 \cdot (1+\varepsilon)^j$.*

Proof. We define the assignment type based on the function f due to Lemma 2 and let $\mathcal{B}$ be the set of boxes induced by type. We define $S_L := \mathrm{OPT}_L$. The first two properties of a boxed solution as well as the additional property in Lemma 10 follow from Lemma 2.

It only remains to construct the set of fractional solutions $x(B)$ so that Property 3 of a boxed solution holds. We start from OPT_S. By taking a fraction 1/2 of each such task, we obtain a fractional solution $\{y_i\}_{i \in T_S}$ satisfying the conditions of the horizontal slicing Lemma 6 with respect to the UFP instance induced by small tasks and edge capacities $f(e)$. Hence we obtain a horizontally-sliced geometric packing of y within the capacity profile f. We draw a rectangle $R(B)$ for each box $B = (P, j)$ where the left and right coordinates of $R(B)$ are given by the leftmost and rightmost vertices of P, and the bottom and top coordinates are $(1+\varepsilon)^{j-1}$ and $(1+\varepsilon)^j$ if $j \geq 1$ and 0 and 1 if $j = 0$. We further horizontally slice the slices which cross the box boundaries, so that each horizontal slice is fully contained in some box (and hence might have smaller height than Δ).

We next shrink and move some slices as follows. First of all, we shrink the heights of all slices by a factor $1+\varepsilon$. Next, for every type j, we consider all slices of type j that are assigned to boxes of types smaller than $j - (1/\varepsilon^3)$ and we move all such slices to type-j boxes. With a geometric sum argument one can easily show that these slices fit into the free space in type j boxes generated in the shrinking step.

For each box B the slices in B induce a fractional solution $x(B)$: for each task i we define the value $x_i(B)$ such that slices of a total height of $x_i d(i)$ are contained in B. Note that $x(B)$ is feasible for B by construction. Also, the total value $\sum_{B \in \mathcal{B}} \sum_{i \in T_S} x_i(B) w(i)$ of the fractional solutions $x(B)$ is at least $w(\mathrm{OPT}_S)/(2(1+\varepsilon))$ by construction. The tasks fractionally assigned to a box of type j have a type in $\{j, \ldots, j + (1/\varepsilon^3)\}$. In particular, each such task i has demand less than $\mu_1 \cdot (1+\varepsilon)^{j+(1/\varepsilon^3)}$ (by the definition of small tasks), which is less than $\mu_2 \cdot (1+\varepsilon)^j$ using the definition of μ_1 and μ_2. Hence, Property 3 of a boxed solution is satisfied. We define OPT' to be the union of S_L and the fractional solutions $x(B)$ for all boxes B. Then OPT' is a boxed solution satisfying all properties and its weight is at least $\mathrm{opt}_L + (1/2 - O(\varepsilon)) \cdot \mathrm{opt}_S$. □

4 AN IMPROVED APPROXIMATION ALGORITHM

In this section, we present a polynomial time algorithm that computes a feasible UFP solution with profit at least $\mathrm{opt}_L + (1/3 - O(\varepsilon)) \cdot \mathrm{opt}_S$. In order to simplify the presentation, we will assume that in OPT' for each edge e we have $1 \leq \mathrm{type}(e) \leq K$ for some constant $K = O_\varepsilon(1)$, and describe an algorithm for this case. In Section 4.3 we will generalize our construction to arbitrary edge types.

We recall that in the standard linear programming relaxation LP_{UFP} for UFP we have variables $x_i \in [0,1]$ for each task i, and we have to satisfy the capacity constraints $\sum_{i: e \in P(i)} x_i d(i) \leq u(e)$ for every edge e. The objective is to maximize $\sum_{i \in T} x_i w(i)$. Let $\mathrm{type}(B)$ denote the type of a given box B. We say that a task i is *B-large* (resp., *B-small*) if $P(i) \subseteq P(B)$ and $d(i) \geq \mu_2 \cdot (1+\varepsilon)^{\mathrm{type}(B)}$ (resp., $d(i) < \mu_2 \cdot (1+\varepsilon)^{\mathrm{type}(B)}$). The leftmost and rightmost edges of $P(B)$ are denoted by $\ell(B)$ and $r(B)$, resp. Given a collection of boxes $\mathcal{B}$ and a set of tasks T' (that are intuitively placed outside the boxes) we define their total *demand* w.r.t. edge e as $d(T' \cap T_e) + \sum_{B \in \mathcal{B}: e \in P(B)} u(B)$.

4.1 The Algorithm

We first use a DP to compute a feasible fractional solution. Next we round this solution to an integral one.

Our DP runs in phases, one for each type. Intuitively, in the j-th phase it places small tasks into the boxes of type j via LP-rounding. In particular, each type j box B is filled fractionally with B-small tasks, where some fractional portions of those tasks might have already been selected in previous phases (and we cannot select these portions again). Additionally, we guess the boxes of type $j+1$ that, figuratively, are placed on top of the boxes of type j, and the B-large tasks (which are not completely contained in $P(B')$ for any top box B'). Each box of type $j+1$ yields a subproblem on which we recurse.

More formally, we define a DP that we denote by DP_{main}. Each DP cell is indexed by a tuple $(B, S_L, \mathcal{B})$ where B is a box and intuitively, $\mathcal{B}$ are the boxes below B and S_L are the corresponding large tasks that we guessed in *previous* phases, and which interact with B. We allow only tuples that satisfy the following conditions:

(1) $B = B_j$ is a box with $\text{type}(B) = j \in \{-1, \dots, K\}$;
(2) $\mathcal{B} = \{B_{-1}, \dots, B_{j-1}\}$ is a collection of boxes with $\text{type}(B_i) = i$ for all i and $P(B_{-1}) \supseteq \dots \supseteq P(B_j)$;
(3) $|S_L| \le M$, where $M := 2(K+1)/(\mu_2 \cdot \varepsilon^4)$;
(4) S_L consists of B'-large tasks i for some $B' \in \mathcal{B}$ with $P(i) \cap \{\ell(B), r(B)\} \neq \emptyset$ and $P(i) \not\subseteq P(B)$;
(5) The total demand of S_L and $\mathcal{B} \cup \{B\}$ w.r.t. to every $e \in P(B)$ is at most $u(e)$.

We associate a *bottom-up* fractional solution to B as follows. First, we define a fractional solution $x(B_{-1})$ for the box B_{-1} (with zero capacity) to be $x_i(B_{-1}) = 0$ for each task i. Then, for each $k \le j$ we compute an optimal fractional solution of small tasks for B_k disallowing to select the fractions of the tasks that are already assigned to the boxes $B_{k'}$ with $k' < k$. Formally, inductively for $k = 0, \dots, j$ we let $x(B_k)$ be an optimal fractional solution to LP_{UFP} (1) over B_k-small tasks, (2) with uniform capacity $u(B_k)$, and (3) with the additional constraints that $x_i(B_k) \le 1 - \sum_{k'=-1}^{k-1} x_i(B_{k'})$ for any task i. Observe that, for any task i and box B_k, $x_i(B_k) \le 1$.

The value of $DP_{main}(B, S_L, \mathcal{B})$ is defined as follows. Intuitively, we consider any possible way to place boxes $\mathcal{B}^{top}$ directly on top of B, and any possible set of large tasks contained in $P(B)$ but not completely within the path of any box from $\mathcal{B}^{top}$. The profit is given by the value of the large tasks, the LP value of $x(B)$ associated with B, and the sum of the DP values for the problems induced by each box from $\mathcal{B}^{top}$. Formally,

$$DP_{main}(B, S_L, \mathcal{B}) \leftarrow \max_{\mathcal{B}^{top}, S_L^{top}} \Big\{ w(S_L^{top}) + \sum_i x_i(B) w(i)$$
$$+ \sum_{k=1}^{q} DP_{main}(B_k^{top}, (S_L^{top} \cup S_L) \cap (T_{\ell(B)} \cup T_{r(B)}), \mathcal{B} \cup \{B\}) \Big\}$$

where the maximum is taken over:

(1) $\mathcal{B}^{top} = \{B_1^{top}, \dots, B_q^{top}\}$, $0 \le q \le |P(B)|$, for boxes $B_i^{top} = (P_i^{top}, \text{type}(B)+1)$ where $P_i^{top} \subseteq P(B)$ and P_i^{top} is entirely to the left of P_{i+1}^{top} for each $i = 1, \dots, q-1$ (we require that $\mathcal{B}^{top} = \emptyset$ if $\text{type}(B) = K$);
(2) S_L^{top} is a collection of B-large tasks i where $P(i) \not\subseteq P_j^{top}$ for any j, such that $|(S_L \cup S_L^{top}) \cap T_e| \le M$ for every $e \in P(B)$;
(3) the total demand of $S_L \cup S_L^{top}$ and $\mathcal{B} \cup \{B\} \cup \mathcal{B}^{top}$ w.r.t. each $e \in P(B)$ is at most $u(e)$.

