Unsplittable Flow on a Path: The Game!

Fabrizio Grandoni* Tobias Mömke† Andreas Wiese‡

Abstract

The unsplittable flow on a path (UFP) problem is a well-studied optimization problem, and it has applications in various settings like bandwidth allocation, caching, and scheduling. We are given a path with capacities on its edges and a set of n tasks, each of them defined via a demand, a subpath, and a profit. The goal is to select the most profitable set of tasks that together respect the edge capacities, i.e., for each edge e the total demand of the selected tasks whose subpath contains e is at most the capacity of e. The best known polynomial time approximation algorithm for UFP is a $(5/3+\varepsilon)$-approximation [Grandoni et al., STOC 2018]. It is an important open question whether the problem admits a PTAS. Informally, a task is *large* if its demand is at least an ε-fraction of the capacity of some edge on its path, and *small* otherwise. If all tasks are large, a PTAS can be obtained via dynamic programming: intuitively each edge e is used by only $O(1)$ relevant tasks in the optimal solution OPT. The same approach fails for small tasks since then this number can be up to $\Omega(n)$ which would yield an exponential number of states.

In this paper we introduce a novel *randomized sketching* technique to address this issue. We model the computation of a solution as a solitary game where tasks are presented one by one to a player, who has to decide for each task i whether to select i (hence getting its profit) or not. When a small task i is selected, with some probability its demand is rounded up to some large value (and then i behaves like a large task), and otherwise down to zero (and then i can be "forgotten" afterwards), so that in expectation the demand of i does not change. The optimal strategy to play this game can be computed using similar ideas as used in the DP for large tasks. Furthermore, the expected profit of this strategy is at least as large as the profit of OPT. One complication is that the player's solution might be infeasible, e.g., when too many tasks are rounded down. Still, via probabilistic arguments, we can use it to construct a feasible UFP solution which is $1+\frac{1}{1+e}+\varepsilon<1.269$ approximate in expectation. It is potentially possible that a more sophisticated probabilistic analysis gives a PTAS for the problem.

We believe that randomized sketching might turn out to be useful to address also other problems in which "large" and "small" objects interact, for example in packing, scheduling, or resource allocation settings, in particular when dynamic programming works if there are only large objects.

*IDSIA, USI-SUPSI, Switzerland, fabrizio@idsia.ch. Partially supported by the SNSF Excellence Grant 200020B_182865/1.

†Department of Computer Science, University of Augsburg, Germany, moemke@informatik.uni-augsburg.de. This work was partially supported by the DFG Grant 439522729 (Heisenberg-Grant), the ERC Advanced Investigators Grant 695614 (POWVER), and by DFG Grant 389792660 as part of TRR 248 (CPEC).

‡Department of Industrial Engineering, Universidad de Chile, Chile awiese@dii.uchile.cl. Partially supported by the ANID Fondecyt Regular grant 1200173.

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

1 Introduction

Unsplittable flow on a path (UFP) is a well-studied problem in combinatorial optimization. We are given an undirected path $G = (V, E)$ with a capacity $u(e) \in \mathbb{N}_0$ for each edge $e \in E$ and a set of n tasks T. Each task $i \in T$ is specified by a subpath $P(i) \subseteq G$, a demand $d(i) \in \mathbb{N}_0$, and a weight (or profit) $w(i) \in \mathbb{N}_0$. For each $e \in E$ we denote by T_e be the set of all tasks $i \in T$ such that $P(i)$ contains e. The goal is to select a subset $T' \subseteq T$ of tasks of maximum total weight $w(T')$ such that for each edge e the total demand of the selected tasks using e does not exceed $u(e)$, i.e., $d(T' \cap T_e) \leq u(e)$, where for any subset of tasks T'' we define $w(T'') := \sum_{i \in T''} w(i)$ and $d(T'') := \sum_{i \in T''} d(i)$.

UFP and its variants have applications in bandwidth allocation [BYBCR06, CHT02, LMSV00], multi-commodity demand flow [CMS07], caching [CWMX10], and resource allocation [BNBYF$^+$00, CCKR11, DPS10, PUW00]. For example one can interpret the path G as a time period subdivided into time slots, and edge capacities as a resource (such as energy or memory) that varies over time. Here each task is a job that we might want to schedule and that requires some amount of the considered resource in a given time interval. Alternatively, the path might represent a communication channel with different capacities on its links and each task corresponds to a request for using a portion of the available bandwidth along a subpath.

UFP contains knapsack as a special case (namely, for $|E| = 1$); hence it is NP-hard and in fact it is even strongly NP-hard [BSW14, CWMX10]. A lot of research was devoted to the design of approximation algorithms for UFP [AGLW13, AGLW14, BFKS09, BSW14, CCKR11, CEK09, CMS07, CHT02] and its generalizations [ACEW16, CCG$^+$14, GIU15, ACEW16, CEK09, CMS07, GVY97]. A natural idea is to partition the input tasks into small tasks and large tasks: intuitively, a task i is *large* if its demand is relatively large compared to the capacity of some edge e on $P(i)$, and *small* otherwise. If all input tasks are small, then LP-rounding yields a $(1 + \varepsilon)$-approximation as shown by Chekuri, Mydlarz, and Shepherd [CMS07] and there is a complex dynamic program (DP) that provides a PTAS for the case where all input tasks are large, due to Anagnostopoulos, Grandoni, Leonardi, and Wiese [AGLW14]. Combining the latter two results one obtains a $2 + \varepsilon$ approximation for UFP.

Since the problem admits a QPTAS [BCES06, BGK$^+$15] it is likely that there is also a PTAS for it: finding one is a major open problem in the area. PTASs are known for some special cases [BGK$^+$15, GMWZ17]. In order to obtain a PTAS, one has to combine small and large tasks. A first step in this direction was made by the authors of [GMWZ18], where they introduced a new notion of *boxed solution* (which will be crucial also in this paper), see Figure 1. A box B is characterized by a subpath $P(B)$ and a demand $d(B) \in \mathbb{N}_0$. The demand $d(B)$ is large enough so that, roughly speaking, B behaves like a large task. Intuitively, boxes play the role of *place-holders*: when a box is added to a solution under construction, then it has to respect edge capacities together with previously selected large tasks and previously created boxes. Small tasks can then be assigned to a box B if they form a feasible solution for the UFP instance defined on the path $P(B)$ with uniform edge capacities $d(B)$.

In [GMWZ18] it is shown that there exists a boxed solution with a laminar structure of profit $\mathrm{opt}_L + \frac{1}{2}\mathrm{opt}_S$ (neglecting factors of $1 - \varepsilon$), where opt_L and opt_S is the profit due to large and small tasks, resp., in the optimal solution. Finding the best such solution is however a non-trivial task. There is a DP for this in [GMWZ18] which loses an extra factor $2/3$ of the (residual) profit from the small tasks. Altogether this gives a solution of profit $\mathrm{opt}_L + \frac{1}{3}\mathrm{opt}_S$. Combining this with the mentioned PTAS for small tasks [CMS07], one obtains a $(5/3 + \varepsilon)$-approximation for UFP, which currently is the best know polynomial time result.

In order to achieve a PTAS with the above approach one would need: (1) a stronger structural result about boxed solutions that preserves almost all the optimal profit $\mathrm{opt} = \mathrm{opt}_L + \mathrm{opt}_S$, and (2) an efficient algorithm to compute an almost optimal solution of the mentioned type. As we will see, in this paper we solve (1) and make progress on (2).

1.1 Our Contribution

Our main result is an improved approximation algorithm for UFP.

THEOREM 1. *There is a polynomial time* $1 + \frac{1}{e+1} + \varepsilon < 1.26895 + \varepsilon$ *approximation algorithm for UFP.*

The above result is based on the following two main technical contributions.

A Loss-Less Boxed Solution. We show that there exists a *loss-less* boxed solution that achieves a profit of essentially opt. Intuitively, in this solution each edge e is used by $O_\varepsilon(1)$ boxes only, which will turn out to be extremely helpful. For proving this, we build on a method by Buchsbaum, Karloff, Kenyon, Reingold, and Thorup [BKK$^+$04]. Intuitively, this method processes the edges in some order and for each edge e it assigns almost all tasks using e into some boxes, such that the size of these boxes corresponds to the total demand of the tasks using e. The remaining tasks using e are assigned into some extra boxes which later will be used also by tasks that do not use e. Using it directly would yield solutions in which an edge can be used by up to $\Omega(\log n)$ boxes, which is too much for our purposes. Instead, we introduce an alternative

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

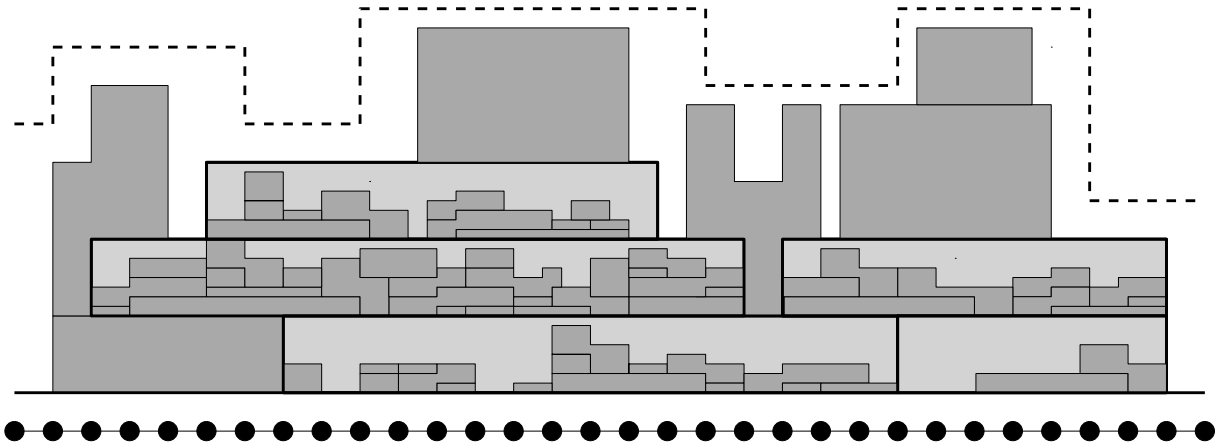

Figure 1: A boxed solution. Small tasks are assigned into the (light gray) boxes, large tasks are not assigned into boxes. The dashed lines denote the edge capacities.

ordering in which the edges are processed. As a result, we use the space in the extra boxes more efficiently and in particular, we waste less space inside them. Our structural result might turn out to be useful for future work on UFP and on related problems.

The UFP-Game. The second main technical contribution of this paper is a radically new way to compute solutions for UFP. Consider first the following naive approach. The tasks are considered in increasing order of the leftmost edge on their respective paths $P(i)$. For each task i we decide whether we select i (in which case we assign i to a box if i is small, possibly creating a new box for this purpose). We make sure that our selected tasks and the boxes respect the edge capacities and the capacities of the boxes. A simple DP can implement this strategy optimally. When we consider a task i, then there are at most $O_\varepsilon(1)$ previously selected large tasks and boxes that use the considered leftmost edge of $P(i)$ and hence interfere with i. Thus, we can afford to *remember* them in the states of the DP. Unfortunately, this is not the case for small tasks: there can be $\Omega(n)$ small tasks using an edge e in a solution. In order to remember them, the DP table would need to have exponential size.

In order to circumvent this issue—at least partially—we introduce a novel *randomized sketching* technique. This can be seen as a *solitary game* in which the tasks are presented to a player in the same fashion as above. However, the space occupied by each small task i is not its demand $d(i)$, but rather a *random demand* $d'(i)$ that is revealed to the player *only after her irrevocable decision* to select i or not. In more details, $d'(i)$ is set to zero (i.e., i is *rounded down*) with some probability and otherwise set to a relatively large value, similar to the size of a large task (i.e., i is *rounded up*), so that $\mathbb{E}[d'(i)] = d(i)$. For a large task i we always have that $d'(i) = d(i)$. The player needs to ensure that the tasks obey the capacities according to their random demands $d'(i)$, rather than their original demands $d(i)$. Therefore, she does not need to remember the previously selected rounded down tasks (since they have zero rounded demand), and can afford to remember the previously selected rounded up tasks using the current edge e in the state of a DP (since they are only constantly many). Hence, on the one hand, we can compute the optimal strategy for the player (maximizing the expected profit of the selected tasks) via a DP in polynomial time, which does not work for solving the given UFP instance directly. On the other hand, in the game the tasks behave similarly as in the given UFP instance since $\mathbb{E}[d'(i)] = d(i)$ for each task i (and for large tasks even deterministically $d'(i) = d(i)$). Furthermore, it turns out that the optimal strategy obtains an expected profit of (essentially) at least opt.

Intuitively, rounded up small tasks serve as a probabilistic *sketch* of the previously selected small tasks. In particular, the expected random demand of the selected tasks using each edge e equals their actual demand. Unfortunately, it might be that the tasks selected by the player do not form a feasible solution (e.g., when selected small tasks are rounded down too often) and it can be that these events contribute a significant portion to the expected profit of the player. Still, using various probabilistic arguments, we are able to transform the (optimal) strategy of the player into a *feasible* solution of the underlying UFP instance whose profit is roughly a $\frac{e+1}{e+2}$-fraction of the player's expected profit. We remark that, possibly with a more sophisticated (probabilistic) analysis, one might be able to derive a solution with more profit from the player's strategy. Indeed, it might even be that (some variant of) our approach provides a PTAS. This is an interesting topic for future research.

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

We believe that our randomized sketching technique might turn out to be useful to address also other problems, in particular those involving "large" and "small" objects, and when dynamic programming works if there are only large objects (the reader might think, e.g., about packing, scheduling, or resource allocation settings). For this reason it might be of independent interest.

We refer to the appendix for proofs that were omitted in the main body due to space constraints.

1.2 Preliminaries Let $\varepsilon > 0$ be a constant and assume that $1/\varepsilon \in \mathbb{N}$. By duplicating and contracting edges w.l.o.g. we can assume that each vertex is the start vertex of exactly one task or the end vertex of exactly one task. For a path P and a node v, we write $v \in P$ if P contains v, and similarly $e \in P$ for an edge e if P contains e.

We first present our results in the setting of *resource augmentation*, i.e., assuming that there is a constant $\delta > 0$ and on each edge e we are allowed to use $(1+\delta)u(e)$ units of capacity, while the compared optimal solution OPT can use only $u(e)$ units of capacity. This is sufficient to highlight our main new ideas. We will show later in Appendix C how to extend our results to the case without resource augmentation by exploiting the *slack lemma* introduced in [GMWZ17]. When needed, we will assume that δ is sufficiently small.

We define $u_{\max} := \max_{e\in E}\{u(e)\}$ and $u_{\min} := \min_{e\in E}\{u(e)\}$. Next, we classify tasks into *small* and *large* tasks. We say that a task i is *small* if $d(i) \le \varepsilon^2\delta u_{\min}$ and *large* if $d(i) > \varepsilon^2\delta u_{\min}$. Let $T_S, T_L \subseteq T$ denote the small and large tasks in the given instance, resp. Also, we define $\mathrm{OPT}_L := \mathrm{OPT} \cap T_L$ and $\mathrm{OPT}_S := \mathrm{OPT} \cap T_S$, and denote their respective weights by $\mathrm{opt}_L := w(\mathrm{OPT}_L)$ and $\mathrm{opt}_S := w(\mathrm{OPT}_S)$. We remark that this definition of small and large tasks is different from the respective notion in previous papers, e.g., [GMWZ17, AGLW14, BSW14]; however, it is better suited for our reasoning in this paper.

Using resource augmentation, by standard reductions we can simplify the input instance as follows. Note that the following lemma implies that each edge can be used by $O_{\varepsilon,\delta}(1)$ large tasks.

LEMMA 2. *Using $1+\delta$ resource augmentation and losing a factor $1+\varepsilon$ in the approximation guarantee, there is a polynomial time reduction from UFP to its special case where $u_{\max} \le O_{\varepsilon,\delta}(u_{\min})$ and $d(i) = 1$ for each task $i \in T_S$.*

2 A Loss-Less Boxed Solution

In this section we present our improved structural lemma about boxed solutions which are defined as follows.

DEFINITION 3. *Let $T'_L \subseteq T_L$, let $\mathcal{B}$ be a set of boxes, and let $\{T'_{S,B}\}_{B\in\mathcal{B}}$ be pairwise disjoint subsets of T_S. The pair $(T'_L, \{T'_{S,B}\}_{B\in\mathcal{B}})$ is a* boxed solution *if*

1. *the large tasks T'_L and the boxes $\mathcal{B}$ together* respect the edge capacities*, i.e., for each $e \in E$ it holds that $d(T'_L \cap T_e) + \sum_{B\in\mathcal{B}:e\in P(B)} d(B) \le u(e)$.*
2. *for each $B \in \mathcal{B}$ the tasks in $T'_{S,B}$* fit *into B, i.e., for each edge $e \in P(B)$ it holds that $d(T'_{S,B} \cap T_e) \le d(B)$.*

The same pair is a boxed solution with $(1+\delta)$ resource augmentation *if instead of property 1. for each edge e it holds only that $d(T'_L \cap T_e) + \sum_{B\in\mathcal{B}:e\in P(B)} d(B) \le (1+\delta)u(e)$.*

LEMMA 4. *Assume that $d(i) = 1$ for each task $i \in T_S$ and that $\delta^4 u_{\min} \ge 1$. There is a boxed solution $\mathrm{OPT}_{\mathrm{box}} = (\mathrm{OPT}_L, \{\mathrm{OPT}_{S,B}\}_{B\in\mathcal{B}^*})$ with $(1+\delta)$ resource augmentation with profit opt in which $d(B) = b := \lfloor \delta^4 u_{\min} \rfloor$ holds for each box $B \in \mathcal{B}^*$.*

Note that, since $u_{\max} \le O_{\varepsilon,\delta}(u_{\min})$, in $\mathrm{OPT}_{\mathrm{box}}$ each edge can be used by at most $O_{\varepsilon,\delta}(1)$ boxes. In the remainder of this section we prove Lemma 4. Assume w.l.o.g. that $\delta^4 u_{\min} \in \mathbb{N}$. Our reasoning builds on an argument of Buchsbaum et al. [BKK$^+$04]. However, by applying their reasoning directly an edge would be used by $\Omega(\log n)$ boxes, while we can afford only a constant number of boxes for our purposes. Therefore, we need to extend their reasoning. We start with OPT. Our goal is to create a set of boxes $\mathcal{B}^*$ and assign the tasks in OPT_S into the boxes in $\mathcal{B}^*$ such that intuitively for each edge e the total demand of the boxes in $\mathcal{B}^*$ using e essentially equals the total demand of the tasks in OPT_S using e. To construct $\mathcal{B}^*$, we will repeatedly invoke the following lemma from Buchsbaum et al. [BKK$^+$04]. Intuitively, it states that if we have a set of tasks $\bar{T} \subseteq \mathrm{OPT}_S \cap T_e$ for some edge e, then one can construct a set of boxes $\bar{\mathcal{B}}$ such that essentially all tasks from $\bar{T}$ fit into $\bar{\mathcal{B}}$ and on each edge e' the total demand of the boxes in $\bar{\mathcal{B}}$ using e' is essentially the same as the demand of the tasks in $\bar{T}$ using e'.