Denote by ALG the fractional solution which corresponds to the value of $DP_{main}((G, -1), \emptyset, \emptyset)$. Observe that ALG might not be a feasible boxed solution, however it is a feasible UFP solution.

Unfortunately, we cannot compute the values of DP_{main} directly using the above formula since there exists an exponential number of choices for the pairs $(\mathcal{B}^{top}, S_L^{top})$. We will therefore use a secondary DP, next called DP_{sub}, for this goal. Intuitively, we scan the edges e_r of B from right to left, and each time guess whether there exists some box B^{top} with $r(B^{top}) = e_r$. Furthermore, we guess a set of large tasks using the edge e_r.

Formally, we label the edges of G with integers from 1 to $n-1$ from left to right (so that $e-1$ is the edge to the left of e etc.). We define $T_e = \emptyset$ if $e \le 0$ or $e \ge n$. A cell of DP_{sub} is indexed by a tuple $(B, S_L, \mathcal{B}, e_r, L_r)$, where $(B, S_L, \mathcal{B})$ is restricted in the same way as for DP_{main}. Furthermore $e_r \in P(B)$ and L_r is a set of at most M B-large tasks whose path contains e_r, or $e_r \in \{\ell(B)-1, r(B)+1\}$ in which case L_r is irrelevant. For each fixed $(B, S_L, \mathcal{B})$ we fill in the values of DP_{sub} by considering the edges $e_r \in P(B) \cup \{\ell(B)-1, r(B)+1\}$ *from left to right* and we assume that we already computed all values for the cells $DP_{main}(B', S'_L, \mathcal{B}')$ with $\text{type}(B') > \text{type}(B)$.

The base case is $DP_{sub}(B, S_L, \mathcal{B}, \ell(B)-1, L_r) = 0$. In the remaining cases $DP_{sub}(B, S_L, \mathcal{B}, e_r, L_r)$ is the maximum over the following two values. The first value intuitively corresponds to the case that there is no top box using edge e_r:

$$\max_{L_\ell} \{ w(L_r \setminus T_{e_r - 1}) + DP_{sub}(B, S_L, \mathcal{B}, e_r - 1, L_\ell) \}$$

where the maximum is taken over

- L_ℓ is a set of B-large tasks i with $e_r - 1 \in P(i)$ such that $|L_\ell| \le M$, and that L_ℓ is *consistent* with L_r, i.e., $L_\ell \cap T_{e_r} = L_r \cap T_{e_r - 1}$;
- $|(L_\ell \cup L_r \cup S_L) \cap T_e| \le M$ for every $e \in P(B)$;
- the total demand of $L_\ell \cup L_r \cup S_L$ and $\mathcal{B} \cup \{B\}$ w.r.t. every $e \in P(B)$ is at most $u(e)$.

The second value intuitively corresponds to the case that there is some top box ending at e_r:

$$\max_{B^{top}, L_\ell} \Big\{ w(L_r \setminus T_{\ell(B^{top})-1}) + DP_{main}\Big(B^{top},$$
$$(S_L \cup L_r \cup L_\ell) \cap (T_{\ell(B^{top}))} \cup T_{r(B^{top}))}), \mathcal{B} \cup \{B\}\Big)$$
$$+ DP_{sub}(B, S_L, \mathcal{B}, \ell(B^{top}) - 1, L_\ell) \Big\}$$

where the maximum is taken over:

- $B^{top} = (P^{top}, \text{type}(B)+1)$ with $P^{top} \subseteq P(B)$, $r(B^{top}) = e_r$, and $P(i) \not\subseteq P^{top}$ for any $i \in L_r$;
- L_ℓ is a set of at most M B-large tasks i with $\ell(B^{top}) - 1 \in P(i)$ such that $L_\ell \cap T_{e_r} = L_r \cap T_{\ell(B^{top})-1}$;
- $|(L_\ell \cup L_r \cup L_S) \cap T_e| \le M$ for every $e \in P(B)$;
- the total demand of $L_\ell \cup L_r \cup S_L$ and $\mathcal{B} \cup \{B, B^{top}\}$ w.r.t every $e \in P(B)$ is at most $u(e)$.

At the end of the process we set

$$DP_{main}(B, S_L, \mathcal{B}) \leftarrow \sum_i x_i(B) \cdot w(i) + DP_{sub}(B, S_L, \mathcal{B}, r(B)+1, \emptyset).$$

It is not hard to see that the computation of table DP_{sub} (hence of table DP_{main}) takes polynomial time.

4.2 Approximation ratio

We show that the profit of our DP solution is at least $\text{opt}_L + (1/3 - O(\varepsilon))\text{opt}_S$. Observe that by Property 3 of the boxed solution and using the rounding algorithm from [15], we can round the fractional solution over small tasks into an integral solution (while keeping feasibility) by losing only a factor of $1+\varepsilon$ in the profit. This implies Lemma 5.

Let APX be a solution with the same large tasks as OPT′, with the same boxes as OPT′, and with the small tasks APX_S that are obtained by executing the same procedure as in the DP algorithm for every box. Formally, for each box B we compute the bottom-up fractional solution $x(B)$ as described in Section 4.1 and we define $x := \sum_B x(B)$. To prove that the computed solution has large profit, we use a charging argument to show that the profit of APX_S is at least a $(1/3 - O(\varepsilon))$-fraction of opt_S. In our dynamic program, at every step, we take the choice that maximizes the profit of the respective subproblem. Thus, by an induction on the DP-cells one can show that the profit of the computed solution ALG is at least the profit of APX.

Lemma 11. *The DP computes a solution with a profit of at least* $\text{opt}_L + (1/3 - O(\varepsilon)) \cdot \text{opt}_S$.

Proof. From the previous discussion, we only need to show that $\text{apx}_S \geq (1/3 - O(\varepsilon))\text{opt}_S$, where apx_S denotes the profit of APX_S. We use a charging argument. For each box B let $y(B)$ denote the fractional solution according to OPT′ and let $y := \sum_B y(B)$. Recall that $y_i \leq 1/2$ for each task i. Let $y_i^{(1)} := \max\{x_i + y_i - 1, 0\}$ for each task i. The reader may imagine that for every task i we consider the interval $[0, 1)$, associate the subinterval $[0, x_i)$ with APX_S, associate the subinterval $[1 - y_i, 1)$ with OPT'_S, and then $y_i^{(1)}$ represents the length of the intersection of these two intervals. For each task i we have $y_i^{(1)} \leq x_i/2$ since $y_i \leq 1/2$ and thus $x_i + y_i - 1 \leq x_i - 1/2 \leq x_i/2$. Therefore, $\text{opt}'_{S,1} := \sum_i w(i)y_i^{(1)} \leq \text{apx}_S/2$. Let $y_i^{(2)} := y_i - y_i^{(1)}$ which intuitively represents the part of $[1 - y_i, 1)$ that has *not* been associated with APX_S. For each box B we define a value $y_i^{(2)}(B)$ such that $y_i^{(2)}(B) \leq y_i(B)$ and $\sum_B y_i^{(2)}(B) = y_i^{(2)}$. Note that in every step of the computation, $y_i^{(2)}$ units of task i are still available to be selected and hence when the algorithm computes the solution for a box B, one possible solution is given by $y_i^{(2)}(B)$ for each task i. Therefore, for each box B it holds that $\sum_i w(i)x_i(B) \geq \sum_i w(i)y_i^{(2)}(B)$ and therefore $\text{apx}_S = \sum_i w(i)x_i = \sum_B \sum_i w(i)x_i(B) \geq \sum_B \sum_i w(i)y_i^{(2)}(B) = \sum_i w(i)y_i^{(2)} =: \text{opt}'_{S,2}$. Therefore, we have $\text{opt}'_S = \text{opt}'_{S,1} + \text{opt}'_{S,2} \leq (3/2)\text{apx}_S$, hence $\text{apx}_S \geq (2/3) \cdot \text{opt}'_S \geq (1/3 - O(\varepsilon)) \cdot \text{opt}_S$. □

4.3 General case

In the general case with more than $O_\varepsilon(1)$ levels, the following problems arise. First, we can no longer "remember" all previously selected large tasks, i.e., we would need to allow the set S_L in the definition of the DP-cells to contain more than $O_\varepsilon(1)$ tasks. Using a technique from [21] we "forget" all but $O_\varepsilon(1)$ of these large tasks and ensure that the total demand of the forgotten tasks is bounded by a slack of $\varepsilon^4 \cdot (1+\varepsilon)^j$ that we leave on every edge of level j, like our boxed solution due to Lemma 10. Also, the size of the set of previously selected boxes $\mathcal{B}$ underneath a given box B can be $\Omega_\varepsilon(1)$. To this end, we remember only the last (i.e., largest) $O_\varepsilon(1)$ boxes in $\mathcal{B}$. As a result, it might be that we select a small task to a larger extent than one fractional unit (even though at the end we obtain its profit at most once) and thus waste space in some box B. We can, however, show that the total wasted capacity on each edge is at most an ε-fraction of the capacity of B and hence here we lose only a factor of $1+\varepsilon$ in the approximation ratio.

The main changes to Section 4 are the following.

- Instead of allowing the considered tasks and boxes to use the full capacity $u(e)$ of each edge e, we leave $\varepsilon^4 \cdot (1+\varepsilon)^{\text{type}(e)}$ units of unused capacity, like OPT′ (see Lemma 10). This unused capacity is referred to as the *slack*.
- We do not remember all previously selected large tasks but "forget" those that are sufficiently small compared to the capacity of the current box. We will show that the total demand of the forgotten tasks fits into the slack that we leave.
- Also, we cannot remember all ancestor boxes $\mathcal{B}$ of a given box B but only $O_\varepsilon(1)$ of them. Recall that we remembered the previously selected (fractions of) small tasks indirectly by remembering $\mathcal{B}$. Since we can remember only $O_\varepsilon(1)$ such boxes, we will "forget" some of the previously selected (fractions of) small tasks. As a result, we might select a small task (fractionally) more than once. However, we will show that we can repair this at the end at a negligible cost.

For technical reasons, we first apply a shifting argument to group the edge types into groups such that each group consists of $K = O_\varepsilon(1)$ consecutive types and there is a gap of $c_\varepsilon = \Omega_\varepsilon(1)$ types between two groups. In more details, let c_ε be the smallest integer constant such that $1/(1+\varepsilon)^{c_\varepsilon} \leq \varepsilon$. For an integer $a \geq 1$ define

$$GAP^{(a)} := \bigcup_{k \in \mathbb{N}_0} \bigcup_{\ell=-1}^{c_\varepsilon - 2} \{a \cdot c_\varepsilon + k \cdot c_\varepsilon/\varepsilon + \ell\}.$$

Lemma 12. *There is a constant* $a^* \in \{0, \ldots, 1/\varepsilon - 1\}$ *and a boxed solution* OPT″ *defined via a set of large tasks* S_L, *a set of small tasks* T_S, *a set of boxes* $\mathcal{B}$, *and a fractional solution* $\{x_i(B)\}_{i \in T_S}$ *for each box* $B \in \mathcal{B}$ *with weight* $\text{opt}'' \geq \text{opt}_L + (1/2 - O(\varepsilon))\text{opt}_S$, $x_i(B) = 0$ *for each task i in each box B such that* $\text{type}(B) \in GAP^{(a^*)}$ *and, additionally, for each box B and edge edge* $e \in P(B)$ *we have* $\sum_{i \in T_e \cap T_S} d(i)x_i(B) \leq (1-\varepsilon)u(B)$.

Proof. Observe that, for two different values $a, a' \in \{0, \ldots, 1/\varepsilon - 1\}$ we have that $\text{GAP}^{(a)} \cap \text{GAP}^{(a')} = \emptyset$. Therefore, there is a value $a^* \in \{0, \ldots, 1/\varepsilon - 1\}$ such that in OPT′ the profit of the fractions of the small tasks that are assigned to boxes with a type in $\text{GAP}^{(a^*)}$ is at most $\varepsilon \cdot \text{opt}'_S$ where opt'_S denotes the total profit of the small tasks in OPT$'$. For each box B in OPT$'$, denote by $x'(B)$ the fractional solution of OPT$'$ for box B. We define a new fractional solution $x''(B)$ by defining $x''_i(B) := (1-\varepsilon)x'_i(B)$ for each box B with type$(B) \notin$ GAP$^{(a^*)}$ and each task i and $x''_i(B) := 0$ for each box B with type$(B) \in$ GAP$^{(a^*)}$ and each task i. We lose one factor $1-\varepsilon$ due to the fractions of small tasks in boxes whose type is in GAP$^{(a^*)}$ and another factor $1-\varepsilon$ due to rounding down the fractional solution in the boxes whose type is not in GAP$^{(a^*)}$. We define OPT$''$ to be the solution obtained by the large tasks from OPT$'$, the boxes from OPT$'$, and the fractional solution $x''(B)$ for each box B. □

Let OPT$''$ be the solution obtained from the above lemma. For the constant a^* there are only $O_\varepsilon(1)$ possibilities and we guess the correct value in our algorithm. Note that this splits the edge and box types into groups $G_0 = \{-1,\dots,a^*c_\varepsilon-2\}$, $G_1 = \{(a^*+1)c_\varepsilon-1,\dots,a^*c_\varepsilon+c_\varepsilon/\varepsilon-2\}$, $G_2 = \{a^*c_\varepsilon+c_\varepsilon+c_\varepsilon/\varepsilon-1,\dots,a^*c_\varepsilon+2c_\varepsilon/\varepsilon-2\}$, ... such that only boxes with a type in some group G_i have small tasks assigned to them in OPT$''$. By $K = c_\varepsilon/\varepsilon$ we denote an upper bound on the size of a group.

For each integer j we define $T^{\geq j}$ to be all tasks i with $d(i) \geq \mu_2(1+\varepsilon)^j$. Let also K' be the smallest integer that satisfies $(1+\varepsilon)^{K'} \geq 2(1+\varepsilon)^2/\varepsilon^9$, and $K(j) := \min\{K, j+1\}$.

4.4 The Algorithm

As in Section 4.1, each DP cell is index by a tuple $(B, S_L, \mathcal{B})$ where now:

- $B = B_j$ is a box, $j = \text{type}(B)$;
- $\mathcal{B} = \{B_{j-K(j)},\dots,B_{j-1}\}$ is a collection of up to K boxes with type$(B_k) = k$ for each $k \in \{j-K(j),\dots,j-1\}$ and $P(B_{j-K(j)}) \supseteq \dots \supseteq P(B_{j-1}) \supseteq P(B_j)$;
- S_L consists of tasks $i \in T^{\geq \text{type}(B)-K'}$ with $P(i) \cap \{\ell(B), r(B)\} \neq \emptyset$, and $P(i) \nsubseteq P$;
- $|S_L \cap T_e| \leq M' := 2M(K'+1)$ for every $e \in P(B)$, where $M := 1/(\mu_2\varepsilon^4)$;
- The total demand of S_L and $\mathcal{B} \cup \{B\}$ on every edge $e \in P(B)$ is at most $u(e) - \varepsilon^4 \cdot (1+\varepsilon)^{\text{type}(B)}$ if type$(B) \geq 0$ and at most $u(e)$ if type$(B) = -1$.

So in contrast to Section 4 we do not use the full capacity of each edge e but leave a slack of $\varepsilon^4 \cdot (1+\varepsilon)^{\text{type}(B)}$. Note that our boxed solution OPT$'$ (and hence also OPT$''$) satisfies this property (see Lemma 10).