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

LEMMA 5. (BUCHSBAUM ET AL. [BKK+04]) *Given a set $\bar{T}$ of tasks with $d(i) = 1$ for each $i \in \bar{T}$ such that $\bar{T} \subseteq T_e$ for some edge e, and an integer box height parameter b, and a constant $\hat{\delta} > 0$. There exists a subset $\bar{T}' \subseteq \bar{T}$ with $|\bar{T} \setminus \bar{T}'| \leq 2b/\hat{\delta}^2$, a set $\bar{\mathcal{B}}$ of boxes with $d(B) = b$ for each $B \in \bar{\mathcal{B}}$, and a partition $\{\bar{T}'_B\}_{B \in \bar{\mathcal{B}}}$ of $\bar{T}'$ such that for each $B \in \bar{\mathcal{B}}$ the set $\bar{T}'_B$ fits into B, and such that for each edge f it holds that*

$$b \cdot |\{B \in \bar{\mathcal{B}} \mid f \in P(B)\}| \leq |\bar{T}' \cap T_f| + 4\hat{\delta}|\bar{T} \cap T_f|. \tag{2.1}$$

We will use Lemma 5 repeatedly in order to define the boxes. We start with an edge e^* that in the first iteration we may define arbitrarily. We invoke Lemma 5 with $\bar{T} := \mathrm{OPT}_S \cap T_e$, $\hat{\delta} := \delta/8$, and $b = \lfloor \delta^4 u_{\min} \rfloor$. We obtain the set of boxes $\bar{\mathcal{B}}$ into which fit the tasks in $\bar{T}'$. Inequality (2.1) implies that on each edge f the total demand of the boxes in $\bar{\mathcal{B}}$ using f is bounded by $(1 + 4\hat{\delta})|\bar{T} \cap T_f|$. Intuitively, their demand is justified by the demand of the tasks in $\bar{T}$ using f. However, there are the (few) remaining tasks $\bar{T} \setminus \bar{T}'$. They are so few that they fit into only $\lceil |\bar{T} \setminus \bar{T}'|/b \rceil \leq 2/\hat{\delta}^2$ extra boxes: We create $\lceil |\bar{T} \setminus \bar{T}'|/b \rceil$ additional *special boxes* $\bar{\mathcal{B}}'$ (each with demand b) such that the path of each of them equals G. We assign the tasks in $\bar{T} \setminus \bar{T}'$ to $\bar{\mathcal{B}}'$. Note that the total demand of $\bar{\mathcal{B}}'$ on some edge f might be much bigger than the demand of the tasks in $\bar{T}$ using f (in terms of the ratio between the two values). In fact, there can even be an edge f such that $\bar{T} \cap T_f = \emptyset$ even though each box in $\bar{\mathcal{B}}'$ uses f. Later, we might assign some additional tasks to the boxes $\bar{\mathcal{B}}'$ such that the unused space in them will not be lost; however we will not assign more tasks to the boxes $\bar{\mathcal{B}}$.

We recurse on the left and on the right of e^*. When we recurse on the left (the right case being symmetric), we invoke Lemma 5 on a carefully chosen edge e^*_L on the left of e^* and on the tasks $\bar{T}_L := T_{e^*_L} \cap \mathrm{OPT}_S \setminus \bar{T}$. Note that in the boxes $\bar{\mathcal{B}}'$ there might be empty space. However, we cannot easily use this space for the tasks in $\bar{T}_L$ since this space is potentially fragmented, i.e., there can be tasks in $\bar{T}_L$ that we cannot assign to some box $B \in \bar{\mathcal{B}}'$ since they do not fit together with the task that are already in B. Therefore, we say that the space used by the boxes $\bar{\mathcal{B}}'$ is *wasted.* However, we want to use the empty space in the boxes $\bar{\mathcal{B}}'$ later. To this end, we define e^*_L to be an edge on the left of e^* such that at least $\lfloor |\bar{T} \setminus \bar{T}'|/2 \rfloor$ of the tasks in $\bar{T} \setminus \bar{T}'$ start on the left of e^*_L and at least $\lfloor |\bar{T} \setminus \bar{T}'|/2 \rfloor$ of them start on the right of e^*_L. We apply Lemma 5 on e^*_L and $\bar{T}_L$. Due to our choice of e^*_L, in the next level of the recursion essentially only *half* of the space used by the boxes $\bar{\mathcal{B}}'$ is wasted: on each edge on the left of e^*_L we have $b|\bar{\mathcal{B}}'|/2$ units of completely free and unfragmented capacity in the boxes $\bar{\mathcal{B}}'$ which we can use for the tasks whose path lies completely on the left of e^*_L. More precisely, we can assign *every* set of $b|\bar{\mathcal{B}}'|/2$ such tasks to the empty space of the boxes $\bar{\mathcal{B}}'$. On the other hand, on each edge between e^*_L and e^* the tasks in $\bar{T} \setminus \bar{T}'$ use essentially $b|\bar{\mathcal{B}}'|/2$ units of capacity in the boxes $\bar{\mathcal{B}}'$. Hence, only half of the space in the boxes $\bar{\mathcal{B}}'$ is only partially used (potentially in a fragmented way) and therefore wasted.

We continue recursively. Intuitively, in each recursion level we assign the corresponding tasks $\bar{T} \setminus \bar{T}'$ to the unfragmented empty space from special boxes from previous levels or to additional special boxes if the former space is not sufficient. If the unfragmented empty space is at least $2b/\hat{\delta}^2$ then we do not create any new special boxes. Otherwise, we create at most $2/\hat{\delta}^2$ new special boxes. We select the edges to recurse such that the wasted space due to the current special boxes reduces by a factor 2 in the next iteration. Since in each iteration we create at most $2/\hat{\delta}^2$ new special boxes we maintain the invariant that on each edge e the total wasted space in special boxes is bounded by $O(b/\hat{\delta}^2)$. Therefore, we achieve that on each edge e the total size of the boxes in $\mathcal{B}^*$ is bounded by $(1 + O(\delta))|\mathrm{OPT}_S \cap T_e| + O(b/\hat{\delta}^2)$.

Let $\{OPT_{S,B}\}_{B \in \mathcal{B}^*}$ be the resulting assignment of small tasks in OPT_S to boxes $\mathcal{B}^*$. Due to our resource augmentation, the boxes $\mathcal{B}^*$ and the large tasks OPT_L together respect the edge capacities. Furthermore, by construction, the sets $\mathrm{OPT}_{S,B}$ fit in the respective box B. Hence $(\mathrm{OPT}_L, \{\mathrm{OPT}_{S,B}\}_{B \in \mathcal{B}^*})$ is a boxed solution with $1 + \delta$ resource augmentation. See Appendix A.2 for a more detailed proof.

3 The UFP-Game

In this section we restrict our attention to the setting of $(1 + \delta)$-resource augmentation for some $\delta > 0$. Due to Lemma 2 we can assume that the capacities are in a constant range (i.e., $u_{\max} \leq O_{\varepsilon,\delta}(u_{\min})$) and that $d(i) = 1$ for each $i \in T_S$. By Lemma 4 we know that there exists a boxed solution with optimal profit opt. In Section 3.1 we will define the *UFP-game* that has some similarities with computing a feasible boxed solution. We describe how to compute an optimal strategy to play this game and that with this strategy one can win essentially opt in expectation. In Section 3.2 we show how to convert this strategy into a feasible UFP solution with profit roughly $\frac{e}{e+1}\mathrm{opt}$. In Appendix B we describe a refined procedure that achieves profit roughly $\frac{e+1}{e+2}\mathrm{opt}$. Using known techniques from the literature it is possible to extend the above results (with essentially the same approximation factors) to the setting with general edge capacities and without resource augmentation: this is discussed in Appendix C.

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

3.1 The UFP-Game and How to Play it Optimally We define the following solitary game, the *UFP-game*, that has similarities with computing a boxed solution. Let ε' be a small enough constant depending on ε and δ. The input tasks are presented to a player ordered increasingly by their start vertices. Given a task i with leftmost edge e, the player needs to decide whether she wants to select i or not. In the first case she gets a profit of $w(i)$, otherwise she gets 0. If she accepts i and i is small, she may create a box B with $P(B)$ starting at e with $d(B) = b := \lfloor \delta^4 u_{\min} \rfloor$. Then she assigns i to some box B' with $P(i) \subseteq P(B')$ (it may be that $B' \neq B$). *Afterwards*, a value $d'(i)$ is defined which is the rounded demand of i in this game. If i is large then simply $d'(i) = d(i)$. If i is small, then $d'(i)$ is defined randomly: with probability $p_{up} = \frac{1}{\varepsilon' b}$ it holds that $d'(i) = \varepsilon' \cdot b$ (i is *rounded up*) and otherwise $d'(i) = 0$ (i is *rounded down*). Notice that $\mathbb{E}[d'(i)] = d(i)$.

While the player plays, she has to respect the following *rules* at any point of time:

1. the total demand of the selected large tasks and the created boxes using any edge e is at most $(1+\delta)u(e)$;
2. for each box B and each edge $e \in P(B)$ the total rounded demand $d'(\cdot)$ of the small tasks assigned to B using e is at most $d(B)$. In other words, there are at most $1/\varepsilon'$ such tasks.

In particular, the player cannot create a new box or select a large task if this would violate Rule 1. Similarly, when a small task i is assigned to a box B, the player must ensure that Rule 2 is satisfied, even in the unfortunate event that i is rounded up afterwards. We also remark that the tasks selected by the player do *not* necessarily define a feasible boxed solution. Indeed, the demand of small tasks assigned to a given box B could exceed the capacity on some edge $e \in P(B)$ when such tasks are rounded down more frequently than in expectation.

Since $\mathbb{E}[d'(i)] = d(i)$ for each task i (and even deterministically $d'(i) = d(i)$ if i is large), in the game the tasks behave similarly as in the given UFP instance. On the other hand, in the following we show how to compute the optimal strategy for playing the game with a polynomial time DP (while we cannot directly solve the given UFP instance like this). When deciding whether we should select a given task i or not, we should take into account which large tasks and boxes were selected previously, and which rounded up tasks are assigned to each such box. However, among them only those tasks and boxes matter whose respective paths use the first edge e of $P(i)$, since the other ones do not interact with i and neither with any task i' that will be presented to the player in the future (due to our ordering of the tasks). This motivates the following definition of a *state* which captures the relevant information for the player when she has to decide to select a task i or not:

- a task i, let e be the first edge of $P(i)$,
- a set L of previously selected large tasks, where for each $i' \in L$ it holds that $e \in P(i')$,
- a set $\mathcal{B}$ of previously created boxes, where for each $B \in \mathcal{B}$ it holds that $e \in P(B)$,
- pairwise disjoint sets S_B^{up} for each $B \in \mathcal{B}$ consisting of previously selected rounded up small tasks using edge e that fit into B. Let S^{up} be their union.

For each state v there are several states that can be reached next, depending on the decision of the player and on the random definition of $d'(i)$. It is convenient to represent this by the following *game DAG*. In our DAG there is a node v for each possible state $(i, L, \{S_B^{up}\}_{B\in\mathcal{B}})$; in the sequel, we will identify nodes with states. In a state $v = (i, L, \{S_B^{up}\}_{B\in\mathcal{B}})$, there are potentially several possible decisions for the player where each decision D specifies whether i is accepted or not, if i is accepted and i is small whether a new box B is created and into which of the boxes in $\mathcal{B}$ (or $\mathcal{B} \cup \{B\}$ if B is created in this step) the task i is assigned. Each possible decision D at state v leads to one or two possible *subsequent* states. In more detail, there is one subsequent state $\mathrm{next}_v(D)$ if i is large, or i is discarded, or i ends at edge e (and hence for sure does not belong to the set S^{up} of the next state). In the remaining cases there are two possible subsequent states $\mathrm{next}_v^{\mathrm{up}}(D)$ and $\mathrm{next}_v^{\mathrm{dw}}(D)$ depending on whether i is rounded up or down, resp. (which is not yet known at state v). We add a directed arc a from v to each of the nodes $\mathrm{next}_v(D)$, $\mathrm{next}_v^{\mathrm{up}}(D)$ and $\mathrm{next}_v^{\mathrm{dw}}(D)$ for each possible decision D at state v, and label these arcs with a probability $p(a)$ equal to 1, p_{up} and $1 - p_{up}$, resp. We define the node $r := (i_0, \emptyset, \{\}_{B\in\emptyset})$ to be the root of the DAG where i_0 is the input task that is presented first (i.e., having leftmost start vertex). For notational convenience we introduce a *dummy sink node* s, and add arcs from all states with $i = i_n$ to s labelled with probability 1, where i_n is the input task that is presented last (i.e., having rightmost start vertex).

We define a *strategy* $D(\cdot)$ of the player as a function that maps each state v to the decision $D(v)$ that the player makes if state v is reached. Given a strategy, we can naturally define a *strategy DAG*, which is a subgraph of the game DAG: the node set is the same as for the game DAG and for each node v we keep only the one or two outgoing arcs connecting v to $\mathrm{next}_v(D(v))$ or to $\mathrm{next}_v^{\mathrm{up}}(D(v))$ and $\mathrm{next}_v^{\mathrm{dw}}(D(v))$ depending on $D(v)$ (see Figure 3 in the appendix). To avoid pathological special cases later, we remove all nodes that cannot be reached from the root and we remove also their adjacent arcs. Let $\mathcal{Q}$ denote all paths from r to s in the strategy DAG. Notice that when the player plays the UFP-game this

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

corresponds to constructing a random walk $Q^* \in \mathcal{Q}$ in the strategy DAG. The walk Q^* starts at the root r. Given that state $v \neq s$ is reached, the next state is chosen according to the probability distribution over the outgoing edges: a single edge with probability 1, or two edges with probabilities p_{up} and $1 - p_{up}$, resp. Each path $Q \in \mathcal{Q}$ has an associated profit $w(Q)$ given by the total weight of the tasks selected along Q, and an associated probability $p(Q)$ to be taken (the product of the probabilities of its edges). Thus, the expected profit of the player is $\text{play} := \mathbb{E}[w(Q^*)] = \sum_{Q \in \mathcal{Q}} p(Q)w(Q)$. We say that a player's strategy is *optimal* if it maximizes play. It is easy to compute an optimal strategy via dynamic programming.

LEMMA 6. *We can compute an optimal strategy $D^*(\cdot)$ in time $n^{O_{\varepsilon,\delta,\varepsilon'}(1)}$.*

Let play^* denote the expected profit of $D^*(\cdot)$. We show next that play^* is essentially at least as large as opt, assuming that ε' is sufficiently small. Intuitively, one strategy achieving such a profit is the following. Take the boxed solution $\text{OPT}_{\text{box}} = (\text{OPT}_L, \{OPT_{S,B}\}_{B \in \mathcal{B}^*})$ due to Lemma 4. The player selects OPT_L and creates boxes $\mathcal{B}^*$. Furthermore, we remove a small profit subset of the tasks $\text{OPT}_{S,B}$ assigned to each $B \in \mathcal{B}^*$ such that the remaining tasks $\text{OPT}'_{S,B}$ leave a free space $\varepsilon d(B)$ on each edge of $P(B)$. The player selects small tasks from sets $\text{OPT}'_{S,B}$ only, and assigns them to B whenever possible. The free space in each box B makes it very unlikely (by Chernoff's bound) that a task $i \in OPT'_{S,B}$ is rejected because of too many previously selected rounded up tasks, hence the expected profit is at least $(1 - O(\varepsilon))\text{opt}$.

LEMMA 7. *If $\varepsilon' \leq \frac{\varepsilon^2}{3(1-\varepsilon)\ln(1/\varepsilon)}$ we have that $\text{play}^* \geq (1 - O(\varepsilon))\text{opt}$.*

3.2 From a Strategy to a Solution In this subsection we show how to convert a strategy of the player with expected profit play into a feasible UFP solution (under $1 + \delta$ resource augmentation) with a profit close enough to play. Notice that just simulating the player and selecting all tasks that she selects does not necessarily work since the resulting solution might not respect the edge capacities, e.g., when too many small tasks are rounded down. Instead, we will prove the following lemma which together with Lemmas 6 and 7 will yield a $(\frac{e+1}{e} + O(\varepsilon))$-approximation under resource augmentation.

LEMMA 8. *Given a strategy of the player with expected profit* play*, there exists a randomized polynomial-time algorithm that computes a feasible UFP solution under $(1+\delta)$-resource augmentation with expected profit at least $(\frac{e}{e+1} - O(\varepsilon))$play.*

In the remainder of this subsection we prove Lemma 8. Fix a strategy of the player and consider an execution of the UFP-game with this strategy, corresponding to a random path Q^* in the strategy DAG. We define the following random variables and probabilities. We define $X_i := 1$ if the player selects task i and $X_i := 0$ otherwise; let $x_i = \mathbb{E}[X_i]$. Furthermore, for each state v we define $x_{v,i}$ to be the probability that the player selects task i and assigns it to the box $B(v)$ created at that state v, conditioning on the event that $v \in Q^*$. If no box is created in state v then we simply set $x_{v,i} := 0$. Given a strategy, it is easy to compute the probability that each state is visited by Q^*, and hence also the above probabilities.

LEMMA 9. *Given a strategy DAG, the probabilities $\{x_i, x_{v,i}\}_{i,v}$ can be computed in polynomial time.*

Let $\text{play}_L = \sum_{i \in T_L} w(i)x_i$ and $\text{play}_S = \sum_{i \in T_S} w(i)x_i$ be the expected profit of the player due to large and small tasks, resp. First, we show how to compute a solution of profit roughly play_S. To this end, we prove that the variables $\{x_i\}_{i \in T_S}$ define a feasible fractional UFP solution for the small tasks (under resource augmentation); since they all have unit demand we can round this fractional solution to an integral one without any loss.

LEMMA 10. *There is a polynomial time algorithm that computes a feasible integral solution under $(1 + \delta)$-resource augmentation with expected profit at least* play_S.

The solution due to Lemma 10 is good when play_S is *close* to play. We next describe an alternative solution which is good in the complementary case. The best of the two solutions will yield Lemma 8. Intuitively, we simulate the player playing the game and we select each large task and each box that she selects. Whenever she selects a box $B(v)$ in a state v, then we try to assign each small task i *fractionally* into $B(v)$ to an extent of $x_{v,i}$. Note that we do this even if the task i is not assigned to box $B(v)$ later. Then it might happen that we select a small task i to a total fractional extent of more than 1 (if we assign i fractionally into several boxes). To avoid this, we cap its total fractional extent at 1.

Formally, let $Q^* \in \mathcal{Q}$ be a random path in the strategy DAG. Based on Q^* we define the following *fractional* random variables $X'_i \in [0, 1]$:

- for each large task i, we set $X'_i := 1$ if the player selects i and $X'_i := 0$ otherwise;
- for each small task i, we set $X'_i := \min\left\{\sum_{v \in Q^*} x_{v,i}, 1\right\}$.

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

REMARK 1. *One can show that* $\mathbb{E}[X_i] = \mathbb{E}[\sum_{v \in Q^*} x_{v,i}]$*, although we do not explicitly use this fact. However, possibly* $\mathbb{E}[X'_i] = 1 < \sum_{v \in Q^*} x_{v,i}$*, and therefore our solution* $\{X'_i\}_{i \in T_S}$ *might obtain less profit than* play_S *in expectation. On the other hand, deterministically* $X'_i \leq 1$*, which will help us to convert* $\{X'_i\}_{i \in T}$ *into a feasible solution.*

It is relatively easy to convert X'_i into a feasible boxed solution with a small loss in the profit: for each state v the values $\{x_{v,i}\}_{i \in T_S}$ happen to define a feasible fractional solution for the box $B(v)$. Therefore, given X'_i, it is possible to define a quantity $X'_{v,i}$ for each node v such that $0 \leq X'_{v,i} \leq x_{v,i}$ and also $\sum_v X'_{v,i} = X'_i$. This yields a solution that selects large tasks and boxes integrally and small tasks fractionally. Finally, we can use standard LP-rounding techniques to convert this into a feasible integral solution.

LEMMA 11. *In polynomial time we can compute a feasible integral solution under* $(1+\delta)$*-resource augmentation with expected profit at least* $\sum_{i \in T_L} w(i)\mathbb{E}[X'_i] + (1 - O(\varepsilon)) \sum_{i \in T_S} w(i)\mathbb{E}[X'_i]$.

Observe that $\mathbb{E}[w(i)X_i]$ and $\mathbb{E}[w(i)X'_i]$ are the expected profit of the player's strategy and of the fractional solution $\{X'_i\}_{i \in T}$ due to task i, respectively. Our goal is to show that $\frac{\mathbb{E}[w(i)X_i]}{\mathbb{E}[w(i)X'_i]} = \frac{\mathbb{E}[X_i]}{\mathbb{E}[X'_i]}$ is sufficiently small. This is done in the following lemma, which is the technical heart of our argumentation.

LEMMA 12. *For each small task* i *it holds that* $\mathbb{E}[X'_i] \geq (1 - \frac{1}{e})\mathbb{E}[X_i]$.

REMARK 2. *Notice that there might be alternative ways to turn the strategy of the player into a feasible solution with a lower value of* $\mathbb{E}[X_i]/\mathbb{E}[X'_i]$*. Indeed, in Appendix B we describe a more complex transformation that decreases the latter quantity to* $\frac{e+2}{e+1}$*. This might even lead to a PTAS.*

Lemma 12 implies that the solution due to Lemma 11 has a profit of at least $\text{play}_L + (1 - \frac{1}{e} - O(\varepsilon))\text{play}_S$ (since by definition $\sum_{j \in T_L} w(j)\mathbb{E}[X'_j] = \text{play}_L$). Then this or the solution due to Lemma 10 has a profit of at least $(\frac{e}{e+1} - O(\varepsilon))\text{play}$ (the worst case is achieved for roughly $\text{play}_L = \text{play}_S/e$). This completes the proof of Lemma 8.