We slightly change the definition of bottom-up fractional solutions as follows: given a cell $(B, S_L, \mathcal{B})$ let $\mathcal{B}' \subseteq \mathcal{B}$ denote the boxes $B' \in \mathcal{B}$ such that type$(B') \in G_{\ell'}$ for the group $G_{\ell'}$ with type$(B) \in G_{\ell'}$. We compute the bottom-up fractional solution $x(B)$ based on the boxes $\mathcal{B}'$ only, rather than based on *all* boxes $\mathcal{B}$. We set $x(B) = 0$ for the boxes not belonging to any group.

Since now the number of levels can be larger than a constant, we cannot afford to remember all large tasks that we selected previously. Hence, we modify the DP-transition such that intuitively we "forget" some of the large tasks, similar as in [21]. More formally, when we reduce the problem of a DP-cell $(B, S_L, \mathcal{B})$ with $B = (P, j)$ to DP-cells of the form $(B^{top}, S_L^{top}, \mathcal{B}^{top})$ with $B^{top} = (P^{top}, j+1)$ then we do not add to S_L^{top} the tasks i of S_L that satisfy $d(i) < \mu_2(1+\varepsilon)^{\text{type}(B^{top})-K'}$. In order to argue that the computed solution is still feasible, we use the slack of $\varepsilon^4 \cdot (1+\varepsilon)^{\text{type}(B)}$ that we have on each edge and argue that the total capacity of the forgotten tasks is upper-bounded by this slack.

Formally, we change the transition of DP_{sub} as follows. The value $DP_{sub}(B, S_L, \mathcal{B}, e_r, L_r)$ is again the maximum over the following two values. The first value intuitively corresponds to the case that there is no top box using edge e_r:

$$\max_{L_\ell}\{w(L_r \setminus T_{e_r-1}) + DP_{sub}(B, S_L, \mathcal{B}, e_r-1, L_\ell)\}$$

where the maximum is taken over

- L_ℓ is a set of B-large tasks i with $e_r - 1 \in P(i)$ which is *consistent* with L_r, i.e. $L_\ell \cap T_{e_r} = L_r \cap T_{e_r-1}$;
- $|(L_\ell \cup L_r \cup S_L) \cap T_e| \leq M'$ for every $e \in P(B)$;
- for each $e \in P$ there are at most $2M$ tasks $i \in (L_\ell \cup L_r \cup S_L) \cap T_e$ such that $d(i) \geq \mu_2(1+\varepsilon)^{\text{type}(B)}$;
- the total demand of $L_\ell \cup L_r \cup S_L$ and $\mathcal{B} \cup \{B, B^{top}\}$ is at most $u(e) - \varepsilon^4 \cdot (1+\varepsilon)^{\text{type}(B)}$ on every $e \in P$ if type$(B) \geq 0$ and at most $u(e)$ if type$(B) = -1$.

The second value, which intuitively corresponds to the case that there is some top box ending at e_r, is

$$\max_{B^{top}, L_\ell}\Big\{w(L_r \setminus T_{\ell(B^{top})-1}) + DP_{main}\Big(B^{top},$$
$$(S_L \cup L_r \cup L_\ell) \cap (T_{\ell(B^{top})} \cup T_{r(B^{top})}) \cap T^{\geq \text{type}(B^{top})-K'},$$
$$\mathcal{B} \cup \{B\} \setminus \mathcal{B}_{out}\Big) + DP_{sub}(B, S_L, \mathcal{B}, \ell(B^{top})-1, L_\ell)\Big\}.$$

Here $\mathcal{B}_{out} = \{B_{out}\}$ where B_{out} is the box of smallest type in $\mathcal{B}$, unless type$(B_0) = -1$ in which case $\mathcal{B}_{out} = \emptyset$. Furthermore, the maximum is taken over:

- $B^{top} = (P^{top}, \text{type}(B)+1)$ with $P^{top} \subseteq P(B)$, $r(B^{top}) = e_r$, and $P(i) \nsubseteq P^{top}$ for any $i \in L_r$;
- L_ℓ is a set of B-large tasks i with $\ell(B^{top}) - 1 \in P(i)$, and $L_\ell \cap T_{e_r} = L_r \cap T_{\ell(B^{top})-1}$;
- $|(L_\ell \cup L_r \cup L_S) \cap T_e| \leq M'$ for every $e \in P(B)$;
- for each $e \in P$ there are at most $2M$ tasks $i \in (L_\ell \cup L_r \cup S_L) \cap T_e$ such that $d(i) \geq \mu_2(1+\varepsilon)^{\text{type}(B)}$;
- the total demand of $L_\ell \cup L_r \cup S_L$ and $\mathcal{B} \cup \{B, B^{top}\}$ is at most $u(e) - \varepsilon^4 \cdot (1+\varepsilon)^{\text{type}(B)}$ on every $e \in P$ if type$(B) \geq 0$ and at most $u(e)$ if type$(B)) = -1$.

At the end of the process we set

$$DP_{main}(B, S_L, \mathcal{B}) \leftarrow \sum_i x_i(B) \cdot w(i) + DP_{sub}(B, S_L, \mathcal{B}, r(B)+1, \emptyset).$$

The dynamic program runs in polynomial time since the number of DP-cells is bounded by a polynomial and also the transition clearly runs in polynomial time. The final solution consists of all large tasks that were selected at some point, and a fractional solution of small tasks defined as follows. For each box B we define its bottom-up fractional solution $x(B)$ based on the boxes $\mathcal{B}'$ that are placed underneath B in the same group as B, i.e., the boxes B' for which $P(B) \subseteq P(B')$ holds and such that there is a group G_ℓ with type$(B) \in G_\ell$ and type$(B') \in G_\ell$. We define the final solution by setting $\bar{x}_i := \min\{1, \sum_B x_i(B)\}$ for each task i corresponding to some box. We will show later that in case that $\bar{x}_i < \sum_B x_i(B)$ this yields only a negligible loss in the profit. It is convenient to define values $\bar{x}_i(B)$ as follows. Let $B_{-1},\dots,B_\ell$ be the boxes containing i, with $\text{type}(B_j) = j$. If $\sum_{j=-1}^{\ell} x_i(B_j) \leq 1$, we set $\bar{x}_i(B_j) = x_i(B_j)$ for all j. Otherwise, let ℓ' be the smallest index such that $\sum_{j=-1}^{\ell'} x_i(B_j) > 1$. We set $\bar{x}_i(B_j) = x_i(B_j)$ for $j < \ell'$, $\bar{x}_i(B_{\ell'}) = 1 - \sum_{j=-1}^{\ell'-1} x_i(B_j)$, and $\bar{x}_i(B_j) = 0$ for $j > \ell'$. Observe that in all cases we have $\bar{x}_i(B_j) \leq x_i(B_j)$ and $\sum_j \bar{x}_i(B_j) = \bar{x}_i$. We also define $x_i^{\text{waste}}(B) := x_i(B) - \bar{x}_i(B) \geq 0$.

Since we forgot some of the large tasks it is not immediately clear that the computed solution is a feasible boxed solution. To this end, we prove the following lemma in which we show in particular that on each edge e, the total demand of the forgotten large tasks fits into the slack of $\varepsilon^4 \cdot (1+\varepsilon)^{\text{type}(B)}$.

Lemma 13. *The dynamic program computes a feasible solution.*

Proof. Let e be an edge and let $B = (P, j)$ be the box of maximum type in the computed solution that satisfies $e \in P(B)$. Let S'_L denote the large tasks using e in the computed solution. The DP-transition guarantees us that the total demand of the boxes plus the total demand of all tasks in $S'_L \cap T^{\geq \text{type}(B)-K'}$ is at most $u(e) - \varepsilon^4 \cdot (1+\varepsilon)^{\text{type}(B)}$. We would like to bound the total demand of all tasks in $S'_L \setminus T^{\geq \text{type}(B)-K'}$ by $\varepsilon^4 \cdot (1+\varepsilon)^{\text{type}(B)}$.