In the reminder of this subsection we prove Lemma 12. Let us focus on a specific small task i. In order to simplify the analysis, it is convenient to express $\mathbb{E}[X_i]$ and $\mathbb{E}[X'_i]$ in a different form. First of all, we consider the prefix tree[1] T of the set $\mathcal{Q}$ of root-to-sink paths in the strategy DAG. In particular, T contains precisely one path for each possible prefix of a path in $\mathcal{Q}$. While T might have exponential size, we remark that we use T only for the analysis.

Let $\tilde{r}$ denote the root of T. Observe that the same node v of the strategy DAG might correspond to multiple nodes of T. Whenever we refer to a node $\tilde{v}$ in T, we let v denote the corresponding node in the strategy DAG. We let $T(\tilde{w})$ be the path from the root to $\tilde{w}$. Notice that each leaf $\tilde{s}$ of T corresponds to the unique sink node s in the strategy DAG.

Notice that there exists a bijection between the root-to-sink paths $\mathcal{Q}$ in the strategy DAG and the root-to-leaf paths $\tilde{Q}$ in T: for a path $Q \in \mathcal{Q}$ we denote by $\tilde{Q}$ the corresponding path in $\tilde{\mathcal{Q}}$ and vice versa. If $Q \in \mathcal{Q}$ assigns the task i to some box $B(v)$, we call the last node $\tilde{s}$ of $\tilde{Q}$ a $B(v)$*-success* node, or a *success* node for short. Otherwise $\tilde{s}$ is a *fail* node.

With this new notation in mind, we can redefine all the relevant quantities in a natural way. Recall that each edge $e = (v, u)$ of the strategy DAG is labeled with some probability $p(e) =: p(v, u)$ which is the probability of reaching state u given that state v is reached. We label each edge $\tilde{e} = (\tilde{v}, \tilde{u})$ in T with the probability $p(\tilde{e}) := p(\tilde{v}, \tilde{u}) := p(e)$. Given any subpath P of T, let $p(P) = \prod_{\tilde{e} \in P} p(\tilde{e})$. Notice that the probability that $Q \in \mathcal{Q}$ is sampled in the strategy DAG is precisely $p(\tilde{Q})$. So we can rather focus on sampling a random root-to-sink path $\tilde{Q}^* \in \tilde{\mathcal{Q}}$ in T according to the probabilities p.

We introduce now random variables Y_i, Y'_i for T which intuitively have the same meaning as X_i, X'_i in the strategy DAG. Formally, let $\tilde{Q}^* \in \tilde{\mathcal{Q}}^*$ be the random sampled path. We set $Y_i := 1$ if $\tilde{Q}^*$ ends in a success node and $Y_i := 0$ otherwise. For each internal node $\tilde{v}$, we define $y_{\tilde{v},i}$ to be the probability that the player selects task i and assigns it to the box $B(v)$ created at state v, conditioning on the event that $\tilde{v} \in \tilde{Q}^*$; we define $y_{\tilde{v},i} = 0$ if no box is created at state v. We define the random variable $Y'_i := \min\left\{\sum_{\tilde{v} \in \tilde{Q}^*} y_{\tilde{v},i}, 1\right\}$.

From the above discussion, it follows easily that $\mathbb{E}[X_i] = \mathbb{E}[Y_i]$ and $\mathbb{E}[X'_i] = \mathbb{E}[Y'_i]$, thus we can focus on upper bounding the ratio $\mathbb{E}[Y_i]/\mathbb{E}[Y'_i]$. For technical reasons, it is convenient to append to each internal node $\tilde{v}$ a dummy node $\tilde{v}^{succ}$ that we define as a $B(v)$-success node (even when $B(v)$ is not defined or it cannot contain i). We set the probability $p(\tilde{v}, \tilde{v}^{succ})$ of the new dummy edge to 0, so that this affects neither $\mathbb{E}[Y_i]$ nor $\mathbb{E}[Y'_i]$. In the following, $\tilde{\mathcal{Q}}$, the set of root-sink paths, and the random path $\tilde{Q}^*$ refer to this augmented version of T.

[1] In more detail, T can be built as follows. Let $Q_1, \ldots, Q_m$ be the paths in $\mathcal{Q}$ in arbitrary order. Initially T consists of a copy of Q_1. Then at step $i = 2, \ldots, m$ we consider $Q_i = (v_1, \ldots, v_q)$. Let $Q'_i = (v_1, \ldots, v_h)$ be the maximal prefix of Q_i such that T contains a path of type $(\tilde{v}_1, \ldots, \tilde{v}_h)$. Then we augment T by appending to node $\tilde{v}_h$ a path $(\tilde{v}_{h+1}, \ldots, \tilde{v}_q)$ consisting of new copies of nodes $v_{h+1}, \ldots, v_q$.

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

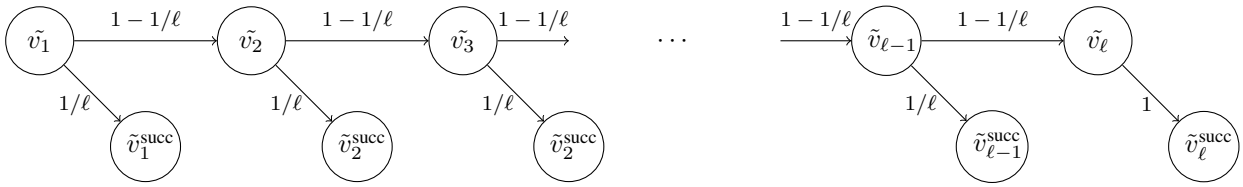

Figure 2: The worst-case situation which we obtain from Lemma 13.

In order to upper bound the ratio $\mathbb{E}[Y_i]$ and $\mathbb{E}[Y_i']$, it is convenient to see the latter quantities as functions $\mathbb{E}[Y_i(p)]$ and $\mathbb{E}[Y_i'(p)]$ of the probabilities p of the edges of T. This way we can define worst-case probabilities p^* which are edge probabilities p' that maximize $\mathbb{E}[Y_i(p')]/\mathbb{E}[Y_i'(p')]$, under the constraint that the sum of the probabilities on edges leaving each internal node is 1 (so that the random path $\tilde{Q}^* = \tilde{Q}^*(p')$ in T according to p' is well-defined). It turns out that p^* has a very simple support, as proved in the following lemma.

LEMMA 13. *There are worst-case probabilities p^* satisfying the following properties:*

1. *For any path $\tilde{Q} \in \tilde{\mathcal{Q}}$ ending in a fail node it holds that $p^*(\tilde{Q}) = 0$ (and hence always $Y_i(p^*) = 1$).*
2. *For each internal node $\tilde{v}$, $\tilde{v}^{succ}$ is the only $B(v)$-success node that might be contained in $\tilde{Q}^*$ with positive probability.*
3. *For each internal node $\tilde{v}$, there exists at most one internal child $\tilde{u}$ of $\tilde{v}$ such that $p^*(\tilde{v}, \tilde{u}) > 0$.*

Proof. [Proof sketch.] 1. Similarly to Remark 1, we obtain less profit than the player if sometimes $Y_i' = 1 < \sum_{\tilde{v} \in \tilde{Q}^*} y_{\tilde{v},i}$. Intuitively, this is more likely to happen if the player selects task i with larger probability.
2. Intuitively, this can be shown by moving the probability mass from all paths in $\tilde{\mathcal{Q}}$ with a $B(v)$-success node to the single path in $\tilde{\mathcal{Q}}$ that ends in $\tilde{v}^{succ}$. This does not change $\mathbb{E}[Y_i]$, while one can show that $\mathbb{E}[Y_i']$ can only decrease.
3. If an internal node $\tilde{v}$ has two internal children, say $\tilde{u}_1$ and $\tilde{u}_2$, then intuitively one is worse than the other for the ratio $\mathbb{E}[Y_i]/\mathbb{E}[Y_i']$ and hence in p^* only the worse node among $\tilde{u}_1$ and $\tilde{u}_2$ is contained in $\tilde{Q}^*$ with positive probability. ∎

Let us drop all the nodes that are reached with zero probability according to p^*. Clearly this does not affect the ratio $\mathbb{E}[Y_i(p^*)]/\mathbb{E}[Y_i'(p^*)]$. The structure of the remaining part of T is very simple: the root $\tilde{r}$ and the internal nodes induce a path $\tilde{r} = \tilde{v}_1, \ldots, \tilde{v}_\ell$, and there is one edge $(\tilde{v}_j, \tilde{v}_j^{succ})$ for each internal node $\tilde{v}_j$ (see Figure 2). By induction and simple algebraic steps one can derive the worst-case probabilities p^* for this structure, which are given by $p^*(\tilde{v}_j, \tilde{v}_j^{succ}) = 1/\ell$ and $p^*(\tilde{v}_j, \tilde{v}_{j+1}) = 1 - 1/\ell$ for each $j = 1, \ldots, \ell - 1$ and additionally $p^*(\tilde{v}_\ell, \tilde{v}_\ell^{succ}) = 1$. This implies that $\frac{\mathbb{E}[Y_i(p^*)]}{\mathbb{E}[Y_i'(p^*)]} = \frac{1}{1-(1-1/\ell)^\ell} < \frac{e}{e-1}$ which yields the following lemma.

LEMMA 14. *It holds that $\frac{\mathbb{E}[Y_i(p)]}{\mathbb{E}[Y_i'(p)]} \leq \frac{\mathbb{E}[Y_i(p^*)]}{\mathbb{E}[Y_i'(p^*)]} \leq \frac{e}{e-1}$.*

The proof of Lemma 12 follows from Lemmas 13 and 14.

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

References

[ACEW16] Anna Adamaszek, Parinya Chalermsook, Alina Ene, and Andreas Wiese. Submodular unsplittable flow on trees. In *IPCO*, volume 9682 of *Lecture Notes in Computer Science*, pages 337–349, 2016.

[AGLW13] Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Andreas Wiese. Constant integrality gap LP formulations of unsplittable flow on a path. In *IPCO*, pages 25–36, 2013.

[AGLW14] Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Andreas Wiese. A mazing 2+ε approximation for unsplittable flow on a path. In *SODA*, pages 26–41, 2014.

[BCES06] N. Bansal, A. Chakrabarti, A. Epstein, and B. Schieber. A quasi-PTAS for unsplittable flow on line graphs. In *STOC*, pages 721–729. ACM, 2006.

[BFKS09] N. Bansal, Z. Friggstad, R. Khandekar, and R. Salavatipour. A logarithmic approximation for unsplittable flow on line graphs. In *SODA*, pages 702–709, 2009.

[BGK+15] Jatin Batra, Naveen Garg, Amit Kumar, Tobias Mömke, and Andreas Wiese. New approximation schemes for unsplittable flow on a path. In *SODA*, pages 47–58, 2015.

[BKK+04] Adam L Buchsbaum, Howard Karloff, Claire Kenyon, Nick Reingold, and Mikkel Thorup. Opt versus load in dynamic storage allocation. *SIAM Journal on Computing*, 33(3):632–646, 2004.

[BNBYF+00] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. In *STOC*, pages 735–744, 2000.

[BSW14] Paul Bonsma, Jens Schulz, and Andreas Wiese. A constant-factor approximation algorithm for unsplittable flow on paths. *SIAM Journal on Computing*, 43:767–799, 2014.

[BTV99] Dimitris Bertsimas, Chung-Piaw Teo, and Rakesh Vohra. On dependent randomized rounding algorithms. *Oper. Res. Lett.*, 24(3):105–114, 1999.

[BYBCR06] R. Bar-Yehuda, M. Beder, Y. Cohen, and D. Rawitz. Resource allocation in bounded degree trees. In *ESA*, pages 64–75, 2006.

[CCG+14] Venkatesan T. Chakaravarthy, Anamitra R. Choudhury, Shalmoli Gupta, Sambuddha Roy, and Yogish Sabharwal. Improved algorithms for resource allocation under varying capacity. In *ESA*, pages 222–234, 2014.

[CCKR11] Gruia Călinescu, Amit Chakrabarti, Howard J. Karloff, and Yuval Rabani. An improved approximation algorithm for resource allocation. *ACM Transactions on Algorithms*, 7:48:1–48:7, 2011.

[CEK09] C. Chekuri, A. Ene, and N. Korula. Unsplittable flow in paths and trees and column-restricted packing integer programs. In *APPROX-RANDOM*, pages 42–55, 2009.

[CHT02] B. Chen, R. Hassin, and M. Tzur. Allocation of bandwidth and storage. *IIE Transactions*, 34:501–507, 2002.

[CMS07] C. Chekuri, M. Mydlarz, and F. Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Transactions on Algorithms*, 3, 2007.

[CWMX10] M. Chrobak, G. Woeginger, K. Makino, and H. Xu. Caching is hard, even in the fault model. In *ESA*, pages 195–206, 2010.

[DPS10] A. Darmann, U. Pferschy, and J. Schauer. Resource allocation with time intervals. *Theoretical Computer Science*, 411:4217–4234, 2010.

[GIU15] Fabrizio Grandoni, Salvatore Ingala, and Sumedha Uniyal. Improved approximation algorithms for unsplittable flow on a path with time windows. In *WAOA*, pages 13–24, 2015.

[GMWZ17] Fabrizio Grandoni, Tobias Mömke, Andreas Wiese, and Hang Zhou. To augment or not to augment: Solving unsplittable flow on a path by creating slack. In *SODA*, pages 2411–2422, 2017.

[GMWZ18] Fabrizio Grandoni, Tobias Mömke, Andreas Wiese, and Hang Zhou. A (5/3 + ϵ)-approximation for unsplittable flow on a path: placing small tasks into boxes. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 607–619, 2018.

[GVY97] N. Garg, V. V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.

[LMSV00] S. Leonardi, A. Marchetti-Spaccamela, and A. Vitaletti. Approximation algorithms for bandwidth and storage allocation problems under real time constraints. In *FSTTCS*, pages 409–420, 2000.

[PUW00] C. A. Phillips, R. N. Uma, and J. Wein. Off-line admission control for general scheduling problems. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '00)*, pages 879–888. ACM, 2000.

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

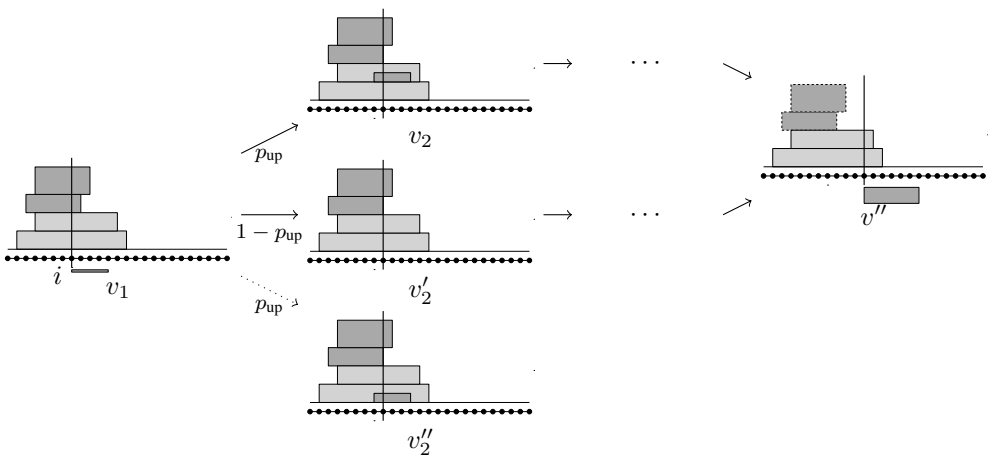

Figure 3: Part of a game DAG, where solid arrows indicate the strategy DAG of some player. In particular, the player selects a small task i in state v_1 and places it in the top box. With probability p_{up}, task i is rounded up, leading to state v_2, and otherwise it is rounded down, leading to state v_2'. State v_2'' belongs to an alternative strategy where i is placed in the bottom box (and is rounded up). Notice that v_2' can also be reached by placing i in the bottom box in case i is rounded down. With the depicted strategy, assuming that in the subsequent steps the player does not select any further tasks, the two branches starting at v_2 and v_2' eventually arrive at the state v''.

A Omitted Proofs

A.1 Proof of Lemma 2 In [GMWZ17] it is shown that, using $1+\delta$ resource augmentation and losing a factor $1+\varepsilon$ in the approximation guarantee, in polynomial time one can reduce the input problem to a polynomial number of instances where demands range between $u_{min} \geq (\delta/3)^{1/\varepsilon} u_{max}$ and u_{max}. Furthermore, using $(1+\delta)$ resource augmentation, it is not hard to see that we can assume that $d(i) \geq \delta u_{min}/n$ for each i. By scaling we can further assume that $d(i)$ is an integer lower bounded by 1. We can then replace each small tasks i with $d(i)$ tasks with demand 1, path $P(i)$ and weight $w(i)/d(i)$. Once a solution for this problem is found, one can convert it into a solution for the original instance using the LP-rounding techniques of Chekuri et al. [CMS07] , while losing a factor $1+O(\varepsilon)$ in the approximation.

A.2 Proof of Lemma 4 We use a recursive methods which is based on sub-problems of the following form. We are given

- a subpath G',
- a set T' of tasks such that $P(i) \subseteq G'$ for each $i \in T'$,
- a set of boxes $\mathcal{B}'$ such that $G' \subseteq P(B)$ and $d(B) = \delta^4 u_{\min}$ for each box $B \in \mathcal{B}$,
- for each box $B \in \mathcal{B}'$ a set of tasks $T(B)$ (that intuitively were previously assigned to B) where
  - $T' \cap T(B) = \emptyset$,
  - each task $i \in T(B)$ uses the leftmost edge of G' or the rightmost edge of G',
  - let $T^{(L)}(B) \subseteq T(B)$ denote the tasks in $T(B)$ that use the leftmost edge of G' but not the rightmost edge of G',
  - let $T^{(R)}(B) \subseteq T(B)$ denote the tasks in $T(B)$ that use the rightmost edge of G' but not the leftmost edge of G',
  - let $T^{(C)}(B)$ use each edge of G',
  - it holds that $|T^{(L)}(B)| + |T^{(R)}(B)| + |T^{(C)}(B| \leq d(B)$ (and hence $T^{(L)}(B) \cup T^{(R)}(B) \cup T^{(C)}(B$ fit into B).

Our method returns a set of boxes $\mathcal{B}''$ and an assignment of all tasks $i \in T'$ into $\mathcal{B}' \cup \mathcal{B}''$, i.e., a partition $\{T'_B\}_{B \in \mathcal{B}' \cup \mathcal{B}''}$ of T', such that for each box $B \in \mathcal{B}' \cup \mathcal{B}''$ the tasks in $T(B) \cup T'_B$ fit into B. In the main call of the algorithm we define $G' := G$, $T' := \mathrm{OPT}_S$, and $\mathcal{B}' := \emptyset$.

For each box $B \in \mathcal{B}$, we define $\mathrm{avail}(B) := d(B) - |T^{(L)}(B)| - |T^{(R)}(B)| - |T^{(C)}(B)|$, which implies that any set of $\mathrm{avail}(B)$ tasks i with $P(i) \subseteq G'$ fits into B together with $T^{(L)}(B), T^{(R)}(B)$ and $T^{(C)}(B)$. Assume w.l.o.g. that $\sum_{B \in \mathcal{B}'} \left|T^{(L)}(B)\right| \geq \sum_{B \in \mathcal{B}'} \left|T^{(R)}(B)\right|$. We define $e^* \in G'$ to be an edge such that at least $\left\lfloor \frac{1}{2} \sum_{B \in \mathcal{B}'} |T^{(L)}(B)| \right\rfloor$ tasks from $\bigcup_{B \in \mathcal{B}'} T^{(L)}(B)$ have their end vertex on the left of e^* (this end vertex could be the left vertex incident to e^*) and

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

at least $\left\lfloor \frac{1}{2} \sum_{B \in \mathcal{B}'} |T^{(L)}(B)| \right\rfloor$ tasks from $\bigcup_{B \in \mathcal{B}'} T^{(L)}(B)$ have their end vertex on the right of e^* (this end vertex could be the right vertex incident to e^*). We apply Lemma 5 to $\bar{T} := T' \cap T_{e^*}$ with $\hat{\delta} := \delta/8$. The boxes $\bar{\mathcal{B}}$ due to Lemma 5 will belong to the boxes for the claimed boxed solution and the latter will use the assignment of the tasks $\bar{T}'$ into $\bar{\mathcal{B}}$ due to Lemma 5. We assign the tasks in $\bar{T} \setminus \bar{T}'$ greedily into the boxes $\mathcal{B}'$ such that into each box $B \in \mathcal{B}'$ we assign at most $\mathrm{avail}(B)$ tasks from $\bar{T} \setminus \bar{T}'$. If $|\bar{T} \setminus \bar{T}'| \leq \sum_{B \in \mathcal{B}'} \mathrm{avail}(B)$ then in this way we assign all tasks from $\bar{T} \setminus \bar{T}'$. Otherwise, we create $\left\lceil \left| |\bar{T} \setminus \bar{T}'| - \sum_{B \in \mathcal{B}'} \mathrm{avail}(B) \right| / b \right\rceil$ new boxes B with $P(B) = G'$ and with demand b and we assign the remaining tasks from $\bar{T} \setminus \bar{T}'$ into these new boxes. Denote by $\mathcal{B}''$ these new boxes. For each box $B \in \mathcal{B}' \cup \mathcal{B}''$ let $\tilde{T}(B)$ denote the tasks from $\bar{T} \setminus \bar{T}'$ that we assigned into B.