Let $i \in S'_L \setminus T^{\geq \text{type}(B)-K'}$. Then i was selected before the box B was processed, so there must be a level j' in which it was chosen such that $d(i) \geq \mu_2 \cdot (1+\varepsilon)^{j'}$ but $d(i) < \mu_2 \cdot (1+\varepsilon)^{j'+1}$ with $j' < \text{type}(B) - K'$. So intuitively, i is large for level j' but not for level $j'+1$. The number of such large tasks using e (for this specific value j') is bounded by $2M$ since each such task must use the closest edge on the left or the closest edge on the right of e that is of level j'. Therefore, the total demand of the tasks in $S'_L \setminus T^{\geq \text{type}(B)-K'} \cap T_e$ is bounded by

$$d(S'_L \setminus T^{\geq \text{type}(B)-K'} \cap T_e) < \sum_{j'=0}^{\text{type}(B)-K'-1} 2M\mu_2 \cdot (1+\varepsilon)^{j'+1}$$
$$\leq 2M\mu_2 \frac{1}{\varepsilon}(1+\varepsilon)^{\text{type}(B)-K'+1} \leq \varepsilon^4 \cdot (1+\varepsilon)^{\text{type}(B)}$$

due to our choice of K'. Hence, the computed solution is feasible. □

4.5 Approximation ratio

We argue that the computed solution has large profit. Like in Section 4 we use a charging argument. For the sake of analysis, let APX be a solution with the same large tasks as OPT″, with the same boxes as OPT″, and by computing a bottom-up fractional solution for all boxes as described above. First, we argue that in the recursion, the DP might indeed select the large tasks and the boxes such that at the end all large tasks and all boxes from OPT″ are chosen. (Note however, since in each step the DP takes the choice that maximizes the profit, the DP might select other tasks for the final solution.)

Lemma 14. *In* OPT″*, for each integer j each edge e is used by at most M' large tasks in $T^{\geq j-K'}$ which also use an edge of type at most j.*

Proof. Let i be a large task using e in OPT″ with $i \in T^{\geq j-K'}$ which also uses an edge of type at most j. If $d(i) < \mu_2 \cdot (1+\varepsilon)^{j+1}$ then let j' be the largest integer such that $d(i) \geq \mu_2 \cdot (1+\varepsilon)^{j'}$ but $d(i) < \mu_2 \cdot (1+\varepsilon)^{j'+1}$, otherwise let $j' := j$. We have that $j \geq j' \geq j - K'$. Then, $P(i)$ contains the closest edge e' on the left of e or on the right of e with $f(e') = (1+\varepsilon)^{j'}$ (and hence of type j'). In OPT″ each edge e' of type j' is used by at most $M = 1/(\mu_2\varepsilon^4)$ tasks i' with $d(i') \geq \mu_2 \cdot (1+\varepsilon)^{j'}$. Hence, the total number of large tasks using e in OPT″ with $i \in T^{\geq j-K'}$ and which also use an edge of type j is bounded by $2(K'+1)M = M'$. □

The tasks S_L in the tuple $(B, S_L, \mathcal{B})$ corresponding to a DP-cell of DP_{main} are large tasks in $T^{\geq \text{type}(B)-K'}$ that use an edge of type smaller than type(B) outside the path $P(B)$ of B. Due to Lemma 14 there can be at most M' such tasks on each edge $e \in P(B)$. Therefore, it is justified to require that $|S_L \cap T_e| \leq M'$ for each edge e of $P(B)$ in the definition of the DP-cells of DP_{main}. Moreover, this also justifies that we require that $|(L_\ell \cup L_r \cup S_L) \cap T_e| \leq M'$ for every $e \in P(B)$ in DP_{sub}. Finally, note that in OPT″ each edge of type j is used by at most M large tasks $i \in S_L$ such that $d(i) \geq \mu_2(1+\varepsilon)^j$ which justifies that we require this also in the definition of DP_{sub}. Therefore, in the recursion the DP could choose all large tasks and all boxes from OPT″ in the respective maximization steps.

Denote by apx the weight of APX and by apx_S the weight of the small tasks in APX. For each box B denote by $x(B)$ the resulting fractional solution.

Profit from small tasks. Recall that when we process a DP-cell $(B, S_L, \mathcal{B})$ then we compute the bottom-up solution $x(B)$ and to this end considered the boxes $\mathcal{B}' \subseteq \mathcal{B}$ that belong to the same group as B. Note that when we process the current DP-cell, we assume that for each $B' \in \mathcal{B} \setminus \mathcal{B}'$ one has $x(B') = 0$. However, consider the boxes $\tilde{\mathcal{B}}$ underneath $\mathcal{B}$. When we compute $x(B)$, we neglect the value of $x(B')$ for $B' \in \tilde{\mathcal{B}}$. As a consequence, it might happen that $\sum_B x_i(B) > 1$ for some task i which is bigger than the final fractional value $\bar{x}_i = \min\{1, \sum_B x_i(B)\}$ for i. We argue now that we still obtain a profit of at least $\text{opt}_L + (\frac{1}{3} - O(\varepsilon))\text{opt}_S$ overall.

For each box B let $y(B)$ denote the fractional solution according to OPT″ and let $y := \sum_B y(B)$. Recall that $y_i \leq 1/2$ for each task i. Denote by opt''_S the weight of the small tasks in OPT″ and by apx_S the weight of the small tasks in APX. We want to prove that $\text{opt}''_S \leq \frac{3}{2}\text{apx}_S$. To this end, we split the solution y into two parts as in Section 4.2. For the first part, let $y_i^{(1)} := \max\{\bar{x}_i + y_i - 1, 0\}$ for each task i. We have that $y_i \leq 1/2$ and thus $\bar{x}_i + y_i - 1 \leq \bar{x}_i - 1/2 \leq \bar{x}_i/2$ which implies that $y_i^{(1)} \leq \bar{x}_i/2$. Therefore, $\text{opt}''_{S,1} := \sum_i w(i) y_i^{(1)} \leq \text{apx}_S/2$. For the second part, let $y_i^{(2)} := y_i - y_i^{(1)}$. For each box B we can find a value $y_i^{(2)}(B)$ (which intuitively represents the part of $y_i^{(2)}$ in the box B) such that $y_i^{(2)}(B) \leq y_i(B)$ and overall $\sum_B y_i^{(2)}(B) = y_i^{(2)}$.

Consider a DP-cell $(B, S_L, \mathcal{B})$. Assume that type$(B) \notin GAP$ since otherwise $x(B) = \bar{x}(B) = y(B) = \mathbf{0}$. Let $\mathcal{B}' \subseteq \mathcal{B}$ denote the boxes $B' \in \mathcal{B}$ such that type$(B') \in G_{\ell'}$ for the group $G_{\ell'}$ with type$(B) \in G_{\ell'}$.

For each box B recall that $x(B)$ is the computed bottom-up fractional solution. Note that in every step of the computation, $y_i^{(2)}$ units of task i are still available to be selected and hence when we compute the (bottom-up fractional) solution for a box B, one possible solution is given by $y_i^{(2)}(B)$ for each task i. Therefore, for each box B it holds that $\sum_i w(i)x_i(B) \geq \sum_i w(i)y_i^{(2)}(B)$. Unfortunately, we cannot infer directly that also $\sum_i w(i)\bar{x}_i(B) \geq \sum_i w(i)y_i^{(2)}(B)$. This can happen if $\sum_{B' \in \mathcal{B} \cup \tilde{\mathcal{B}} \cup \{B\}} x_i(B') > 1$ for some task i with $x_i(B) > 0$. In particular, when we compute $x(B)$, then some space of B will be wasted by (fractions of) tasks that we select for the box B but which were already selected in the boxes $\tilde{\mathcal{B}}$. However, since the heights of the boxes are geometrically increasing, we will show that in this way we waste at most an ε-fraction of the capacity of B on each edge. Since also the solution y does not use the full capacity in each box B (see Lemma 12) we can argue that still $\sum_i w(i)\bar{x}_i(B) \geq \sum_i w(i)y_i^{(2)}(B)$ holds.

Let us assume that $\text{type}(B) \geq K+1$, since otherwise $x(B) = \bar{x}(B)$ (in this case $\tilde{\mathcal{B}} = \emptyset$). We have that $x_i^{\text{waste}}(B) \leq \sum_{B' \in \tilde{\mathcal{B}}} x_i(B')$. We also have that $x(B')$ defines a feasible fractional solution w.r.t. each box B'. Hence, for each edge $e \in P(B)$, we obtain

$$\begin{aligned}\sum_{i \in T_e} d(i)x_i^{\text{waste}}(B) &\leq \sum_{i \in T_e, B' \in \tilde{\mathcal{B}}} d(i)x_i(B') \\ &\leq \sum_{B':\text{type}(B') \leq \text{type}(B) - c_\varepsilon} u(B') = (1+\varepsilon)^{\text{type}(B) - c_\varepsilon} \leq \varepsilon u(B)\,.\end{aligned}$$

In the second last inequality above we used that, if $\text{type}(B) \in G_\ell$, then, for any $B' \in \tilde{\mathcal{B}}$, $\text{type}(B') \in G_{\ell'}$ for some $\ell' < \ell$ or $x(B') = \mathbf{0}$. In the last inequality we used the definition of c_ε.

One feasible solution to box B is $y_i^{(2)}(B) + x_i^{\text{waste}}(B)$ for each task i. Since $x(B)$ is the optimal solution to box B (considering the already selected fractions of tasks in the boxes $\mathcal{B}$), we have $\sum_i x_i(B) \geq \sum_i y_i^{(2)}(B) + x_i^{\text{waste}}(B)$ and hence we have

$$\sum_i w(i)\bar{x}_i(B) = \sum_i w(i)(x_i(B) - x_i^{\text{waste}}(B)) \geq \sum_i w(i)y_i^{(2)}(B)$$

for each box B. We conclude that

$$\begin{aligned}\text{apx}_S := \sum_i w(i)\bar{x}_i &= \sum_B \sum_i w(i)\bar{x}_i(B) \\ &\geq \sum_B \sum_i w(i)y_i^{(2)}(B) = \sum_i w(i)y_i^{(2)} =: \text{opt}''_{S,2}.\end{aligned}$$

Putting everything together one achieves:

$$\text{opt}''_S = \text{opt}''_{S,1} + \text{opt}''_{S,2} \leq \frac{3}{2}\text{apx}_S.$$

Finally, we note that in each step the DP takes the decision that optimizes the value of the computed solution. Therefore, by an induction over the DP-cells one can show that the DP computes a solution whose profit is at least $\text{apx} \geq \text{opt}_L + \frac{2}{3}\text{opt}''_S \geq \text{opt}_L + (\frac{1}{3} - O(\varepsilon))\text{opt}_S$. Finally, we round the fractional solution over small tasks into an integral solution (while keeping feasibility) while losing at most a factor of $1+\varepsilon$ in the profit of the small tasks, using the algorithm in [15]. This completes the proof of Theorem 1.

REFERENCES

[1] Anna Adamaszek, Parinya Chalermsook, Alina Ene, and Andreas Wiese. 2016. Submodular Unsplittable Flow on Trees. In *IPCO (Lecture Notes in Computer Science)*, Vol. 9682. 337–349. https://doi.org/10.1007/978-3-319-33461-5_28

[2] Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Andreas Wiese. 2013. Constant Integrality Gap LP Formulations of Unsplittable Flow on a Path. In *IPCO*. 25–36. https://doi.org/10.1007/978-3-642-36694-9_3

[3] Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Andreas Wiese. 2014. A Mazing 2+ε Approximation for Unsplittable Flow on a Path. In *SODA*. 26–41.

[4] E. M. Arkin and E. B. Silverberg. 1987. Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics* 18 (1987), 1–8. Issue 1.

[5] N. Bansal, A. Chakrabarti, A. Epstein, and B. Schieber. 2006. A quasi-PTAS for unsplittable flow on line graphs. In *STOC*. ACM, 721–729.

[6] N. Bansal, Z. Friggstad, R. Khandekar, and R. Salavatipour. 2009. A logarithmic approximation for unsplittable flow on line graphs. In *SODA*. 702–709.

[7] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. 2000. A unified approach to approximating resource allocation and scheduling. In *STOC*. 735–744.

[8] R. Bar-Yehuda, M. Beder, Y. Cohen, and D. Rawitz. 2006. Resource Allocation in Bounded Degree Trees. In *ESA*. 64–75.

[9] Jatin Batra, Naveen Garg, Amit Kumar, Tobias Mömke, and Andreas Wiese. 2015. New Approximation Schemes for Unsplittable Flow on a Path. In *SODA*. 47–58. https://doi.org/10.1137/1.9781611973730.5 arXiv:http://epubs.siam.org/doi/pdf/10.1137/1.9781611973730.5

[10] Paul Bonsma, Jens Schulz, and Andreas Wiese. 2014. A Constant-Factor Approximation Algorithm for Unsplittable Flow on Paths. *SIAM J. Comput.* 43 (2014), 767–799.

[11] Gruia Călinescu, Amit Chakrabarti, Howard J. Karloff, and Yuval Rabani. 2011. An improved approximation algorithm for resource allocation. *ACM Transactions on Algorithms* 7, Article 48 (2011), 48:1–48:7 pages. https://doi.org/10.1145/2000807.2000816

[12] Venkatesan T. Chakaravarthy, Anamitra R. Choudhury, Shalmoli Gupta, Sambuddha Roy, and Yogish Sabharwal. 2014. Improved Algorithms for Resource Allocation under Varying Capacity. In *ESA*. 222–234.

[13] A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar. 2007. Approximation Algorithms for the Unsplittable Flow Problem. *Algorithmica* 47 (2007), 53–78.

[14] C. Chekuri, A. Ene, and N. Korula. 2009. Unsplittable Flow in Paths and Trees and Column-Restricted Packing Integer Programs. In *APPROX-RANDOM*. 42–55.

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

[16] B. Chen, R. Hassin, and M. Tzur. 2002. Allocation of bandwidth and storage. *IIE Transactions* 34 (2002), 501–507.

[17] M. Chrobak, G. Woeginger, K. Makino, and H. Xu. 2010. Caching Is Hard, Even in the Fault Model. In *ESA*. 195–206.

[18] A. Darmann, U. Pferschy, and J. Schauer. 2010. Resource allocation with time intervals. *Theoretical Computer Science* 411 (2010), 4217–4234. Issue 49.

[19] N. Garg, V. V. Vazirani, and M. Yannakakis. 1997. Primal-Dual Approximation Algorithms for Integral Flow and Multicut in Trees. *Algorithmica* 18, 1 (1997), 3–20.

[20] Fabrizio Grandoni, Salvatore Ingala, and Sumedha Uniyal. 2015. Improved Approximation Algorithms for Unsplittable Flow on a Path with Time Windows. In *WAOA*. 13–24.

[21] Fabrizio Grandoni, Tobias Mömke, Andreas Wiese, and Hang Zhou. 2017. To Augment or Not to Augment: Solving Unsplittable Flow on a Path by Creating Slack. In *SODA*. 2411–2422.

[22] S. Leonardi, A. Marchetti-Spaccamela, and A. Vitaletti. 2000. Approximation Algorithms for Bandwidth and Storage Allocation Problems under Real Time Constraints. In *FSTTCS*. 409–420.

[23] C. A. Phillips, R. N. Uma, and J. Wein. 2000. Off-line admission control for general scheduling problems. In *SODA*. 879–888.

[24] A. Schrijver. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, Berlin.

A PROOF OF LEMMA 2

We start from an optimal solution OPT with profit opt. First, we use a result by the authors of this paper [21] to create some *slack* capacity on each edge by deleting some tasks from OPT, and afterwards we use a shifting argument to define the large and the small tasks. We assume that the constant $\varepsilon > 0$ is sufficiently small and that $1/\varepsilon$ is an integer.