Then we recurse. Let G'_L denote the subpath of G' on the left of e^* (excluding e^*) and let G'_R denote the subpath of G' on the right of e^* (excluding e^*). When we recurse on G'_L, the parameters of the recursion are the tasks in $T'_L := \{i \in T' \mid P(i) \subseteq G'_L\}$, the set of boxes $\mathcal{B}'_L := \mathcal{B}' \cup \mathcal{B}''$, and for each box $B \in \mathcal{B}' \cup \mathcal{B}''$ the set $T_L(B) := (T(B) \cup \tilde{T}(B)) \cap \{i \in T \mid P(i) \cap G'_L \neq \emptyset\}$. We recurse on $(G'_L, \mathcal{B}'_L, T'_L, \{T_L(B)\}_{B \in \mathcal{B}_L})$. We define the right subproblem $(G'_R, \mathcal{B}'_R, T'_R, \{T_R(B)\}_{B \in \mathcal{B}_R})$ symmetrically.

Consider a recursive call $(G', \mathcal{B}', T', \{T(B)\}_{B \in \mathcal{B}'})$. In each box $B \in \mathcal{B}'$ we have that the tasks in $T^{(L)}(B) \cup T^{(R)}(B)$ use the capacity in B partially but not completely. Intuitively, they prevent us from adding tasks into B since they "fragment" the remaining capacity inside B. On the one hand, we define the *waste* of this recursive call to be $\mathrm{wst}(G', \mathcal{B}', T', \{T(B)\}_{B \in \mathcal{B}'}) := \sum_{B \in \mathcal{B}'} |T^{(L)}(B)| + |T^{(R)}(B)|$. On the other hand, in each box B we have $\mathrm{avail}(B)$ units of capacity available on each edge and we define $\mathrm{avail}(G', \mathcal{B}', T', \{T(B)\}_{B \in \mathcal{B}'}) := \sum_{B \in \mathcal{B}'} \mathrm{avail}(B)$. Next we bound $\mathrm{wst}(G', \mathcal{B}', T', \{T(B)\}_{B \in \mathcal{B}'})$ and $\mathrm{avail}(G', \mathcal{B}', T', \{T(B)\}_{B \in \mathcal{B}'})$.

LEMMA 15. *For each recursive call $(G', \mathcal{B}', T', \{T(B)\}_{B \in \mathcal{B}'})$ we have that*

$$\mathrm{wst}(G', \mathcal{B}', T', \{T(B)\}_{B \in \mathcal{B}'}) + \mathrm{avail}(G', \mathcal{B}', T', \{T(B)\}_{B \in \mathcal{B}'}) \leq 12b/\hat{\delta}^2 + 4\,.$$

Proof. We prove by induction that for each recursive call $(G', \mathcal{B}, \tilde{T}, \{T(B)\}_{B \in \mathcal{B}})$ it holds that

$$\mathrm{wst}(G', \mathcal{B}', T', \{T(B)\}_{B \in \mathcal{B}'}) \leq 8b/\hat{\delta}^2 + 4 \quad \text{and that}$$

$$\mathrm{wst}(G', \mathcal{B}', T', \{T(B)\}_{B \in \mathcal{B}'}) + \mathrm{avail}(G', \mathcal{B}', T', \{T(B)\}_{B \in \mathcal{B}'}) \leq 12b/\hat{\delta}^2 + 4\,.$$

In the main call, we have that $\mathcal{B} = \emptyset$ and hence the claim is true. Suppose that the claim is true for some recursive call $(G', \mathcal{B}', T', \{T(B)\}_{B \in \mathcal{B}'})$ and let $\mathrm{wst} := \mathrm{wst}(G', \mathcal{B}', T', \{T(B)\}_{B \in \mathcal{B}'})$ and $\mathrm{avail} := \mathrm{avail}(G', \mathcal{B}', T', \{T(B)\}_{B \in \mathcal{B}'})$. Assume w.l.o.g. that $\sum_{B \in \mathcal{B}'} \left|T^{(L)}(B)\right| \geq \sum_{B \in \mathcal{B}'} \left|T^{(R)}(B)\right|$.

Let $(G'_L, \mathcal{B}'_L, T'_L, \{T_L(B)\}_{B \in \mathcal{B}_L})$ and $(G'_R, \mathcal{B}'_R, T'_R, \{T_R(B)\}_{B \in \mathcal{B}_R})$ denote the corresponding left and right subproblem, respectively. We define

$$\begin{aligned}
\mathrm{wst}_L &:= \mathrm{wst}(G'_L, \mathcal{B}'_L, T'_L, \{T_L(B)\}_{B \in \mathcal{B}_L}),\\
\mathrm{avail}_L &:= \mathrm{avail}(G'_L, \mathcal{B}'_L, T'_L, \{T_L(B)\}_{B \in \mathcal{B}_L}),\\
\mathrm{wst}_R &:= \mathrm{wst}(G'_R, \mathcal{B}'_R, T'_R, \{T_R(B)\}_{B \in \mathcal{B}_R}), \text{ and}\\
\mathrm{avail}_R &:= \mathrm{avail}(G'_R, \mathcal{B}_R, T'_R, \{T_R(B)\}_{B \in \mathcal{B}_R})\,.
\end{aligned}$$

Observe that at least $\left\lfloor \frac{1}{2} \sum_{B \in \mathcal{B}'} |T^{(L)}(B)| \right\rfloor \geq \mathrm{wst}/4 - 1$ tasks from $\bigcup_{B \in \mathcal{B}'} T^{(L)}(B)$ have their end vertex on the right of e^*. Hence, these tasks do not contribute towards the waste wst_L of the left subproblem since they use each edge of G'_L. Similarly, at least $\left\lfloor \frac{1}{2} \sum_{B \in \mathcal{B}'} |T^{(L)}(B)| \right\rfloor \geq \mathrm{wst}/4 - 1$ tasks from $\bigcup_{B \in \mathcal{B}'} T^{(L)}(B)$ have their end vertex on the left of e^* and hence these tasks do not contribute towards wst_R.

Recall that $|\bar{T} \setminus \bar{T}'| \leq 2b/\hat{\delta}^2$. If $\mathrm{avail} \geq 2b/\hat{\delta}^2$ then we do not create any new boxes $\mathcal{B}''$ for the tasks $\bar{T} \setminus \bar{T}'$. In this case we have that $\mathrm{avail}_L + \mathrm{wst}_L \leq \mathrm{avail} + \mathrm{wst} \leq 12b/\hat{\delta}^2 + 4$ and

$$\mathrm{wst}_L \leq \mathrm{wst} - \left\lfloor \frac{1}{2} \sum_{B \in \mathcal{B}'} |T^{(L)}(B)| \right\rfloor + \frac{2b}{\hat{\delta}^2} \leq \mathrm{wst} - \frac{\mathrm{wst}}{4} + 1 + \frac{2b}{\hat{\delta}^2} \leq \frac{3}{4}\mathrm{wst} + 1 + \frac{2b}{\hat{\delta}^2} \leq \frac{8b}{\hat{\delta}^2} + 4.$$

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

Similarly $\text{avail}_R + \text{wst}_R \leq \text{avail} + \text{wst} \leq 12b/\hat{\delta}^2 + 4$ and

$$\text{wst}_R \leq \text{wst} - \left\lfloor \frac{1}{2} \sum_{B \in \mathcal{B}'} |T^{(R)}(B)| \right\rfloor + \frac{2b}{\hat{\delta}^2} \leq \text{wst} - \frac{\text{wst}}{4} + 1 + \frac{2b}{\hat{\delta}^2} \leq \frac{3}{4}\text{wst} + 1 + \frac{2b}{\hat{\delta}^2} \leq \frac{8b}{\hat{\delta}^2} + 4\,.$$

If $\text{avail} < 2b/\hat{\delta}^2$ then we create at most $\left\lceil 2/\hat{\delta}^2 \right\rceil$ boxes $\mathcal{B}''$. By the same calculation as above, we have that $\text{wst}_L \leq 8b/\hat{\delta}^2 + 4$ and $\text{wst}_R \leq 8b/\hat{\delta}^2 + 4$. Moreover, we have that $\text{avail}_L + \text{wst}_L \leq \text{wst} + 2b/\hat{\delta}^2 + 2b/\hat{\delta}^2 \leq 12b/\hat{\delta}^2 + 4$ since each unit of waste or available space in the left subproblem stems from either a unit of waste or available space in the parent subproblem, or from one unit of (available) space in the boxes $\mathcal{B}''$. Similarly, we have that $\text{avail}_R + \text{wst}_R \leq \text{wst} + 2b/\hat{\delta}^2 + 2b/\hat{\delta}^2 \leq 12b/\hat{\delta}^2 + 4$. ∎

Let $\mathcal{B}^*$ denote the computed set of boxes and for each $B \in \mathcal{B}^*$ let $OPT_{S,B}$ be the tasks assigned to B. We next show that $(\text{OPT}_L, \{OPT_{S,B}\}_{B \in \mathcal{B}^*})$ is a boxed solution with $1+\delta$ resource augmentation. Intuitively, the additional term $12b/\hat{\delta}^2 + 4$ in the upper bound in Lemma 15 is compensated by resource augmentation.

LEMMA 16. *We have that* $(\text{OPT}_L, \{OPT_{S,B}\}_{B \in \mathcal{B}^*})$ *is a boxed solution with* $1+\delta$ *resource augmentation.*

Proof. By construction, for each $B \in \mathcal{B}^*$ the tasks in $\text{OPT}_{S,B}$ fit into B. It remains to bound the capacity needed by the tasks in OPT_L and by the boxes $\mathcal{B}^*$ on each edge e. Let $\text{OPT}_S^{(1)} \subseteq \text{OPT}_S$ denote the tasks in OPT_S that in their respective iteration are assigned to boxes $\bar{\mathcal{B}}$ by Lemma 5 and let $\text{OPT}_S^{(2)} := \text{OPT}_S \setminus \text{OPT}_S^{(1)}$. Let $\mathcal{B}^{(1)} \subseteq \mathcal{B}^*$ denote the boxes into which the tasks in $\text{OPT}_S^{(1)}$ are assigned, and let $\mathcal{B}^{(2)} := \mathcal{B}^* \setminus \mathcal{B}^{(1)}$. For each edge $e \in E$ we have that

$$\sum_{B \in \mathcal{B}^{(1)} : e \in P(B)} d(B) \leq d(\text{OPT}_S^{(1)} \cap T_e) + 4\hat{\delta} d(\text{OPT}_S \cap T_e)$$

and

$$\sum_{B \in \mathcal{B}^{(2)} : e \in P(B)} d(B) \leq d(\text{OPT}_S^{(2)} \cap T_e) + \frac{12b}{\hat{\delta}^2} + 4 \leq d(\text{OPT}_S^{(2)} \cap T_e) + \frac{\delta u_{min}}{2}.$$

Above we used the fact that, for δ small enough, $\frac{12b}{\hat{\delta}^2} = 12 \cdot 64 \cdot \delta^2 u_{min} \leq \frac{\delta u_{min}}{4}$ and $4 \leq \frac{1}{4\delta^3} \leq \frac{\delta}{4} u_{min}$. This yields that

$$\begin{aligned}
d(\text{OPT}_L \cap T_e) + \sum_{B \in \mathcal{B}^* : e \in P(B)} d(B) &\leq d(\text{OPT}_L \cap T_e) + \sum_{B \in \mathcal{B}^{(1)} : e \in P(B)} d(B) + \sum_{B \in \mathcal{B}^{(2)} : e \in P(B)} d(B) \\
&\leq d(\text{OPT}_L \cap T_e) + (1 + 4\hat{\delta}) d(\text{OPT}_S \cap T_e) + \delta u_{\min}/2 \\
&= d(\text{OPT}_L \cap T_e) + (1 + \delta/2) d(\text{OPT}_S \cap T_e) + \delta u_{\min}/2 \\
&\leq u(e) + \delta d(\text{OPT}_S \cap T_e)/2 + \delta u_{\min}/2 \\
&\leq u(e) + \delta u(e)/2 + \delta u(e)/2 \\
&= (1+\delta) u(e).
\end{aligned}$$

∎

A.3 Proof of Lemma 6 Let us first upper bound the number of possible states (i.e., nodes in the game DAG). Let us fix a given step t (in particular, i and e are fixed). Recall that large tasks have demand at least $\varepsilon^2 \delta u_{min}$ and that $u_{max} = O_{\varepsilon,\delta}(u_{min})$. Hence there can be at most $\frac{(1+\delta)u_{max}}{\varepsilon^2 \delta u_{min}} = O_{\varepsilon,\delta}(1)$ large tasks using e. Thus there are at most $O(n^{O_{\varepsilon,\delta}(1)})$ choices for L. Recall that each box has demand $b = \delta^4 u_{min}$. Therefore the number of created boxes using e is at most $\frac{(1+\delta)u_{max}}{b} = O_{\varepsilon,\delta}(1)$. There are at most n^2 choices for the endpoints of a given box B. Thus there are at most $O(n^{O_{\varepsilon,\delta}(1)})$ choices for $\mathcal{B}$. The number of rounded up tasks assigned to a box $B \in \mathcal{B}$ and using e is at most $1/\varepsilon'$. It follows that $|S^{up}| \leq \frac{1}{\varepsilon'} \cdot O_{\varepsilon,\delta}(1) = O_{\varepsilon,\delta,\varepsilon'}(1)$, hence there are at most $O(n^{O_{\varepsilon,\delta,\varepsilon'}(1)})$ choices for S^{up}. Finally $\{S_B^{up}\}_{B \in \mathcal{B}}$ is an assignment of up to $O_{\varepsilon,\delta,\varepsilon'}(1)$ tasks into up to $O_{\varepsilon,\delta}(1)$ boxes, and there are at most $O_{\varepsilon,\delta,\varepsilon'}(1)$ such possible assignments. Hence the possible number of states is $n^{O_{\varepsilon,\delta,\varepsilon'}(1)}$.

Let us define $\text{play}(v)$ as the maximum expected profit that any player can make from state v on. We can easily compute these values in polynomial time in the number of states in the game DAG as follows. We set $\text{play}(s) = 0$ for the sink node

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

s. Now consider nodes v in any reverse topological order. Let $\mathcal{D}_v^1$ be the possible decisions D at node v leading to a unique next state $next_v(D)$. Let also $\mathcal{D}_v^2$ be the remaining decisions D at node v leading to two possible next states $next_v^{up}(D)$ with probability p_{up} and $next_v^{down}(D)$ with probability $1-p_{up}$. We also set $w_v(D)=0$ if the task i associated with state v is discarded by D, and $w_v(D)=w(i)$ otherwise. We obtain

$$\begin{aligned}\text{play}(v) = \max\{ &\max_{D\in\mathcal{D}_v^1}\{w_v(D)+\text{play}(next_v(D))\},\\ &\max_{D\in\mathcal{D}_v^2}\{w_v(D)+p_{up}\cdot\text{play}(next_v^{up}(D))+(1-p_{up})\cdot\text{play}(next_v^{down}(D))\}.\end{aligned}$$

We let $D^*(v)$ be the the decision leading to the maximum above for each node v, breaking ties arbitrarily. It follows that the value $\text{play}(r)$ at the root node is the maximum expected profit, and the optimal strategy is described by $D^*(\cdot)$.

A.4 Proof of Lemma 7 We need the following technical Lemma.

LEMMA 17. *Let T' be a set of tasks such that $d(T'\cap T_e)\le u(e)$ for every edge e and given edge capacities $u(e)$ and such that, for each $i\in T'$ and for a given constant $\varepsilon>0$, $d(i)\le\varepsilon^2\min_{e\in P(i)}\{u(e)\}$. Then in polynomial time one can compute a subset $T''\subseteq T'$ such that $w(T'')\ge(1-O(\varepsilon))w(T')$ and $d(T''\cap T_e)\le(1-\varepsilon)u(e)$ on every edge e.*

Proof. We consider T' as a integral solution $\{x_i\}_{i\in T'}$ for the standard UFP LP. We next multiply the variables by a factor $(1-\varepsilon)$, so that this fractional solution has demand at most $(1-\varepsilon)u(e)$ on each edge e. The claim follows by the LP-rounding technique for ε^2-small tasks [CMS07]. □

It is sufficient to describe a valid strategy of the player with large enough profit. Consider the optimal boxed solution $(\text{OPT}_L,\{\text{OPT}_{S,B}\}_{B\in\mathcal{B}^*})$ due to Lemma 4. Using Lemma 17, for each $B\in\mathcal{B}^*$ we determine a set $\text{OPT}'_{S,B}\subseteq\text{OPT}_{S,B}$ such that $w(\text{OPT}'_{S,B})\ge(1-O(\varepsilon))w(\text{OPT}_{S,B})$ and on each edge $e\in P(B)$ the tasks in $\text{OPT}'_{S,B}\cap T_e$ have a total demand of at most $(1-\varepsilon)d(B)=(1-\varepsilon)b$. We next run the player with the following strategy. Tasks in OPT_L are always selected and each box in $\mathcal{B}^*$ is created. For a small task $i\in\text{OPT}'_{S,B}$, the player accepts i and assigns it to B whenever she can. In particular, if e is the current edge and $S_B^{up}\subseteq\text{OPT}'_{S,B}$ is the current set of selected rounded up tasks assigned to B and using edge e, then i is accepted iff $|S_B^{up}|<\frac{1}{\varepsilon'}$. Observe that this player satisfies all the constraints.

In order to prove the claim it is sufficient to show that, for each B, each $i\in\text{OPT}'_{S,B}$ is selected with probability at least $1-\varepsilon$. With the above notation, in order for i to be discarded it must necessarily happen that $|S_B^{up}|\ge\frac{1}{\varepsilon'}$. Observe that $|\text{OPT}'_{S,B}|\le(1-\varepsilon)b$ by construction, thus $E[|S_B^{up}|]\le(1-\varepsilon)b\cdot p_{up}=\frac{1-\varepsilon}{\varepsilon'}$. Therefore by Chernoff's bound:

$$Pr[|S_{B,t}^{up}|\ge\frac{1}{\varepsilon'}]\le e^{-\frac{1-\varepsilon}{3\varepsilon'}(\frac{1}{1-\varepsilon}-1)^2}=e^{-\frac{\varepsilon^2}{3\varepsilon'(1-\varepsilon)}}\le\varepsilon.$$

A.5 Proof of Lemma 9 First of all, in polynomial time we can compute the probability $p(v)$ that each state v in the strategy DAG is reached. Recall that each edge e in the DAG is labelled with a probability $p(e)\in\{1,p_{up},1-p_{up}\}$. Obviously $p(r)=1$ for the root state r. For any other state v, following any topological order in the DAG, we set $p(v)=\sum_{e=(u,v)}p(e)\cdot p(u)$.

For each task i let V_i denote the states of the strategy DAG in which i is selected. Observe that in any root-to-sink path P of the strategy DAG contains at most one node in V_i. Hence, i is selected at most once. In particular the states V_i are not descendants of each other. Hence the probability that any such state is reached is the sum of the corresponding probabilities, which implies $x_i=\sum_{v\in V_i}p(v)$.

Let $V_{v,i}$ be the descendant states of v (v included) such that task i is selected and assigned to box $B(v)$ in those states. By the same argument as above, the probability that i is assigned to box $B(v)$ in any one of such states is the sum $y_{v,i}=\sum_{w\in V_{v,i}}p(w)$. Since we are conditioning on the event that state v is reached, we can conclude that $x_{v,i}=y_{v,i}/p(v)$ (notice that $p(v)>0$ since v is reachable from the root).

A.6 Proof of Lemma 10 For each task i we have that x_i is the probability that the player selects i and then gains the profit $w(i)$ due to i. Therefore, we have that $\sum_{i\in T_S}w(i)x_i=\text{play}_S$. Next, we show that the variables $\{x_i\}_{i\in T_S}$ form a feasible fractional solution under $(1+\delta)$ resource augmentation. Recall call that $d(i)=1$ for every small task i.

LEMMA 18. *For each edge e it holds that $\sum_{i\in T_e\cap T_S}x_i\le(1+\delta)u(e)$.*

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

Proof. Let e be an edge. Recall that $b := \delta^4 u_{\min}$ is the (uniform) size of our boxes. For a small task i, let $\hat{x}_i := x_i \cdot \frac{1}{\varepsilon' b}$ and observe that $\hat{x}_i$ denotes the probability that i is selected *and* rounded up. When the player plays the game, then the total number of tasks in $T_e \cap T_S$ that are selected and rounded up is bounded by $\frac{(1+\delta)u(e)}{\varepsilon' b}$, since otherwise the rounded demand of the rounded up tasks would exceed the increased capacity $(1+\delta)u(e)$. This implies that $\sum_{i \in T_e \cap T_S} \hat{x}_i \leq \frac{(1+\delta)u(e)}{\varepsilon' b}$ (since otherwise with some positive probability more than $\frac{(1+\delta)u(e)}{\varepsilon' b}$ tasks from $T_e \cap T_S$ would be rounded up). Therefore we have that $\sum_{i \in T_e \cap T_S} x_i = \sum_{i \in T_e \cap T_S} \varepsilon' b \cdot \hat{x}_i \leq (1+\delta)u(e)$. ∎

Lemma 18 implies that the variables $\{x_i\}_{i \in T_S}$ form a feasible fractional solution to the the canonical LP for UFP with $1+\delta$ resource augmentation and small tasks only, i.e.,

$$\begin{aligned} \max & \sum_{i \in T_S} w(i) y_i & \\ \text{s.t.} & \sum_{i \in T_e \cap T_S} y_i \leq (1+\delta)u(e) & \forall e \in E \\ & y_i \leq 1 & \forall i \in T_S \\ & y_i \geq 0 & \forall i \in T_S \end{aligned}$$

The constraint matrix of this LP is totally unimodular. Therefore, in polynomial time we can compute an integral solution with profit at least $\sum_{i \in T_S} w(i) x_i = \text{play}_S$.

A.7 Proof of Lemma 11 First, we show that for each state v the variables $x_{v,i}$ define a feasible fractional solution for box $B(v)$. Recall that we assume $d(i) = 1$ for small tasks.

LEMMA 19. *Let v be a state where $B = B(v)$ is defined. For each edge $e \in P(B)$ it holds that $\sum_{i \in T_e \cap T_S} x_{v,i} \leq d(B)$. Therefore, for each $v \in P^*$ and each edge $e \in P(B)$ it holds that $\sum_{i \in T_e \cap T_S} X'_{v,i} \leq \sum_{i \in T_e \cap T_S} x_{v,i} \leq d(B)$.*

Proof. For the first claim, we argue similarly as in the proof of Lemma 10. For each small task i let $\hat{x}_{v,i} := x_{v,i} \cdot \frac{1}{\varepsilon' b}$ and observe that $\hat{x}_{v,i}$ denotes the probability that after state v the task i is selected, assigned to the box B *and* rounded up. When the player plays the game, then the total number of tasks in $T_e \cap T_S$ that are selected, assigned to B, and rounded up is upper bounded deterministically by $\frac{1}{\varepsilon'}$, since otherwise the rounded demand of the rounded up tasks would exceed $d(B)$. This implies that $\sum_{i \in T_e \cap T_S} \hat{x}_{v,i} \leq \frac{1}{\varepsilon'}$. Therefore, we have that $\sum_{i \in T_e \cap T_S} x_{v,i} = \sum_{i \in T_e \cap T_S} \varepsilon' b \cdot \hat{x}_{v,i} \leq d(B)$. From the definition of the variables $X'_{v,i}$ we have that $X'_{v,i} \leq x_{v,i}$ for each state v and each small task i which implies the second claim. ∎

Lemma 19 implies that for each edge e we have that

$$\begin{aligned} \sum_{i \in T_e \cap T_S} X'_i &= \sum_{v \in P^*} \sum_{i \in T_e \cap T_S} X'_{v,i} \\ &\leq \sum_{v \in P^*} \sum_{i \in T_e \cap T_S} x_{v,i} \\ &\leq \sum_{v \in P^* : e \in P(B(v))} d(B(v)) =: \hat{u}(e). \end{aligned}$$

Hence, the variables $\{X'_i\}_{i \in T_S}$ define a feasible fractional solution to the following LP:

$$\begin{aligned} \max & \sum_{i \in T_S} w(i) y_i & \\ \text{s.t.} & \sum_{i \in T_e \cap T_S} y_i \leq \hat{u}(e) & \forall e \in E \\ & y_i \leq 1 & \forall i \in T_S \\ & y_i \geq 0 & \forall i \in T_S \end{aligned}$$

The constraint matrix of this LP is totally unimodular. Therefore, in polynomial time we can compute an integral solution with profit at least $\sum_{i \in T_S} X'_i$. The solution claimed in Lemma 11 consists of the tasks in this integral solution and additionally the large tasks selected by the player.

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

A.8 Proof of Lemma 13 Recall that probabilities p on the edges induces a probability distribution $p(\tilde{Q})$ over the paths $\tilde{Q} \in \tilde{\mathcal{Q}}$. Symmetrically, a probability distribution $\left\{p(\tilde{Q})\right\}_{\tilde{Q}\in\tilde{\mathcal{Q}}}$ induces probabilities $p(e)$ on the edges (modulo edges that are contained in $\tilde{Q}^*$ with probability zero, whose value is irrelevant for our goals). We will therefore use p interchangeably to denote the probabilities on the edges or the corresponding probability distribution over $\tilde{\mathcal{Q}}$.

Our plan is as follows. We start with worst-case probabilities p^* and gradually transform them while maintaining the fact that they are worst-case. At the end of the process, p^* will satisfy the claim. Sometimes it will more convenient for us to update p^* indirectly by updating the corresponding probability distribution over $\tilde{\mathcal{Q}}$.

We next focus on a specific task i and sometimes drop the pedex i in the variables to lighten the notation. For example, we will use Y' instead of Y'_i etc. Recall that $Y' := \min\left\{\sum_{\tilde{v}\in\tilde{Q}^*} y_{\tilde{v}}, 1\right\}$. One way of interpreting Y' is as follows. Assume that $\tilde{Q}^* = \tilde{v}_1, \tilde{v}_2, ..., \tilde{v}_k$ with $\tilde{r} = \tilde{v}_1$ and $\tilde{s} = \tilde{v}_k$. When we define the variable Y', we traverse $\tilde{Q}^*$ from $\tilde{r}$ to $\tilde{s}$. On each node $\tilde{v}_k$ we collect a value of $y_{\tilde{v}_k}$ if $\sum_{\ell=1}^{k} y_{\tilde{v}_\ell} \leq 1$. Then at some point we reach a node $\tilde{v}_{k^*}$ such that $\sum_{\ell=1}^{k^*} y_{\tilde{v}_\ell} > 1$. On this node $\tilde{v}_{k^*}$ we collect a value of $1 - \sum_{\ell=1}^{k^*-1} y_{\tilde{v}_\ell}$. Afterwards, on each node $\tilde{v}_\ell$ with $\ell > k^*$ we collect a value of 0. Then, Y' is the sum of these collected values. Now we observe that the value collected by Y' on a node $\tilde{v}$ is always the same, i.e., it is the same for each path $\tilde{Q}^* \in \tilde{\mathcal{Q}}$ with $\tilde{v} \in \tilde{Q}^*$. We denote this value by $y'_{\tilde{v}}$. Formally, we define

$$y'_{\tilde{v}} := \begin{cases} y_{\tilde{v}} & \text{if } \sum_{\tilde{v}'\in T(\tilde{v})} y_{\tilde{v}'} \leq 1 \\ 1 - \sum_{\tilde{v}'\in T(\tilde{v})} y_{\tilde{v}'} & \text{if } \sum_{\tilde{v}'\in T(\tilde{v})} y_{\tilde{v}'} > 1 \text{ and } \sum_{\tilde{v}'\in T(\tilde{v})\setminus\{\tilde{v}\}} y_{\tilde{v}'} \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

It is convenient to interpret the quantities $y_{\tilde{v}}$ and $y'_{\tilde{v}}$ as functions $y_{\tilde{v}}(p)$ and $y'_{\tilde{v}}(p)$ of the considered probabilities p.

We prove a technical lemma which we will invoke several times.

LEMMA 20. *Let p be a distribution over $\tilde{\mathcal{Q}}$. Let $\tilde{Q} \in \tilde{\mathcal{Q}}$, $\tilde{v} \in \tilde{Q}$, and let $\tilde{v}'$ be a leaf that is a child of $\tilde{v}$. Assume that either $\tilde{Q}$ ends in a $B(\tilde{v})$-success node and $\tilde{v}'$ is a $B(\tilde{v})$-success node, or that $\tilde{Q}$ ends in a fail node and $\tilde{v}'$ is a fail node. Define a new distribution q over $\tilde{\mathcal{Q}}$ by*

- $q(\tilde{Q}) = 0$,
- $q(T(\tilde{v}')) = p(T(\tilde{v}')) + p(\tilde{Q})$, *and*
- $q(\tilde{P}) = p(\tilde{P})$ for each path $\tilde{P} \in \tilde{\mathcal{Q}} \setminus \{\tilde{Q}, T(\tilde{v}')\}$.

Then we have that $\mathbb{E}[Y(q)] = \mathbb{E}[Y(p)]$ and $\mathbb{E}[Y'(q)] \leq \mathbb{E}[Y'(p)]$.

Proof. It is clear that $\mathbb{E}[Y(q)] = \mathbb{E}[Y(p)]$ because either both $\tilde{Q}$ and $T(\tilde{v}')$ end in a success node, or both end in a fail node. We want to prove now that $\mathbb{E}[Y'(q)] \leq \mathbb{E}[Y'(p)]$. We observe that $y_{\tilde{v}}(p) = y_{\tilde{v}}(q)$. Let $\tilde{v}_1, ..., \tilde{v}_\ell$ be the nodes contained in $\tilde{Q}$ that $\tilde{Q}$ visits after $\tilde{v}$ and which create a box (when starting from the root, excluding $\tilde{v}$).

Assume first that there is a value $j \in \{1, ..., \ell\}$ such that $y'_{\tilde{v}_j}(p) < y_{\tilde{v}_j}(p)$. In this case we define an intermediate distribution z as follows. Let ℓ^* be the largest value $j \in \{1, ..., \ell\}$ such that $y'_{\tilde{v}_j}(p) = y_{\tilde{v}_j}(p)$. For our argumentation, we temporarily introduce a child $\tilde{v}^{child}_{\ell^*}$ of $\tilde{v}_{\ell^*}$ and we define that $\tilde{v}^{child}_{\ell^*}$ is a $B(\tilde{v})$-success node if $\tilde{Q}$ ends in a $B(\tilde{v})$-success node and that $\tilde{v}^{child}_{\ell^*}$ is a fail-node if $\tilde{Q}$ ends in a fail-node. Let $\tilde{Q}_{\ell^*} := T(\tilde{v}^{child}_{\ell^*})$. We define the intermediate distribution z in which intuitively the probability mass of $\tilde{Q}$ is moved to $\tilde{Q}_{\ell^*}$:

- $z(\tilde{Q}) = 0$,
- $z(\tilde{Q}_{\ell^*}) = p(\tilde{Q})$, and
- $z(\tilde{P}) = p(\tilde{P})$ for each path $\tilde{P} \in \tilde{\mathcal{Q}} \setminus \{\tilde{Q}, \tilde{Q}_{\ell^*}\}$.

Like above, it holds that $\mathbb{E}[Y(z)] = \mathbb{E}[Y(p)]$. For each $j > \ell^*$, it holds that $y_{\tilde{v}_j}(z) \geq y_{\tilde{v}_j}(p)$ since $p(\tilde{Q}) > 0$ and if $\tilde{Q} = \tilde{Q}^*$ then $\tilde{v}_j \in \tilde{Q}^*$ but $\tilde{Q}$ does not end in a $B(\tilde{v}_j)$-success node; however, $z(\tilde{Q}) = 0$. Since $j > \ell^*$ we have that $y'_{\tilde{v}_j}(p) < y_{\tilde{v}_j}(p)$ and therefore $y'_{\tilde{v}_j}(z) = y'_{\tilde{v}_j}(p)$. The reason is that if $\tilde{v}_j \in \tilde{Q}$, the minimum in the definition of $Y'_{\tilde{v}_j}(z)$ is *not* attained in $y_{\tilde{v}_j}(z)$. Observe that for each $j \geq \ell^* + 2$, both $y'_{\tilde{v}_j}(z)$ and $y'_{\tilde{v}_j}(p)$ are zero.

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

Also, for each $j \leq \ell^*$ it holds that $y_{\tilde{v}_j}(z) = y_{\tilde{v}_j}(p)$. Therefore, it holds that

$$\begin{aligned}
\mathbb{E}[Y'(z)] &= \sum_{\tilde{P}\in\tilde{\mathcal{Q}}} z(\tilde{P}) \cdot \sum_{\tilde{w}\in\tilde{P}} y'_{\tilde{w}}(z) \\
&= \sum_{\tilde{P}\in\tilde{\mathcal{Q}}\setminus\{\tilde{Q}_{\ell^*}\}} z(\tilde{P}) \cdot \sum_{\tilde{w}\in\tilde{P}} y'_{\tilde{w}}(z) + z(\tilde{Q}_{\ell^*}) \cdot \sum_{\tilde{w}\in\tilde{Q}_{\ell^*}} y'_{\tilde{w}}(z) \\
&= \sum_{\tilde{P}\in\tilde{\mathcal{Q}}\setminus\{\tilde{Q}\}} p(\tilde{P}) \cdot \sum_{\tilde{w}\in\tilde{P}} y'_{\tilde{w}}(p) + z(\tilde{Q}_{\ell^*}) \cdot \sum_{\tilde{w}\in\tilde{Q}_{\ell^*}} y'_{\tilde{w}}(z) \\
&\leq \sum_{\tilde{P}\in\tilde{\mathcal{Q}}\setminus\{\tilde{Q}\}} p(\tilde{P}) \cdot \sum_{\tilde{w}\in\tilde{P}} y'_{\tilde{w}}(p) + p(\tilde{Q}) \cdot \sum_{\tilde{w}\in\tilde{Q}} y'_{\tilde{w}}(p). \\
&= \mathbb{E}[Y'(p)]
\end{aligned}$$

where the third equality and the inequality follow from the discussion before the calculation and the fact that the paths in $\tilde{\mathcal{Q}}\setminus\{\tilde{Q}_{\ell^*}, \tilde{Q}\}$ have the same probabilities in p and z, $z(\tilde{Q}) = p(\tilde{Q}_{\ell^*}) = 0$, and $z(\tilde{Q}_{\ell^*}) = p(\tilde{Q})$. This completes our treatment for the case that initially there is a value $j \in \{1, ..., \ell\}$ such that $y'_{\tilde{v}_j}(p) < y_{\tilde{v}_j}(p)$. In case that initially there is no such value $j \in \{1, ..., \ell\}$, we simply define $z := p$ and $\tilde{Q}_{\ell^*} := \tilde{Q}$.

Now observe that we obtain q from z by decreasing $z(\tilde{Q}_{\ell^*})$ to 0 and increasing $z(T(\tilde{v}'))$ by $z(\tilde{Q}_{\ell^*}) = p(\tilde{Q})$. To this end, for each $j \in \{1, ..., \ell\}$ let $\tilde{\mathcal{Q}}(\tilde{v}_j)$ denote the paths in $\tilde{\mathcal{Q}}$ that contain $\tilde{v}_j$ and let $\tilde{\mathcal{Q}}^{succ}(\tilde{v}_j) \subseteq \tilde{\mathcal{Q}}(\tilde{v}_j)$ denote the paths in $\tilde{\mathcal{Q}}(\tilde{v}_j)$ which contain a $B(\tilde{v}_j)$-success node. We observe that $y_{\tilde{v}_j}(z) = \frac{\sum_{\tilde{P}\in\tilde{\mathcal{Q}}^{succ}(\tilde{v}_j)} z(\tilde{P})}{\sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{v}_j)} z(\tilde{P})}$ and $y_{\tilde{v}_j}(q) = \frac{\sum_{\tilde{P}\in\tilde{\mathcal{Q}}^{succ}(\tilde{v}_j)} q(\tilde{P})}{\sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{v}_j)} q(\tilde{P})}$. Also, we have that $\sum_{\tilde{P}\in\tilde{\mathcal{Q}}^{succ}(\tilde{v}_j)} z(\tilde{P}) = \sum_{\tilde{P}\in\tilde{\mathcal{Q}}^{succ}(\tilde{v}_j)} q(\tilde{P})$ and $\sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{v}_j)} z(\tilde{P}) \leq \sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{v}_j)} q(\tilde{P})$.