Lemma 15 (Lemma 2.1 from [21]). *Let $\tilde{\varepsilon} > 0$ be a constant. For a UFP instance with an optimal solution* OPT, *there exists a feasible solution* $\widehat{\text{OPT}}$ *with profit* $\widehat{\text{opt}} \geq (1 - O(\tilde{\varepsilon})) \cdot \text{opt}$ *such that for each edge e there is a value $\tilde{\delta}_e \geq 0$ satisfying the following properties:*

(1) *either* $\tilde{\delta}_e = (1/\tilde{\varepsilon}^2)^j$ *for some integer* $j \geq 0$*, or* $\tilde{\delta}_e = 0$*;*
(2) $d(T_e \cap \widehat{\mathrm{OPT}}) \leq u(e) - \tilde{\delta}_e$*;*
(3) *there are at most* $1/\tilde{\varepsilon}^5$ *tasks* $i \in T_e \cap \widehat{\mathrm{OPT}}$ *such that* $d(i) \geq \tilde{\varepsilon}^2 \cdot \tilde{\delta}_e$*;*
(4) *the total demand of all tasks* $i \in T_e \cap \widehat{\mathrm{OPT}}$ *such that* $d(i) < \tilde{\varepsilon}^2 \cdot \tilde{\delta}_e$ *is at most* $5\tilde{\delta}_e/\tilde{\varepsilon}^3$*.*

The following lemma is a corollary of Lemma 15.

Lemma 16. *Let* $\varepsilon > 0$ *be a constant. For a UFP instance with an optimal solution* OPT*, there exists a feasible solution* $\widehat{\mathrm{OPT}}$ *with profit* $\widehat{\mathrm{opt}} \geq (1-O(\varepsilon)) \cdot \mathrm{opt}$ *such that for each edge e there is a value* $\delta_e \geq 0$ *which satisfies the following properties:*

(1) $d(T_e \cap \widehat{\mathrm{OPT}}) \leq u(e) - 3\delta_e$*;*
(2) *there are at most* $16/\varepsilon^5$ *tasks* $i \in T_e \cap \widehat{\mathrm{OPT}}$ *such that* $d(i) \geq \varepsilon^2 \cdot \delta_e$*;*
(3) *the total demand of all tasks* $i \in T_e \cap \widehat{\mathrm{OPT}}$ *such that* $d(i) < \varepsilon^2 \cdot \delta_e$ *is at most* $80\delta_e/\varepsilon^3$*.*

Proof of Lemma 16 using Lemma 15. We apply Lemma 15 with $\tilde{\varepsilon} := \varepsilon/\sqrt{3}$ and obtain $\widehat{\mathrm{OPT}}$ and $\{\tilde{\delta}_e\}_e$. For every edge e, let $\delta_e = \tilde{\delta}_e/3$. Then Property 1 of the lemma statement holds, and $\tilde{\varepsilon}^2 \cdot \tilde{\delta}_e = \varepsilon^2 \cdot \delta_e$. There are at most $1/\tilde{\varepsilon}^5 \leq 16/\varepsilon^5$ tasks $i \in T_e \cap \widehat{\mathrm{OPT}}$ such that $d(i) \geq \varepsilon^2 \cdot \delta_e$, and in addition, the total demand of all tasks $i \in T_e \cap \widehat{\mathrm{OPT}}$ such that $d(i) < \varepsilon^2 \cdot \delta_e$ is at most $5\tilde{\delta}_e/\tilde{\varepsilon}^3 \leq 80\delta_e/\varepsilon^3$. □

We apply Lemma 16 and obtain the set $\widehat{\mathrm{OPT}}$ and $\{\delta_e\}_e$. Next, we use a shifting argument to define the large and the small tasks in Lemma 2. Define $\alpha = \varepsilon^8/(1+\varepsilon)^{1/\varepsilon^3}$. For every $k \in \{1, \ldots, 1/\varepsilon\}$ we define a set of tasks $\hat{Z}^{(k)}$ as:

$$\hat{Z}^{(k)} := \{i \in \widehat{\mathrm{OPT}} \mid d(i) < (\alpha^k/\varepsilon^4) \cdot \delta_e \text{ for every } e \in P(i)\}.$$

We define a profile $\hat{f}^{(k)}$ by $\hat{f}^{(k)}(e) := d(\hat{Z}^{(k)} \cap T_e) + 2\delta_e$ for each edge e. Thus the demand of the tasks in $\hat{Z}^{(k)}$ using the edge e is at most $\hat{f}^{(k)}(e)$. We round each value $\hat{f}^{(k)}(e)$ to a power of $(1+\varepsilon)$ to obtain a profile $f^{(k)}$ defined by $f^{(k)}(e) := (1+\varepsilon)^{\lfloor \log_{1+\varepsilon} \hat{f}^{(k)}(e) \rfloor}$. As a result, $\hat{f}^{(k)}(e) \geq f^{(k)}(e) > \hat{f}^{(k)}(e)/(1+\varepsilon)$ and $f^{(k)}(e) \geq \delta_e$ for every e.

Fact 17. $f^{(k)}(e) \leq \delta_e/\varepsilon^4$ *for any* $k \in \{1, \ldots, 1/\varepsilon\}$ *and for any edge* $e \in E$*.*

Proof. We have

$$f^{(k)}(e) \leq \hat{f}^{(k)}(e) \leq d(\hat{Z}^{(k)} \cap T_e) + 2\delta_e \leq 80\delta_e/\varepsilon^3 + 2\delta_e \leq \delta_e/\varepsilon^4,$$

where the third inequality follows by Lemma 16 and the fact that any task $i \in \hat{Z}^{(k)} \cap T_e$ is such that $d(i) < (\alpha^k/\varepsilon^4) \cdot \delta_e < \varepsilon^2 \cdot \delta_e$, and the last inequality holds since we assume that ε is sufficiently small. □

Intuitively we want to shrink each task in $\hat{Z}^{(k)}$ by an ε fraction so that all tasks in $\hat{Z}^{(k)}$ fit into the profile $f^{(k)}$. However, we cannot simply reduce the demand of each task. Instead, using the following lemma, we obtain a subset $Z^{(k)} \subseteq \hat{Z}^{(k)}$ such that for each edge e the total demand of the tasks in $Z^{(k)} \cap T_e$ is at most $(1-\varepsilon)$ times the total demand of the tasks in $\hat{Z}^{(k)} \cap T_e$.

Lemma 18 (Follows from [15]). *Consider a UFP instance such that for each input task i it holds that* $d(i) \leq \varepsilon^2 \cdot u(e)$ *for every edge* $e \in P(i)$*. Let* $\hat{S}$ *be a set of tasks that is a feasible solution to this instance. Let* $\beta \in (0,1)$ *be any constant. Then there is a subset* $S \subseteq \hat{S}$ *such that on each edge e the total demand of the tasks in S using e is at most* $\beta \cdot u(e)$ *and that* $w(S) \geq (\beta - O(\varepsilon))w(\hat{S})$*.*

We apply Lemma 18 with $\hat{S} := \hat{Z}^{(k)}$, $\beta := 1-\varepsilon$, and $u(e) := \hat{f}^{(k)}(e)$ for every edge e. The condition of the lemma is satisfied, since for every task $i \in \hat{Z}^{(k)}$ and for every edge $e \in P(i)$, we have

$$d(i) < (\alpha^k/\varepsilon^4) \cdot \delta_e < \varepsilon^2 \cdot \hat{f}^{(k)}(e).$$

Let $Z^{(k)} \subseteq \hat{Z}^{(k)}$ be the set S obtained from Lemma 18. Then $w(Z^{(k)}) \geq (1-O(\varepsilon))w(\hat{Z}^{(k)})$ and, for each edge e, we have

$$d(Z^{(k)} \cap T_e) \leq \hat{f}^{(k)}(e) \cdot (1-\varepsilon) < \hat{f}^{(k)}(e)/(1+\varepsilon) < f^{(k)}(e). \quad (1)$$

We define $\widetilde{\mathrm{OPT}}^{(k)} = (\widehat{\mathrm{OPT}} \setminus \hat{Z}^{(k)}) \cup Z^{(k)}$. We note that $\widehat{\mathrm{OPT}} \setminus \hat{Z}^{(k)}$ and $Z^{(k)}$ are disjoint subsets of $\widehat{\mathrm{OPT}}$ and hence $\widetilde{\mathrm{OPT}}^{(k)} \subseteq \widehat{\mathrm{OPT}}$. We then define the small tasks $\mathrm{OPT}_S^{(k)}$ and the large tasks $\mathrm{OPT}_L^{(k)}$ as follows:

$$\mathrm{OPT}_S^{(k)} := \{i \in \widetilde{\mathrm{OPT}}^{(k)} : d(i) < \mu_1^{(k)} \cdot f^{(k)}(e) \text{ for every edge } e \in P(i)\}$$

$$\mathrm{OPT}_L^{(k)} := \{i \in \widetilde{\mathrm{OPT}}^{(k)} : d(i) \geq \mu_2^{(k)} \cdot f^{(k)}(e) \text{ for some edge } e \in P(i)\}$$

where we define $\mu_1^{(k)} = \alpha^k$ and $\mu_2^{(k)} = \alpha^{k-1} \cdot \varepsilon^4$.

Fact 19. $\mathrm{OPT}_S^{(k)} \subseteq Z^{(k)}$ *and* $\mathrm{OPT}_L^{(k)} \subseteq \widehat{\mathrm{OPT}} \setminus \hat{Z}^{(k)}$*.*

Proof. Consider any task $i \in \mathrm{OPT}_S^{(k)}$. For every edge $e \in P(i)$, we have $d(i) < \mu_1^{(k)} \cdot f^{(k)}(e) \leq \alpha^k \cdot \delta_e/\varepsilon^4$ by Fact 17. Thus $i \in \hat{Z}^{(k)}$ and hence $i \in Z^{(k)}$.

Next, consider any task $i \in \mathrm{OPT}_L^{(k)}$. There exists some edge $e \in P(i)$ such that $d(i) \geq \mu_2^{(k)} \cdot f^{(k)}(e) = (\alpha^{k-1} \cdot \varepsilon^4) \cdot f^{(k)}(e) > (\alpha^k/\varepsilon^4) \cdot \delta_e$. Thus $i \notin \hat{Z}^{(k)}$ and hence $i \in \widehat{\mathrm{OPT}} \setminus \hat{Z}^{(k)}$. □

From Fact 19, $\mathrm{OPT}_L^{(k)} \cap \mathrm{OPT}_S^{(k)} = \emptyset$. However, there might be tasks in $\widetilde{\mathrm{OPT}}^{(k)}$ that are neither in $\mathrm{OPT}_S^{(k)}$ nor in $\mathrm{OPT}_L^{(k)}$. We denote them by $\mathrm{OPT}_M^{(k)} := \widetilde{\mathrm{OPT}}^{(k)} \setminus (\mathrm{OPT}_S^{(k)} \cup \mathrm{OPT}_L^{(k)})$. To make our shifting argumentation work, we first prove that each task can appear in at most one such set $\mathrm{OPT}_M^{(k)}$.

Lemma 20. *For each pair* $k, k' \in \mathbb{N}^+$ *with* $k \neq k'$ *we have* $\mathrm{OPT}_M^{(k)} \cap \mathrm{OPT}_M^{(k')} = \emptyset$*.*

Proof. Assume that $k' > k$. Let $i \in \mathrm{OPT}_M^{(k')}$. Then for every edge $e \in P(i)$ we have $d(i) < \mu_2^{(k')} \cdot f^{(k')}(e)$ since otherwise $i \in \mathrm{OPT}_L^{(k')}$. Using Fact 17, we have

$$d(i) < \mu_2^{(k')} \cdot f^{(k')}(e) \leq \mu_2^{(k')} \cdot (1/\varepsilon^4) \cdot \delta_e \leq \mu_1^{(k)} \cdot \delta_e \leq \mu_1^{(k)} \cdot f^{(k)}(e).$$

Hence $i \notin \mathrm{OPT}_M^{(k)}$. □

Due to Lemma 20 we can find a value $k^* \in \{1, \ldots, 1/\varepsilon\}$ such that $\mathrm{opt}_M^{(k^*)} \leq \varepsilon \cdot \mathrm{opt}$, where $\mathrm{opt}_M^{(k^*)} = w(\mathrm{OPT}_M^{(k^*)})$. For this value k^*, let $\mu_1 = \mu_1^{(k^*)}$ and $\mu_2 = \mu_2^{(k^*)}$, which satisfy that $\mu_1, \mu_2 \in (0, \varepsilon^4)$ and

$\mu_1 < \mu_2/(1+\varepsilon)^{1/\varepsilon^3}$. Using μ_1 and μ_2, we define the sets T_L and T_S using the formula in the lemma statements. Let $\mathrm{OPT}_L = \mathrm{OPT}_L^{(k^*)}$, $\mathrm{OPT}_S = \mathrm{OPT}_S^{(k^*)}$, $\hat{f} = \hat{f}^{(k^*)}$, $f = f^{(k^*)}$, and $\hat{Z} = \hat{Z}^{(k^*)}$. Properties 1, 3, and 4 of Lemma 2 follow directly from the construction, and Property 6 of Lemma 2 follows from Eq. (1) and Fact 19. It only remains to show Properties 2, 5, and 7 of Lemma 2.

To show Property 2 of Lemma 2, we have

$$\begin{aligned}
&w(\mathrm{OPT}_L \cup \mathrm{OPT}_S)\\
&\quad\geq w\left(\widetilde{\mathrm{OPT}}^{(k^*)}\right) - \varepsilon\cdot\mathrm{opt} && (\text{since } \mathrm{opt}_M^{(k^*)} \leq \varepsilon\cdot\mathrm{opt})\\
&\quad\geq (1-O(\varepsilon))\cdot w(\widehat{\mathrm{OPT}}) - \varepsilon\cdot\mathrm{opt} && (\text{construction of } \widetilde{\mathrm{OPT}}^{(k^*)})\\
&\quad\geq (1-O(\varepsilon))\cdot\mathrm{opt}\,. && (\text{by Lemma 16})
\end{aligned}$$

To show Property 5 of Lemma 2, first, we note that $\mathrm{OPT}_L \subseteq \widehat{\mathrm{OPT}}$. Thus by Lemma 16, there can be at most $16/\varepsilon^5$ tasks $i \in T_e \cap \mathrm{OPT}_L$ such that $d(i) \geq \varepsilon^2\cdot\delta_e$. Also by Lemma 16, the total demand of the tasks $i \in T_e \cap \mathrm{OPT}_L$ with $d(i) < \varepsilon^2\cdot\delta_e$ is at most $80\delta_e/\varepsilon^3$. Hence, among the latter tasks there can be at most $\frac{80\delta_e/\varepsilon^3}{\mu_2\cdot f(e)} \leq \frac{80\delta_e/\varepsilon^3}{\mu_2\cdot\delta_e} = 80/(\mu_2\cdot\varepsilon^3)$ tasks i with $d(i) \geq \mu_2\cdot f(e)$ in $T_e \cap \mathrm{OPT}_L$. Therefore, at each edge e, the total number of tasks $i \in T_e \cap \mathrm{OPT}_L$ such that $d(i) \geq \mu_2\cdot f(e)$ is at most $(16/\varepsilon^5) + 80/(\mu_2\cdot\varepsilon^3)$, which is at most $1/(\mu_2\cdot\varepsilon^4)$ for sufficiently small ε.

Finally, we show Property 7 of Lemma 2. For each edge e, we have

$$\begin{aligned}
&d(\mathrm{OPT}_L \cap T_e)\\
&\quad\leq d((\widehat{\mathrm{OPT}} \setminus \hat{Z}) \cap T_e) && (\text{by Fact 19})\\
&\quad= d(\widehat{\mathrm{OPT}} \cap T_e) - d(\hat{Z} \cap T_e) && (\text{since } \hat{Z} \subseteq \widehat{\mathrm{OPT}})\\
&\quad\leq u(e) - 3\delta_e - d(\hat{Z} \cap T_e) && (\text{by Property 1 of Lemma 16})\\
&\quad= u(e) - \hat{f}(e) - \delta_e && (\text{by the definition of } \hat{f}(e))\\
&\quad\leq u(e) - f(e) - \delta_e && (\text{since } f(e) \leq \hat{f}(e))\\
&\quad\leq u(e) - f(e) - \varepsilon^4\cdot f(e) && (\text{by Fact 17}).
\end{aligned}$$

We completed the proof of Lemma 2.