We have that $y_{\tilde{v}_j}(q) \geq y_{\tilde{v}_j}(z)$ for each $j \in \{1, ..., \ell^*\}$ since $z(\tilde{Q}_{\ell^*}) > 0$ and if $\tilde{Q}_{\ell^*} = \tilde{Q}^*$ then $\tilde{v}_j \in \tilde{Q}^*$ but $\tilde{Q}^*$ does not end in a $B(\tilde{v}_j)$-success node; however, $q(\tilde{Q}) = 0$. Now we observe that $y_{\tilde{v}_j}(q) - y'_{\tilde{v}_j}(q) \geq 0 = y_{\tilde{v}_j}(z) - y'_{\tilde{v}_j}(z)$ for each $j \in \{1, ..., \ell^*\}$ which implies

$$y'_{\tilde{v}_j}(q) - y'_{\tilde{v}_j}(z) \leq y_{\tilde{v}_j}(q) - y_{\tilde{v}_j}(z). \tag{A.1}$$

We have that

$$\begin{aligned}
&\mathbb{E}[Y'(q)] - \mathbb{E}[Y'(z)] \\
&= \sum_{\tilde{P}\in\tilde{\mathcal{Q}}} q(\tilde{P}) \sum_{\tilde{w}\in\tilde{P}} y'_{\tilde{w}}(q) - \sum_{\tilde{P}\in\tilde{\mathcal{Q}}} z(\tilde{P}) \sum_{\tilde{w}\in\tilde{P}} y'_{\tilde{w}}(z) \\
&= \sum_{\tilde{w}\in T} \sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{w})} q(\tilde{P}) \cdot y'_{\tilde{w}}(q) - \sum_{\tilde{w}\in T} \sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{w})} z(\tilde{P}) \cdot y'_{\tilde{w}}(z) \\
&= \sum_{j=1}^{\ell^*} \left(\sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{v}_j)} \left(q(\tilde{P}) \cdot y'_{\tilde{v}_j}(q) - z(\tilde{P}) \cdot y'_{\tilde{v}_j}(z) \right) \right) \\
&= \sum_{j=1}^{\ell^*} \left(\sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{v}_j)\setminus\{\tilde{Q}_{\ell^*}\}} \left(q(\tilde{P}) \cdot y'_{\tilde{v}_j}(q) - z(\tilde{P}) \cdot y'_{\tilde{v}_j}(z) \right) + \underbrace{q(\tilde{Q}_{\ell^*})}_{=0} \cdot y'_{\tilde{v}_j}(q) - z(\tilde{Q}_{\ell^*}) \cdot y'_{\tilde{v}_j}(z) \right) \\
&= \sum_{j=1}^{\ell^*} \left(\sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{v}_j)\setminus\{\tilde{Q}_{\ell^*}\}} q(\tilde{P}) \left(y'_{\tilde{v}_j}(q) - y'_{\tilde{v}_j}(z) \right) - z(\tilde{Q}_{\ell^*}) \cdot y'_{\tilde{v}_j}(z) \right) \\
&\leq \sum_{j=1}^{\ell^*} \left(\sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{v}_j)\setminus\{\tilde{Q}_{\ell^*}\}} q(\tilde{P}) \left(y_{\tilde{v}_j}(q) - y_{\tilde{v}_j}(z) \right) - z(\tilde{Q}_{\ell^*}) \cdot y_{\tilde{v}_j}(z) \right) \\
&= \sum_{j=1}^{\ell^*} \left(y_{\tilde{v}_j}(q) \left(\sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{v}_j)\setminus\{\tilde{Q}_{\ell^*}\}} q(\tilde{P}) \right) - y_{\tilde{v}_j}(z) \left(z(\tilde{Q}_{\ell^*}) + \sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{v}_j)\setminus\{\tilde{Q}_{\ell^*}\}} z(\tilde{P}) \right) \right)
\end{aligned}$$

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

$$
\begin{aligned}
&= \sum_{j=1}^{\ell^*} \left(\frac{\sum_{\tilde{P}\in\tilde{\mathcal{Q}}^{succ}(\tilde{v}_j)} q(\tilde{P})}{\sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{v}_j)} q(\tilde{P})} \left(\sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{v}_j)\setminus\{\tilde{Q}_{\ell^*}\}} q(\tilde{P}) \right) - \frac{\sum_{\tilde{P}\in\tilde{\mathcal{Q}}^{succ}(\tilde{v}_j)} z(\tilde{P})}{\sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{v}_j)} z(\tilde{P})} \left(z(\tilde{Q}_{\ell^*}) + \sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{v}_j)\setminus\{\tilde{Q}_{\ell^*}\}} z(\tilde{P}) \right) \right) \\
&= \sum_{j=1}^{\ell^*} \left(\frac{\sum_{\tilde{P}\in\tilde{\mathcal{Q}}^{succ}(\tilde{v}_j)} q(\tilde{P})}{\sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{v}_j)} q(\tilde{P})} \left(\underbrace{q(\tilde{Q}_{\ell^*})}_{=0} + \sum_{\tilde{P}\in\tilde{\mathcal{Q}}(\tilde{v}_j)\setminus\{\tilde{Q}_{\ell^*}\}} q(\tilde{P}) \right) - \sum_{\tilde{P}\in\tilde{\mathcal{Q}}^{succ}(\tilde{v}_j)} z(\tilde{P}) \right) \\
&= \sum_{j=1}^{\ell^*} \left(\sum_{\tilde{P}\in\tilde{\mathcal{Q}}^{succ}(\tilde{v}_j)} q(\tilde{P}) - \sum_{\tilde{P}\in\tilde{\mathcal{Q}}^{succ}(\tilde{v}_j)} z(\tilde{P}) \right) \\
&= 0,
\end{aligned}
$$

where the inequality follows from (A.1) and $y'_{\tilde{v}_j}(z) = y'_{\tilde{v}_j}(p) = y_{\tilde{v}_j}(p) = y_{\tilde{v}_j}(z)$ for each $j \leq \ell^*$ and the last equality follows since for each j and each $\tilde{P} \in \tilde{\mathcal{Q}}^{succ}(\tilde{v}_j)$ it holds that $q(\tilde{P}) = z(\tilde{P})$ (i.e., when defining q based on z, we changed only the $B(\tilde{v})$-success path $\tilde{Q}_{\ell^*}$). This implies that $\mathbb{E}[Y'(q)] \leq \mathbb{E}[Y'(z)] \leq \mathbb{E}[Y'(p)]$. ∎

Given the above lemma, it is not hard to show that the first two properties of Lemma 13 hold.

LEMMA 21. *Property 1 of Lemma 13 holds.*

Proof. Let us introduce one global artificial fail node $\tilde{r}^{fail}$ and an arc $(\tilde{r}, \tilde{r}^{fail})$ where $\tilde{r}$ is the root of T. We let p^* and $\tilde{Q}^*$ refer to this new probability space. Clearly this can only worsen the worst-case ratio $\mathbb{E}[Y(p^*)]/\mathbb{E}[Y'(p^*)]$ since setting $p^*(\tilde{r}, \tilde{r}^{fail}) = 0$ is an option. We will later see that indeed the latter case happens, hence introducing node $\tilde{r}^{fail}$ is w.l.o.g.

Suppose that p^* are worst-case probabilities not satisfying the claim. We gradually turn p^* into probabilities q^* which are still worst-case, and such that the only fail node that can be reached with positive probability is $\tilde{r}^{fail}$. Initially $q^* = p^*$. While there exists a fail node $\tilde{w} \neq \tilde{r}^{fail}$ which is reached with positive probability according to q^*, we simply transform q^* as suggested by Lemma 20 with parameters $\tilde{Q} = T(\tilde{w})$, $\tilde{v} = \tilde{r}$ and $\tilde{v}' = \tilde{r}^{fail}$. Observe that q^* remains worst-case. At the end of the process q^* satisfies the claim.

After this transformation, let $q = q^*(\tilde{r}, \tilde{r}^{fail})$ be the probability that $\tilde{r}^{fail}$ is reached. We observe that $\mathbb{E}[Y(q^*)] = (1-q) \cdot \mathbb{E}[Y(q^*)|\tilde{r}^{fail} \notin \tilde{Q}^*]$ and $\mathbb{E}[Y'(q^*)] = (1-q) \cdot \mathbb{E}[Y'(p^{(1)})|\tilde{r}^{fail} \notin \tilde{Q}^*]$. We define new probabilities z^* such that $z^*(T(\tilde{r}^{fail})) = z^*(\tilde{r}, \tilde{r}^{fail}) = 0$ and $z^*(\tilde{Q}) = p^*(\tilde{Q})/(1-q)$ for any other path $\tilde{Q} \in \tilde{\mathcal{Q}}$. Observe that $\mathbb{E}[Y(q^*)]/\mathbb{E}[Y'(q^*)] = \mathbb{E}[Y(z^*)]/\mathbb{E}[Y'(z^*)]$, hence z^* are also worst-case probabilities. Probabilities z^* satisfy the claim. ∎

LEMMA 22. *Property 2 of Lemma 13 holds.*

Proof. Suppose p^* are worst-case probabilities not satisfying the claim. We gradually transform them into worst-case probabilities q^* satisfying this property. To that aim, let us initially set $q^* = p^*$. While there exists an internal node $\tilde{w}$ and a $B(\tilde{w})$-success node $\tilde{w}'$ distinct from $\tilde{w}^{succ}$ with $q^*(T(\tilde{w}')) > 0$, we transform q^* as suggested by Lemma 20 with parameters $\tilde{v} = \tilde{w}$, $\tilde{Q} = T(\tilde{w}')$, and $\tilde{v}' = \tilde{v}^{succ}$. Observe that q^* remains worst-case. At the end of the process q^* satisfies the claim. ∎

It remains to prove Property 3.

LEMMA 23. *Property 3 of Lemma 13 holds.*

Proof. Let p^* be worst-case probabilities not satisfying the claim. Consider any internal node $\tilde{v}$ with two internal children $\tilde{v}_1$ and $\tilde{v}_1$ such that $p_1 = p^*(\tilde{v}, \tilde{v}_1)$ and $p_2 = p^*(\tilde{v}, \tilde{v}_2)$ are both strictly positive. We next define alternative-worst case probabilities q^* which are identical to p^* excluding for the probabilities $q_1 = q^*(\tilde{v}, \tilde{v}_1)$ and $q_2 = q^*(\tilde{v}, \tilde{v}_2)$, where at least one among q_1 and q_2 is set to zero. By replacing p^* with q^*, and iterating the process, one obtains worst-case probabilities satisfying Property 3.

Notice that the feasibility of p^* and q^* enforce $q := q_1 + q_2 = p_1 + p_2$. We remark that by Property 1 it must be the case that $\mathbb{E}[Y(q^*)] = 1 = \mathbb{E}[Y(p^*)]$. We next choose q_1 and q_2 in order to minimize $\mathbb{E}[Y'(q^*)]$. Notice that $(q_1, q_2) = (p_1, p_2)$ is a feasible solution, hence q^* has to be worst-case as well. We will show that the maximum can be achieved by setting exactly one of q_1 and q_2 to zero, hence the claim.

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

To this aim, observe that

$$\begin{aligned}\mathbb{E}[Y'(q^*)] =& \Pr_{q^*}[\tilde{v}_1 \notin \tilde{Q}^* \wedge \tilde{v}_2 \notin \tilde{Q}^*] \cdot \mathbb{E}[Y'(q^*)|\tilde{v}_1 \notin \tilde{Q}^* \wedge \tilde{v}_2 \notin \tilde{Q}^*] \\ &+ \Pr_{q^*}[\tilde{v}_1 \in \tilde{Q}^*] \cdot \mathbb{E}[Y'(q^*)|\tilde{v}_1 \in \tilde{Q}^*] + \Pr_{q^*}[\tilde{v}_2 \in \tilde{Q}^*] \cdot \mathbb{E}[Y'(q^*)|\tilde{v}_2 \in \tilde{Q}^*] \\ =& \Pr_{q^*}[\tilde{v}_1 \notin \tilde{Q}^* \wedge \tilde{v}_2 \notin \tilde{Q}^*] \cdot \mathbb{E}[Y'(q^*)|\tilde{v}_1 \notin \tilde{Q}^* \wedge \tilde{v}_2 \notin \tilde{Q}^*] \\ &+ \Pr_{q^*}[\tilde{v} \in \tilde{Q}^*] \cdot q_1 \cdot \mathbb{E}[Y'(q^*)|\tilde{v}_1 \in \tilde{Q}^*] + \Pr_{q^*}[\tilde{v} \in \tilde{Q}^*] \cdot q_2 \cdot \mathbb{E}[Y'(q^*)|\tilde{v}_2 \in \tilde{Q}^*],\end{aligned}$$

where in the first equality we used the fact that the events $\{\tilde{v}_1 \notin \tilde{Q}^* \wedge \tilde{v}_2 \notin \tilde{Q}^*\}$, $\{\tilde{v}_1 \in \tilde{Q}^*\}$ and $\{\tilde{v}_2 \in \tilde{Q}^*\}$ partition the probability space. Indeed, $\tilde{v}_1 \in \tilde{Q}^*$ implies $\tilde{v}_2 \notin \tilde{Q}^*$ and vice versa since $\tilde{v}_1$ and $\tilde{v}_2$ are siblings in the tree T. We observe that by definition

$$a := \Pr_{q^*}[\tilde{v}_1 \notin \tilde{Q}^* \wedge \tilde{v}_2 \notin \tilde{Q}^*] = \Pr_{p^*}[\tilde{v}_1 \notin \tilde{Q}^* \wedge \tilde{v}_2 \notin \tilde{Q}^*]$$

and

$$b := \Pr_{q^*}[\tilde{v} \in \tilde{Q}^*] = \Pr_{p^*}[\tilde{v} \in \tilde{Q}^*].$$

Observe that, by Property 2 of Lemma 13, $y_{\tilde{v}}(q^*) = q^*(\tilde{v}, \tilde{v}^{succ}) = p^*(\tilde{v}, \tilde{v}^{succ}) = y_{\tilde{v}}(p^*)$. For all other nodes $\tilde{w}$ one has by definition $y_{\tilde{w}}(q^*) = y_{\tilde{w}}(p^*)$. This implies $y'_{\tilde{w}}(q^*) = y'_{\tilde{w}}(p^*)$ for all nodes $\tilde{w}$. As a consequence, the quantities $\mathbb{E}[Y'(q^*)|\tilde{v}_1 \in \tilde{Q}^*]$, $\mathbb{E}[Y'(q^*)|\tilde{v}_2 \in \tilde{Q}^*]$, and $\mathbb{E}[Y'(q^*)|\tilde{v}_1 \notin \tilde{Q}^* \wedge \tilde{v}_2 \notin \tilde{Q}^*]$ are independent from q_1 and q_2. Let us set:

$$c := \mathbb{E}[Y'(q^*)|\tilde{v}_1 \notin \tilde{Q}^* \wedge \tilde{v}_2 \notin \tilde{Q}^*] = \mathbb{E}[Y'(p^*)|\tilde{v}_1 \notin \tilde{Q}^* \wedge \tilde{v}_2 \notin \tilde{Q}^*]$$

and

$$d_j := \mathbb{E}[Y'(q^*)|\tilde{v}_j \in \tilde{Q}^*] = \mathbb{E}[Y'(p^*)|\tilde{v}_j \in \tilde{Q}^*], \quad j \in \{1, 2\}.$$

Altogether we have to minimize the function

$$ac + bd_1q_1 + bd_2q_2,$$

where a, b, c, d_1 and d_2 are constants (independent from q_1 and q_2) and $q_1 + q_2 = q$ for some constant q. This is obviously a linear function of q_1 in the domain $[0, q]$, hence the minimum is achieved either for $q_1 = 0$ or for $q_1 = q$ (which would imply $q_2 = 0$). ∎

A.9 Proof of Lemma 14 Also in this case we focus on a given task i, and drop the pedex i. Consider the worst-case probabilities p^* as given by Lemma 13. We next neglect the nodes of T which are reached (i.e., are contained in $\tilde{Q}^*$) with probability 0 since they do not contribute to the values of $\mathbb{E}[Y(p^*)]$ nor of $\mathbb{E}[Y'(p^*)]$. Recall that the internal nodes induce a path $\tilde{r} = \tilde{v}_1, \ldots, \tilde{v}_\ell$ for some integer ℓ, and each $\tilde{v}_j$ has one child $\tilde{v}_j^{succ}$. Furthermore, $p^*(\tilde{v}_\ell, \tilde{v}_\ell^{succ}) = 1$ and $\mathbb{E}[Y(p^*)] = 1$ by Property 1 of Lemma 13. Let $y_j = y_{\tilde{v}_j}$ and $y'_j = y'_{\tilde{v}_j}$ for each j. Notice that $y_j = p^*(\tilde{v}_j, \tilde{v}_j^{succ})$ by Properties 1 and 2 of Lemma 13. The quantity that we wish to minimize is

$$\mathbb{E}[Y'(p^*)] = \sum_{j=1}^{\ell} y'_j \prod_{k=1}^{j-1} p^*(\tilde{v}_k, \tilde{v}_{k+1}) = \sum_{j=1}^{\ell} y'_j \prod_{k=1}^{j-1} (1 - y_j).$$

Observe that for each j it holds that (a) $y'_j = y_j$ or (b) $y'_j = 1 - \sum_{k<j} y'_k$. If case (a) always holds, then the claim follows since

$$\mathbb{E}[Y'(p^*)] = \sum_{j=1}^{\ell} y_j \prod_{k=1}^{j-1} (1 - y_j) = \mathbb{E}[Y(p^*)].$$

Therefore we can assume the converse, and let b the smallest index such that $y'_b = 1 - \sum_{k<b} y'_k$. We call $\tilde{v}_b$ a *boundary* node. Observe that $y'_j = 0$ for $j > b$, hence the corresponding nodes contribute nothing to $\mathbb{E}[Y'(p^*)]$.

LEMMA 24. *If there exists a boundary node $\tilde{v}_b$, w.l.o.g. $p^*(\tilde{v}_b, \tilde{v}_b^{succ}) = 1$.*

Proof. Suppose p^* does not satisfy the claim. We define alternative edge probabilities q^* which are also worst-case and satisfy the claim. In more detail, q^* is identical to p^* with the exception $q^*(\tilde{v}_b, \tilde{v}_b^{succ}) = 1$ and $q^*(\tilde{v}_b, \tilde{v}_{b+1}) = 0$. Notice that $\mathbb{E}[Y(q^*)] = \mathbb{E}[Y(p^*)] = 1$. For the boundary node $\tilde{v}_b$ we have that $y_{\tilde{v}_b}(q^*) = 1 \geq y_{\tilde{v}_q}(p^*)$ which implies that $y'_{\tilde{v}_b}(q^*) = y'_{\tilde{v}_b}(p^*)$. For each non-boundary node $\tilde{v}_j$ we have that $y'_{\tilde{v}_j}(q^*) = y'_{\tilde{v}_j}(p^*)$. This implies that $\mathbb{E}[Y'(q^*)] = \mathbb{E}[Y'(p^*)]$, hence the claim. ∎

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

In particular the above lemma implies that $b = \ell$ since we are neglecting the parts of T which are reached with probability 0. Under these constraints we can rewrite $\mathbb{E}[Y'(p^*)]$ more conveniently as follows. Let $x_j = y_j = y'_j$ for $j < \ell$ and define $x_\ell = 1 - \sum_{j<\ell} x_j = y'_\ell$. Then

$$\mathbb{E}[Y'(p^*)] = \sum_{j=1}^{\ell} x_j \prod_{k=1}^{j-1}(1 - x_j).$$

It is therefore sufficient to minimize the above quantity under the constraint $\sum_{j=1}^{\ell} x_j = 1$ and $x_j > 0$ for all j. This is done in the following technical lemma.

LEMMA 25. *Let $x_1, \ldots, x_\ell$ be positive real numbers with sum $q \in (0, 1]$. Then $\sum_{j=1}^{\ell} x_j \prod_{k=1}^{j-1}(1 - x_j)$ is minimized for $x_j = q/\ell$ for all j. In particular, the value of the minimum is $\sum_{j=1}^{\ell} \frac{q}{\ell}(1 - \frac{q}{\ell})^{j-1} = 1 - (1 - \frac{q}{\ell})^{\ell}$.*

Proof. We prove the claim by induction on ℓ. The claim trivially holds for $\ell = 1$. Next assume it holds up to $\ell - 1 \geq 1$, and consider the value ℓ. Define $q' = \sum_{j=2}^{\ell} x_j = q - x_1$. Then

$$\begin{aligned}
&\sum_{j=1}^{\ell} x_j \prod_{k=1}^{j-1}(1 - x_j) = x_1 + (1 - x_1) \sum_{j=2}^{\ell} x_j \prod_{k=2}^{j-1}(1 - x_j) \\
\leq &x_1 + (1 - x_1)(1 - (1 - \frac{q'}{\ell - 1})^{\ell-1}) = 1 - (1 - x_1)(1 - \frac{q - x_1}{\ell - 1})^{\ell-1} =: f(x_1, q, \ell),
\end{aligned}$$

where in the inequality we applied the inductive hypothesis on $x_2, \ldots, x_\ell$. The derivative in x_1 of $f(x_1, q, \ell)$ is $(1 - \frac{q-x_1}{\ell-1})^{\ell-1}(x_1 \frac{\ell}{\ell-1} - \frac{q}{\ell-1})$. This shows that $f(x_1, q, \ell)$ has a unique minimum for $x_1 = q/\ell$. Notice that this implies $x_j = \frac{q'}{\ell-1} = \frac{q}{\ell}$ for $j \geq 2$ as well. ∎

By the above lemma it follows that

$$\mathbb{E}[Y'(p^*)] \leq 1 - (1 - \frac{1}{\ell})^{\ell} > 1 - \frac{1}{e},$$

hence concluding the proof of Lemma 14.

B A $1 + \frac{1}{1+e} + O(\varepsilon)$ Approximation

In this section we improve the approximation ratio to $1 + \frac{1}{1+e} + O(\varepsilon)$. In Section 3.2 we described two procedures to construct solutions based on the optimal strategy of the player. Here we introduce a third one; the best one of the resulting three solutions gives the claimed approximation ratio.

Again, we simulate the player, i.e., we select a path $Q^* \in \mathcal{Q}$ randomly according to the probabilities $\{p(Q)\}_{Q \in \mathcal{Q}}$. Like in the construction of X'_i, based on the random path Q^* we define random variables $\bar{X}'_i$ as follows:

- For each large task i, we let $\bar{X}_i = 1$ if the player selects i and $\bar{X}_i = 0$ otherwise.
- We create each box that the player creates.
- We traverse the states of Q^* starting with the root of the strategy DAG. If in some state on Q^* the player selects a small task i and assigns i into a box $B = B(v)$ created at some (potentially different) state $v \in Q^*$, then we select i and assign i into B if for each $e \in P(B)$ it holds that $d(i) + \sum_{i' \in T_e \cap T_S} \bar{X}_{v,i'} \leq (1 + \varepsilon)d(B)$ (where for notational convenience we assume that $\bar{X}_{v,i'} = 0$ if $\bar{X}_{v,i'}$ has not been defined before). In this case we set $\bar{X}_{v,i} = 1$, otherwise $\bar{X}_{v,i} = 0$.
- Finally, we set $\bar{X}_i = \sum_{v \in Q^*} \bar{X}_{v,i}$ for each small task i.

We define $\overline{\text{play}}_S = \sum_{i \in T_S} w_i E[\bar{X}_i]$.

LEMMA 26. *There is a polynomial-time randomized algorithm that computes a feasible boxed solution of expected profit at least* $\text{play}_L + (1 - O(\varepsilon))\overline{\text{play}}_S$.

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

Proof. We simulate the player and compute the solution $\{\bar{X}_i\}_i$ as described above. The excepted profit of this solution is $\text{play}_L + \overline{\text{play}}_S$. For each state $v \in Q^*$ in which a box $B = B(v)$ is created and each edge $e \in P(B)$ we have that $\sum_{i \in T_e \cap T_S} \bar{X}_{v,i} \leq (1+\varepsilon)d(B)$. We apply Lemma 17 to the tasks i with $\bar{X}_{v,i} = 1$ and obtain a set of tasks that fits into B with profit at least $(1 - O(\varepsilon)) \sum_{i \in T_e} w(i)\bar{X}_{v,i}$. This yields a boxed solution with profit at least $\text{play}_L + (1 - O(\varepsilon))\overline{\text{play}}_S$. ∎

Note that there can be a small task i that the player accepts in some state $v' \in Q^*$ and assigns to some box $B(v)$ but still $\bar{X}_i = 0$ (since i did not fit to $B(v)$ together with the other tasks i' with $\bar{X}_{v,i'} = 1$ considered previously). If this applies to a task i then we set $\tilde{X}_{v,i} = 1$, otherwise we set $\tilde{X}_{v,i} = 0$. We define $\tilde{X}_i = \sum_v \tilde{X}_{v,i}$. So the quantity $\sum_{i \in T_S} w(i)\tilde{X}_i$ is the profit from small tasks that the player obtains but that the solution due to Lemma 26 does not get. We define $\widetilde{\text{play}}_S = \sum_{i \in T_S} w_i E[\tilde{X}_i]$ and then $\text{play}_S = \overline{\text{play}}_S + \widetilde{\text{play}}_S$.

Our goal is now to improve Lemma 12 as follows.

LEMMA 27. *For each task $i \in T_S$ it holds that $(1 + O(\varepsilon))\mathbb{E}[X'_i] \geq (1 - \frac{1}{e})\mathbb{E}[\bar{X}_i] + \mathbb{E}[\tilde{X}_i]$.*

In the rest of this subsection we will prove Lemma 27. By taking the best solution among the solutions $\{x_i\}_{i \in T_S}$, $\{X'_i\}_{i \in T}$, and $\left\{\bar{X}_i\right\}_{i \in T}$ we will obtain an improved approximation ratio of $1 + \frac{1}{1+e} + O(\varepsilon)$.

For each box $B(v)$ created in some state v, we define $\tilde{x}_{v,i}$ as the probability, that the player assigns task i to box $B(v)$ at v or at one of its descendant nodes and $\tilde{X}_i = 1$, *conditioning on the fact* $v \in Q^*$. Similarly, we define $\bar{x}_{v,i} := x_{v,i} - \tilde{x}_{v,i}$, which is the probability that the player assigns task i to box $B(v)$ at v or at one of its descendant nodes and that $\bar{X}_i = 1$, conditioning on the fact $v \in Q^*$. Recall that when we defined the variables X'_i, for each small task i we defined $X'_i := \min\left\{\sum_{v \in Q^*} x_{v,i}, 1\right\}$. Similarly, we define now random variables $\bar{X}'_i$ and $\tilde{X}'_i$ based on $\bar{x}_{v,i}$ and $\tilde{x}_{v,i}$, respectively. For each small task i we define $\bar{X}'_i := \min\left\{\sum_{v \in Q^*} \bar{x}_{v,i}, 1\right\}$ and $\tilde{X}'_i := \min\left\{\sum_{v \in Q^*} \tilde{x}_{v,i}, 1\right\}$. In fact, the next lemma states that always $\tilde{X}'_i \leq \varepsilon$ and therefore always $\tilde{X}'_i = \sum_{v \in Q^*} \tilde{x}_{v,i}$. Intuitively, this separates the probabilities of the events that $\bar{X}_i = 1$ and that $\tilde{X}_i = 1$.

LEMMA 28. *If $\varepsilon' > 0$ is a sufficiently small constant (depending on ε and δ), it holds that $\tilde{X}'_i \leq \varepsilon$.*

Proof. Let $C = O_{\varepsilon,\delta}(1)$ be the maximum number of boxes that can use a given edge e. First, we show that $\tilde{x}_{v,i} \leq \frac{\varepsilon}{C}$ for each state $v \in Q^*$. To this end, let us consider the random set of tasks S_B that the player assigns to box $B = B(v)$ before task i is considered and that use the leftmost edge of i. Notice that, for task i to be discarded, it must happen that $|S_B| \geq (1+\varepsilon)b$. Let also S_B^{up} be the tasks in S_B that are rounded up. Notice that deterministically $|S_B^{up}| \leq 1/\varepsilon'$. Hence an upper bound on $\tilde{x}_{v,i}$ is given by the probability of the event $\mathcal{E}_{v,i}$ that $|S_B^{up}| \leq 1/\varepsilon'$ under the constraint that $|S_B| \geq (1+\varepsilon)b$. Observe however that, under the same constraint, $\mu := \mathbb{E}[|S_B^{up}|] = p_{up}|S_B| \geq \frac{1+\varepsilon}{\varepsilon'}$. Hence we can use Chernoff's bound to conclude

$$\tilde{x}_{v,i} \leq Pr[|S_B^{up}| \leq \frac{1}{\varepsilon'} \mid \mu \geq \frac{1+\varepsilon}{\varepsilon'}] \leq e^{-\frac{1+\varepsilon}{2\varepsilon'}(1-\frac{1}{1+\varepsilon})^2} = e^{-\frac{\varepsilon^2}{2(1+\varepsilon)\varepsilon'}} \leq \frac{\varepsilon}{C},$$

assuming that ε' is a small enough constant depending on ε and δ. There are at most C boxes that are created in states v of Q^* such that $P(i) \subseteq P(B(v))$ (since each edge can be used by at most C boxes). Therefore, $\tilde{X}'_i = \sum_{v \in Q^*} \tilde{x}_{v,i} \leq \frac{\varepsilon}{C} \cdot |\{v \in Q^* : P(i) \subseteq P(B(v))\}| \leq \varepsilon$. ∎

Notice that Lemma 28 does *not* imply that for some task i we can neglect the events where $\tilde{X}_i = 1$ since it might be that $\tilde{X}_i = 1$ is *always* true if the player selects i. Now the key insight is that essentially $X'_i = \bar{X}'_i + \tilde{X}'_i$.

LEMMA 29. *Let i be a small task. It holds that $(1 - O(\varepsilon))(\bar{X}'_i + \tilde{X}'_i) \leq X'_i \leq \bar{X}'_i + \tilde{X}'_i$.*

Proof. If $\bar{X}'_i + \tilde{X}'_i \leq 1$ then it holds that $X'_i = \bar{X}'_i + \tilde{X}'_i$, using that in this case for each state $v \in P^*$ we have that $x_{v,i} = \tilde{x}_{v,i} + \bar{x}_{v,i}$and we are done. Assume now that $\bar{X}'_i + \tilde{X}'_i > 1$. In this case we have that $X'_i = 1$ and $X'_i \leq \bar{X}'_i + \tilde{X}'_i$. Also, $\bar{X}'_i + \tilde{X}'_i \leq 1 + \varepsilon$ since $\bar{X}'_i \leq 1$ by definition and $\tilde{X}'_i \leq \varepsilon$ (see Lemma 28) which implies that $X'_i \geq \frac{\bar{X}'_i + \tilde{X}'_i}{1+\varepsilon}$. ∎

Due to Lemma 29, for proving Lemma 27 it is sufficient to show that $\mathbb{E}[\tilde{X}'_i] = \mathbb{E}[\tilde{X}_i]$ and $\mathbb{E}[\bar{X}'_i] \geq (1 - \frac{1}{e})\mathbb{E}[\bar{X}_i]$.

LEMMA 30. *Let i be a small task. We have that $\mathbb{E}[\tilde{X}'_i] = \mathbb{E}[\tilde{X}_i]$.*

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

Proof. For each small task i we have that

$$\mathbb{E}[\tilde{X}_i] = \sum_v \text{Prob}[v \in Q^*] \cdot \tilde{x}_{v,i} = \sum_{Q \in \mathcal{Q}} \text{Prob}[Q = Q^*] \sum_{v \in P} \tilde{x}_{v,i} = \mathbb{E}[\tilde{X}'_i].$$

□

LEMMA 31. *Let i be a small task. We have that $\mathbb{E}[\bar{X}'_i] \geq (1 - \frac{1}{e})\mathbb{E}[\bar{X}_i]$.*

Proof. Can be shown similarly as Lemma 12. The key difference is the definition of success nodes. Let $\tilde{s}$ be a leaf of the prefix tree T (corresponding to the sink s of the strategy DAG). Let $\tilde{Q}^*$ denote the path from the root of T to $\tilde{s}$ and let Q^* denote the corresponding path in the strategy DAG. We say that $\tilde{s}$ is a success node if on Q^* task i is selected such that $\bar{X}_i = 1$ if we sample Q^* when defining $\bar{X}$. The remainder of the argumentation is analogous. □

Now Lemmas 30 and 31 imply that

$$\mathbb{E}[X'_i] = \mathbb{E}[\bar{X}'_i + \tilde{X}_i] = \mathbb{E}[\bar{X}'_i] + \mathbb{E}[\tilde{X}_i] \geq (1 - \frac{1}{e})\mathbb{E}[\bar{X}_i] + \mathbb{E}[\tilde{X}_i].$$

Together with Lemma 29 this implies that $(1 + O(\varepsilon))\mathbb{E}[X'_i] \geq (1 - \frac{1}{e})\mathbb{E}[\bar{X}_i] + \mathbb{E}[\tilde{X}_i]$, which completes the proof of Lemma 27.

C Setting Without Resource Augmentation

In this section we generalize our reasoning from Sections 2 and 3 and present a polynomial time $(1+\frac{1}{e+1}+\varepsilon)$-approximation algorithm *without* resource augmentation. We use a technique from [GMWZ18] based on [GMWZ17] which—on a high level—sacrifices a factor $1 + \varepsilon$ in the profit and obtains free capacity on each edge (slack) which we then can use similarly as additional capacity due to the resource augmentation.

In contrast to the case of resource augmentation, however, we cannot assume that the edge capacities are in a constant range; neither can we assume that each small task has demand 1. Next, we state a lemma from [GMWZ18], saying that we can gain some free capacity on each edge via sacrificing tasks with a total profit of at most $O(\varepsilon)\text{opt}$. It also establishes an alternative definition of small and large tasks which we will use in the remainder of this section.

LEMMA 32. (GRANDONI ET AL. [GMWZ18]) *Let $\varepsilon > 0$ be an arbitrary constant. For a UFP instance with optimal value* opt*, there exist disjoint subsets of tasks $\text{OPT}_L \subseteq T$ and $\text{OPT}_S \subseteq T$, a value $sl(e) \geq 0$ 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*

$$T_L = \{i \in T \colon d(i) \geq \mu_2 \cdot sl(e) \text{ for some edge } e \in P(i)\} \quad \textit{(large tasks) and}$$
$$T_S = \{i \in T \colon d(i) < \mu_1 \cdot sl(e) \text{ for every edge } e \in P(i)\} \quad \textit{(small tasks)}.$$

Then:

1. *$\text{OPT}_L \subseteq T_L$ and $\text{OPT}_S \subseteq T_S$;*
2. *$w(\text{OPT}_L \cup \text{OPT}_S) \geq (1 - O(\varepsilon))\text{opt}$;*
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:

1. *$sl(e) \in \{(1+\varepsilon)^j \colon j \in \mathbb{N}_0\} \cup \{0\}$;*
2. *There are at most $1/(\mu_2 \cdot \varepsilon^4)$ tasks $i \in T_e \cap \text{OPT}_L$ such that $d(i) \geq \mu_2 \cdot sl(e)$;*
3. *The total demand of tasks $T_e \cap \text{OPT}_S$ is at most $sl(e)$.*
4. *$sl(e) + d(T_e \cap \text{OPT}_L) \leq u(e) - \varepsilon^4 \cdot sl(e)$.*

From now on, we define T_L and T_S from Lemma 32 to be the large and small tasks, respectively. Note that T_L and T_S depend on an unknown optimal solution. For each edge e we say that e *is of type* j with $j \in \mathbb{N}_0$ and write $\text{type}(e) = j$ if $sl(e) = (1+\varepsilon)^j$ and we say that e is of type -1 ($\text{type}(e) = -1$) if $sl(e) = 0$. For each task $i \in T$, we say that i *is of type* j if $P(i)$ includes an edge of type j and no edge of type j' with $j' < j$. Also here, note that the type of each task and each edge depends on an unknown optimal solution.

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

C.1 Boxed solutions We will again work with boxed solutions. For a box B, we say that B *is of type* j if $P(B)$ includes an edge of type j and no edge of type j' with $j' < j$. In contrast to the setting with resource augmentation, now our boxes might have to have different demands. Each box B of type j will have a demand of $b^{(j)} := d(B) = \lfloor \varepsilon^{31}(1+\varepsilon)^j \rfloor$. Also, we will require that if a task i is assigned to a box B, then $d(i)$ is small in comparison to $d(B)$, i.e., $d(i) \leq \varepsilon^2 d(B)$. We therefore adjust the definition of boxed solutions by requiring the latter property. Additionally, we require that on each edge e the large tasks and the boxes leave some slack. We will use this slack later in order to compensate for the use of resource augmentation as in Lemma 4.

DEFINITION 33. *Let $T'_L \subseteq T_L$, $\mathcal{B}$ be a set of boxes, and $T'_{S,B} \subseteq T_S$, $B \in \mathcal{B}$, be pairwise disjoint sets. The pair $(T'_L, \{T'_{S,B}\}_{B\in\mathcal{B}})$ is a* boxed solution *if*

1. *the large tasks T'_L and the boxes $\mathcal{B}$ together respect the edge capacities and leave $\varepsilon^4 \cdot sl(e)/2$ units of slack, i.e., for each $e \in E$ it holds that*

$$\sum_{i\in T'_L \cap T_e} d(i) + \sum_{B\in\mathcal{B}: e\in P(B)} d(B) \leq u(e) - \varepsilon^4 \cdot sl(e)/2; \tag{C.2}$$

2. *for each $B \in \mathcal{B}$ the tasks in $T'_{S,B}$ fit into B and $d(i) \leq \varepsilon^2 d(B)$ for each $i \in T'_{S,B}$;*

3. *for each edge e and each $j \geq 0$, there are at most $1/\varepsilon^{31}$ boxes B of type j with $e \in P(B)$.*

In order to prove that there exists a boxed solution which has a profit of at least $(1 - O(\varepsilon))\mathrm{opt}$, we need a strengthened version of Lemma 2. For each $j \in \mathbb{N}_0$ let $\mathrm{OPT}_S^{(j)} \subseteq \mathrm{OPT}_S$ denote the tasks of type j in OPT_S.

In order to construct a boxed solution according to Definition 33, we first replace each task i in OPT_S by $d(i)$ tasks with demand 1 and with profit $w(i)/d(i)$ each. Let OPT'_S denote the resulting set of tasks. Observe that $d(\mathrm{OPT}'_S) = \mathrm{opt}_S$. We compute now first a boxed solution for $\mathrm{OPT}_L \cup \mathrm{OPT}'_S$. Later, we will transform it into a boxed solution for essentially all tasks in OPT.

LEMMA 34. *For a sufficiently small $\varepsilon > 0$, there exists a boxed solution $\mathrm{OPT}'_{\mathrm{box}}$ with tasks $\mathrm{OPT}_L \cup \mathrm{OPT}'_S$ and boxes $\mathcal{B}^*$ in which there are no boxes of type -1 and for each $j \in \mathbb{N}_0$ we have that each task $i \in \mathrm{OPT}'_S$ of type j is assigned to a box of type j and for each box $B \in \mathcal{B}^*$ of type j it holds that $d(B) = \lfloor \varepsilon^{31}(1+\varepsilon)^j \rfloor$.*

Proof. Note that OPT'_S does not include any task of type -1. For each $j \in \mathbb{N}_0$ let $(\mathrm{OPT}'_S)^{(j)} \subseteq \mathrm{OPT}'_S$ denote the tasks in OPT'_S of type j. For each $j \in \mathbb{N}_0$ and each edge e we define an auxiliary capacity $u^{(j)}(e)$ which intuitively corresponds to the capacity needed by the tasks in $(\mathrm{OPT}'_S)^{(j)} \cap T_e$. The auxiliary capacity is composed of the demand of the tasks and a share of the available slack. For each j, we define $\mathrm{slack}_j := \varepsilon^6 \cdot (1+\varepsilon)^j$. For each $j \geq 0$ and each edge e of type at least j, we set $u^{(j)}(e) := \sum_{i\in T_e \cap (\mathrm{OPT}'_S)^{(j)}} d(i) + \mathrm{slack}_j$. The remaining edges do not belong to the sub-instances that we will consider for each type j.

We claim that all tasks in the sub-instances are small with respect to our original definition of small tasks in Section 1. Let $u^{(j)}_{\min} := \min_{e\in E:\ \mathrm{type}(e)\geq j} u_e^{(j)}$ and $u^{(j)}_{\max} := \max_{e\in E:\ \mathrm{type}(e)\geq j} u_e^{(j)}$. We set $\delta := \varepsilon^6/2$ and claim that each edge has a fraction at least δ of slack, i.e., for each edge e of a type-j sub-instance, $d((\mathrm{OPT}'_S)^{(j)} \cap T_e) \leq (1-\delta)u^{(j)}(e)$. To this end, we express $u^{(j)}_{\max}$ in terms of the slack. Let us consider the closest type-j edge e_L on the left of e and e_R on the right of e, respectively. By property 3 of Lemma 32, $d(T_{e_L} \cap OPT_S) \leq (1+\varepsilon)^j$ and $d(T_{e_R} \cap OPT_S) \leq (1+\varepsilon)^j$; if e_L or e_R do not exist, the demand is zero. Since these are the only tasks of type at most j in T_e, the total demand of small type-j tasks at e is bounded from above by $2(1+\varepsilon)^j$. Then the fraction of available slack for each edge is at least

$$\mathrm{slack}_j / u^{(j)}_{\max} = \varepsilon^6(1+\varepsilon)^j / u^{(j)}_{\max} \geq \varepsilon^6(1+\varepsilon)^j / 2(1+\varepsilon)^j = \varepsilon^6/2 = \delta\,.$$

Let i be a task in $(\mathrm{OPT}'_S)^{(j)}$. Observe that for each j, $u^{(j)}_{\min} \geq \mathrm{slack}_j$, i.e., the minimum capacity available a for type-j task is at least its share of the slack. By introducing a dummy-edge e of demand exactly slack_j with $T_e = \emptyset$, w.l.o.g. we may assume $\mathrm{slack}_j = u^{(j)}_{\min}$. For small enough ε, the demand of each task i is at most

$$d(i) \leq \mu_1(1+\varepsilon)^j \leq \varepsilon^4/(1+\varepsilon)^{1/\varepsilon^3}(1+\varepsilon)^j \leq \varepsilon^2\delta\varepsilon^6(1+\varepsilon)^j = \varepsilon^2\delta\,\mathrm{slack}_j = \varepsilon^2\delta u^{(j)}_{\min}\,.$$

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

In order to find the set of boxes, we want to apply Lemma 4 to the sub-instance given by $(\mathrm{OPT}'_S)^{(j)}$ and $\left\{u^{(j)}(e)\right\}_{e\in E}$. To apply Lemma 4 we need to ensure that $\delta^4 u^{(j)}_{\min} \geq 1$. For $j > \log_{1+\varepsilon}(1/\varepsilon^{31})$, the condition is satisfied because

$$\delta^4 u^{(j)}_{\min} = (\varepsilon^6/2)^4 \cdot \mathrm{slack}_j = \varepsilon^{24}/16 \cdot \varepsilon^6(1+\varepsilon)^j \geq \varepsilon^{31}(1+\varepsilon)^j\,.$$

For types j with $j \leq \log_{1+\varepsilon}(1/\varepsilon^{31})$ we have that for each task i in $(\mathrm{OPT}'_S)^{(j)}$ it holds that

$$d(i) \leq \mu_1(1+\varepsilon)^j \leq \frac{\varepsilon^4}{(1+\varepsilon)^{1/\varepsilon^3}}\frac{1}{\epsilon^{31}}.$$

If ϵ is sufficiently small this implies that $d(i) < 1$ and hence there are not small tasks of type j, i.e., we can ignore the sub-instances of these types j.

Applying Lemma 4 for each j gives us a set of boxes for the tasks in $(\mathrm{OPT}'_S)^{(j)}$. Assume w.l.o.g. that each box contains at least one task of $(\mathrm{OPT}'_S)^{(j)}$. Note that each box B containing a task i of type j must use also an edge of type j and cannot use an edge of some type $j' < j$ (since otherwise i would not be of type j). Hence, B is also of type j. The demand of each box of type j is $b^{(j)} := \lfloor \delta^4 \cdot u^{(j)}_{\min} \rfloor = \lfloor \delta^4 \cdot \mathrm{slack}_j \rfloor = \lfloor \varepsilon^{31}(1+\varepsilon)^j \rfloor$, as claimed in the lemma. Let $\mathcal{B}^*$ be the set of all boxes generated for all types j above. Let also $\{\mathrm{OPT}'_{S,B}\}_{B\in\mathcal{B}^*}$ be the resulting assignment of the tasks in OPT'_S to the boxes. We claim that $\mathrm{OPT}'_{\mathrm{box}} := (OPT_L, \{\mathrm{OPT}'_{S,B}\}_{B\in\mathcal{B}^*})$ is a boxed solution. Condition 2 of Definition 33 is satisfied by construction, since all tasks in OPT'_S fit into the boxes $\mathcal{B}^*$ and for each task i in box B of type j, $d(i) \leq \varepsilon^2 b^{(j)}$.

For condition 1, we show next that for each edge e of type j, $\sum_{j'\leq j} \mathrm{slack}_j \leq \varepsilon^4 sl(e)/2$, i.e., all boxes using e have a total demand of at most $u(e) - \sum_{i\in\mathrm{OPT}_L\cap T_e} d(i) - \varepsilon^4 \cdot sl(e)/2$. To see that we did not exceed the available slack, let $\hat{e}$ be a type-$\hat{j}$ edge. Then, by Lemma 32, there is a slack of $\varepsilon^4 sl(\hat{e}) = \varepsilon^4(1+\varepsilon)^{\hat{j}}$ available at e. By definition, there are no small tasks using $\hat{e}$ of types larger than $\hat{j}$. For all $\bar{j} \leq \hat{j}$ together we have assigned a slack of

$$\sum_{0\leq\bar{j}\leq\hat{j}} \mathrm{slack}_{\bar{j}} = \sum_{0\leq\bar{j}\leq\hat{j}} \varepsilon^6 \cdot (1+\varepsilon)^{\bar{j}} = \varepsilon^6 \cdot \frac{(1+\varepsilon)^{\hat{j}+1}-1}{\varepsilon} \leq \frac{\varepsilon^4 sl(\hat{e})}{2} \cdot 2\varepsilon(1+\varepsilon) \leq \frac{\varepsilon^4 sl(\hat{e})}{2}\,.$$

Thus half of the slack $\varepsilon^4 sl(\hat{e})$ due to condition 4 of Lemma 32 is still available and therefore

$$\sum_{i\in T'_L\cap T_e} d(i) + \sum_{B\in\mathcal{B}^*: e\in P(B)} d(B) \leq u(e) - \varepsilon^4 \cdot sl(e)/2\,.$$

For condition 3, we observe that the number of boxes crossing an edge e is bounded from above by

$$\frac{u^{(j)}_{\max}}{\lfloor \delta^4 u^{(j)}_{\min} \rfloor} \leq \frac{2(1+\varepsilon)^j}{\lfloor (\varepsilon^6/2)^3 \mathrm{slack}_j \rfloor} \leq \frac{(1+\varepsilon)^j}{\varepsilon^{31}(1+\varepsilon)^j} = 1/\varepsilon^{31}\,.$$

We have not discarded tasks from OPT_L and we have assigned each task in OPT'_S. ∎

Next, we interpret $\mathrm{OPT}'_{\mathrm{box}}$ as a solution in which the small tasks OPT_S are assigned fractionally to the boxes, e.g., if for a task i there are $k_{i,B}$ corresponding tasks from OPT'_S (with demand 1 each) assigned to a box B, then we interpret this such that a $k_{i,B}/d(i)$-fraction of i is assigned to B. Via LP-rounding we transform this to a boxed solution $\mathrm{OPT}_{\mathrm{box}}$ where in the process we might lose a factor $1 - O(\epsilon)$ of the profit from small tasks, and hence obtain an overall profit of $(1 - O(\epsilon))\mathrm{opt}$.

LEMMA 35. *There exists a boxed solution* $\mathrm{OPT}'_{\mathrm{box}}$ *with profit* $\mathrm{opt}_{\mathrm{box}} \geq (1 - O(\varepsilon))\mathrm{opt}$.

Proof. We want to convert $\mathrm{OPT}'_{\mathrm{box}}$ into a boxed solution for $\mathrm{OPT}_S \cup \mathrm{OPT}_L$. Recall that $\mathcal{B}^*$ are its boxes. To this end, we formulate an integer program whose LP-relaxation we denote by (BOX-LP). There is a variable $x_{i,B}$ for each box B and each small task i such that i can be assigned to B. (If i does not fit into B, we assume $x_{i,B} = 0$ for notational convenience.)

$$\begin{array}{lll} \max & \displaystyle\sum_{i\in\mathrm{OPT}_S, B\in\mathcal{B}^*} w(i)x_{i,B} & \\ \text{s.t.} & \sum_{i\in T_e\cap\mathrm{OPT}_S} d(i)x_{i,B} \leq d(B) & \forall B\in\mathcal{B}^*, \forall e\in P(B) \\ & \sum_{B\in\mathcal{B}^*} x_{i,B} \leq 1 & \forall i\in\mathrm{OPT}_S \\ & x_{i,B} \geq 0 & \forall B\in\mathcal{B}^*, \forall i\in\mathrm{OPT}_S : P(i)\subseteq P(B) \end{array} \qquad \text{(C.1)}$$

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

Denote by x^* the following solution to the LP. For each $i \in \mathrm{OPT}_S$ and each $B \in \mathcal{B}^*$ let $k_{i,B}$ denote the number of tasks in OPT'_S (of demand 1 each) corresponding to i that are assigned to box B in $\mathrm{OPT}'_{\mathrm{box}}$; we define $x^*_{i,B} = k_{i,B}/d(i)$. Following a technique by Calinescu et al. [CCKR11] we round x^* using randomized rounding with alteration. For each task i independently, we define random variables $Y_{i,B}$, one for each box $B \in \mathcal{B}^*$. We define them such that for each box $B \in \mathcal{B}^*$ we have $\Pr[Y_{i,B} = 1] = (1-\varepsilon)x_{i,B}$ and for any two boxes $B, B' \in \mathcal{B}^*$ we have that $\Pr[Y_{i,B} = 1 \wedge Y_{i,B'} = 1] = 0$. Such a distribution can easily be obtained via dependent rounding similar to Bertsimas et al. [BTV99]: let p be a random number in $[0,1]$ and suppose the boxes in $\mathcal{B}^*$ with $x_{i,B} > 0$ are numbered $B_1, ..., B_k$. If $(1-\varepsilon)\sum_{j'<j} x_{i,B_j} \leq p < (1-\varepsilon)\sum_{j' \leq j} x_{i,B_j}$ then we define $Y_{i,B_j} := 1$ and $Y_{i,B_{j'}} := 0$ for each $j' \neq j$.

After defining the $Y_{i,B}$ variables we do an alteration phase and define random variables $Z_{i,B} \in \{0,1\}$ such that $Z_{i,B} \leq Y_{i,B}$ for each $i \in T$ and each $B \in \mathcal{B}^*$. Note that due to the dependent rounding we can perform this step on each box independently. Consider a box B. We order the tasks with $Y_{i,B} = 1$ by their start vertex from left to right, breaking ties arbitrarily. In particular, we define $Z_{i,B}$ only after all variables $Z_{i',B}$ have been defined for all tasks $i' < i$. For each task i, let e_i denote the first edge of $P(i)$ We define

$$Z_{i,B} := \begin{cases} 1 & \text{if } Y_{i,B} = 1 \text{ and } \sum_{i'<i:\ e_i \in P(i) \cap P(i')} Z_{i',B} d_{i'} \leq d(B) - d(i) \\ 0 & \text{otherwise} \end{cases}$$

The reason to define the variables $Z_{i,B}$ is that for every outcome of the random experiment they yield a feasible solution. Note that due to the order of added tasks, if adding a task i to a box B exceeds the demand of B, the demand has to be exceeded at e_i. It remains to bound the expected value of this solution. For each pair of two tasks i, i' with $P(i), P(i') \subseteq P(B)$ we have $\Pr[Y_{i,B} = 1 \wedge Y_{i',B} = 1] = \Pr[Y_{i,B} = 1] \cdot \Pr[Y_{i',B} = 1]$ since the dependent rounding only introduced dependencies of Y-variables in distinct boxes. Assuming that ϵ is a sufficiently small constant, we observe that if $x^*_{i,B} > 0$ for some task i and a box B then we have that $d(i) \leq \epsilon^2 d(B)$ since (i) in $\mathrm{OPT}'_{\mathrm{box}}$ each task of some type j is assigned to a box of type j, (ii) each task $i \in \mathrm{OPT}_S$ of type j satisfies that $d(i) \leq \frac{\epsilon^4}{(1+\epsilon)^{1/\epsilon^3}}(1+\epsilon)^j \leq \epsilon^{34}(1+\epsilon)^j$ and (iii) each box B of type j has demand $d(B) = \lfloor \epsilon^{31}(1+\epsilon)^j \rfloor \geq \epsilon^{32}(1+\epsilon)^j$. Therefore, we can use an argumentation by Calinescu et al. [CCKR11] to show that $\Pr[Z_{i,B} = 0 | Y_{i,B} = 1] \leq \epsilon$ and hence $\mathbb{E}[\sum_{B \in \mathcal{B}^*} Z_{i,B}] \geq (1-\epsilon)\sum_{B \in \mathcal{B}^*} x_{i,B}$ for each $i \in \mathrm{OPT}_S$. ∎

C.2 UFP-Game We define a UFP-Game for the case without resource augmentation. The intuition of the game is as follows. The game works in phases with one phase corresponding to each type j. In phase j the player selects the boxes and tasks of type j. For this it would be helpful to know which edges of G are of type j; however, we do not know this (note that Lemmas 32 and 34 give only existential results). Therefore, the player needs to guess these edges. In each phase j she selects the large tasks and the boxes of type j and assigns small tasks of type j into these boxes. Note that each edge of type j is used by at most $O_\varepsilon(1)$ large tasks of type j and by at most $O_\varepsilon(1)$ boxes of type j in a boxed solution. Let e, e' be two edges of type j such that there are no edges of type j between e and e'. Then the player recurses into a *subgame* on the path G' between e and e' in which each edge in G' is assumed to be of type $\bar{j} := j + 1$ or higher.

For each j, let $L^{(j)}$ be the set of tasks $i \in T_L$ such that $\mu_2(1+\varepsilon)^j \leq d(i) < \mu_2(1+\varepsilon)^{j+1}$. Formally, each subgame is defined via the following parameters.

- A phase $\bar{j}$.
- A subpath $\bar{G} \subseteq G$, the intuition being that each edge $e \in \bar{G}$ is of type $\bar{j}$ or higher; we require that $u(e) \geq (1+\varepsilon)^{\bar{j}}$ for each $e \in \bar{G}$.
- A set $\bar{\mathcal{B}}$ of boxes of types j' with $\bar{j} - 1 \geq j' \geq \bar{j} - 1/\varepsilon^3$ where each box $B \in \bar{\mathcal{B}}$ uses the leftmost or the rightmost edge of $\bar{G}$. Intuitively, these boxes were previously created. The boxes in $\bar{\mathcal{B}}$ cannot be used for tasks selected in the subgame.
- A set $\bar{T}_L$ of large tasks such that each task $i \in \bar{T}_L$ uses the leftmost or the rightmost edge of $\bar{G}$ and $i \in \bigcup_{j \geq \bar{j} - 1/\varepsilon^3} L^{(j)}$. Intuitively, these tasks were previously selected.

At the very beginning, the player starts playing the subgame $(-1, G, \emptyset, \emptyset)$ which represents the whole game. We describe now how to play a given subgame $(\bar{j}, \bar{G}, \bar{\mathcal{B}}, \bar{T}_L)$. The goal of this subgame is to select a set of tasks T' such that for each $i \in T'$ we have that $P(i) \subseteq \bar{G}$ and intuitively the tasks in T' form a boxed solution together with the tasks in $\bar{T}_L$ and the boxes in $\bar{\mathcal{B}}$. One complication is that without resource augmentation we can no longer directly argue that we can assume

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

w.l.o.g. that $d(i) = 1$ for each small task i (like in Lemma 2). However, using the slack we replace the potentially small tasks by tasks of size $\mu_{\bar{j}} := \frac{\varepsilon^{27}}{n}(1+\varepsilon)^{\bar{j}}$ each. Formally, we define the input tasks T_{input} for the subgame $(\bar{j}, \bar{G}, \bar{\mathcal{B}}, \bar{T}_L)$ as follows: intuitively, each task $i \in T$ with $P(i) \subseteq \bar{G}$ and $d(i) \geq \mu_2 \cdot (1+\varepsilon)^{\bar{j}}$ is potentially a large task of level $\bar{j}$. Therefore, we add to T_{input} each such task. Each task $i \in T$ with $P(i) \subseteq \bar{G}$ and $d(i) < \mu_1 \cdot (1+\varepsilon)^{\bar{j}}$ is potentially a small task of level $\bar{j}$. For each such task i, we add $n_i := \left\lceil \frac{d(i)}{\mu_j} \right\rceil$ tasks of size μ_j and weight $w(i)/n_i$ each to T_{input}. At the end, for convenience we modify $\bar{G}$ such that each vertex is the start or end vertex of at most one task in T_{input} , i.e., by duplicating edges if necessary.

We now define the states of each subgame. Consider a subgame $(\bar{j}, \bar{G}, \bar{\mathcal{B}}, \bar{T}_L)$. Each state of the subgame is defined very similarly as the states of the UFP-Game defined in Section 3. The key difference is that the player has additional choices if the current edge e is of type $\bar{j}$ (or say it is guessed to be of type $\bar{j}$). The player then defines the next edge e' of type $\bar{j}$, i.e., the edge of type $\bar{j}$ lying on the right of e that is closest to e. Additionally, she selects all large tasks and creates all boxes of type $\bar{j}$ that use e' that she wants to use during the course of the game. In particular, later she cannot select more large tasks or create more boxes that use e'. Therefore, in addition to the properties of the states in the UFP-Game defined in Section 3, now a state defines whether the current edge e is of type $\bar{j}$, and if not, the state includes the next edge e' on the right of e that is of type $\bar{j}$ (which intuitively was guessed before). Formally, each state of a subgame $(\bar{j}, \bar{G}, \bar{\mathcal{B}}, \bar{T}_L)$ is defined by

- an edge e;
- a set L of previously selected large tasks using edge e in $\bigcup_{j \geq \bar{j} - 1/\varepsilon^3} (L^{(j)} \cap T_e)$;
- a set $\mathcal{B}$ of previously created boxes using edge e; for each type $j \geq \bar{j} - 1/\varepsilon^2$ there are at most $O_\varepsilon(1)$ boxes $B \in \mathcal{B}$ with $d(B) = \lfloor \varepsilon^{31}(1+\varepsilon)^j \rfloor$ and there is no box $B \in \mathcal{B}$ with $d(B) < (1+\varepsilon)^j$ with $j = \bar{j} - 1/\varepsilon^3$;
- a set S^{up} of previously selected rounded up small tasks i using edge e;
- a partition $\{S^{up}_B\}_{B \in \mathcal{B}}$ of S^{up} such that for each box $B \in \mathcal{B}$ the tasks in S^{up}_B fits in the respective B if they are all rounded up.
- the information whether e is of type $\bar{j}$ or of a type j with $j > \bar{j}$; in the latter case the definition of the state additionally includes
  - an edge e' on the right of e which intuitively is the edge of type $\bar{j}$ closest to e on the right of e;
  - a set L' of previously selected large tasks in $\bigcup_{j \geq \bar{j} - 1/\varepsilon^3} (L^{(j)} \cap T_e)$ using edge e' (it might be that $L \cap L' \neq \emptyset$);
  - a set $\mathcal{B}'$ of previously created boxes using edge e' (it might be that $\mathcal{B} \cap \mathcal{B}' \neq \emptyset$); for each type $j \geq \bar{j} - 1/\varepsilon^3$ there are at most $O_\varepsilon(1)$ boxes $B \in \mathcal{B}$ with $d(B) = \lfloor \varepsilon^{31}(1+\varepsilon)^j \rfloor$ and there is no box $B \in \mathcal{B}$ with $d(B) < (1+\varepsilon)^j$ with $j = \bar{j} - 1/\varepsilon^3$;

Assume that we are given such a state. We now define the actions of the player given this state. It might be that there is a task $i \in T_{\text{input}}$ such that e is the first edge of $P(i)$ and intuitively the state implies that i can be assigned to a box of type $\bar{j}$. Formally, this happens when according to the state e is the first edge of $P(i)$ and either e is of type $\bar{j}$, $\bar{j} \geq 0$, and $d(i) \leq \mu_1(1+\varepsilon)^{\bar{j}}$ (note that $sl(e) = (1+\varepsilon)^{\bar{j}}$ if e is of type $\bar{j}$), or e is not of type j', but $e' \in P(i)$ and $d(i) \leq \mu_1 sl(e) = \mu_1(1+\varepsilon)^{\bar{j}}$. In this case the player needs to decide whether she accepts i or not. If she accepts i then she gains $w(i)$ and has to assign i into a box $B \in \mathcal{B}$. Afterwards, the size of i is defined randomly. We define $d'(i) := \varepsilon' d(B)$ with probability $p_{up} = \frac{\mu_j}{\varepsilon' \cdot d(B)}$ (i.e., i is rounded up) and $d'(i) := 0$ with probability $1 - p_{up}$ (i.e., i is rounded down). This implies that $\mathbb{E}[d'(i)] = d(i)$. Note that in contrast to the UFP-Game in Section 3, at this point the player is not allowed to create a new box into which she might assign i. Observe also that the player is not allowed to select i if according to the state neither e is of type $\bar{j}$ and $e \in P(i)$ nor e' is of type $\bar{j}$ and $e' \in P(i)$. Intuitively, then i is of some type $j \geq \bar{j} + 1$ and the player can select i in a different subgame of type j.

Independent of whether a task i as above exists, if e is of type $\bar{j}$ (according to the state) then the gambler needs to select an edge e' which intuitively is the edge on the right of e closest to e which is of type $\bar{j}$. Then she can select a set L' of at most $1/(\mu_2 \cdot \varepsilon^4)$ large tasks from $\bigcup_{j \geq \bar{j}} L^{(j)}$ using e', i.e., such that each task $i \in L'$ satisfies that $e' \in P(i)$, $d(i) \geq \mu_2(1+\varepsilon)^{\bar{j}}$, and that in L' there are in total at most $1/(\mu_2 \cdot \varepsilon^4)$ tasks that use e'. It is necessary that L and L' are consistent, i.e., $L' \cap T_e = L \cap T_{e'}$. Then the gambler gains $w(L' \setminus L)$. Then she defines a set of boxes $\mathcal{B}'$ that use e', each with demand $d(B) = \lfloor \varepsilon^{31}(1+\varepsilon)^j \rfloor$ such that at most $1/\varepsilon^{31}$ of them use e'. Also here $\mathcal{B}$ and $\mathcal{B}'$ need to be consistent, i.e., the boxes in $\mathcal{B}'$ that use e need to be identical to the boxes in $\mathcal{B}$ that use e'. She creates the (new) boxes $\mathcal{B}' \setminus \mathcal{B}$. For L' and

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

$\mathcal{B}'$ it is necessary that on each edge e'' on the right of e they obey the edge capacity and leave some slack. Formally, we require

$$\sum_{B\in\mathcal{B}\cup\mathcal{B}'\colon e''\in P(B)} d(B) + \sum_{i\in L'\cap T_{e''}} d(i) \le u(e'') - \varepsilon^4(1+\varepsilon)^{\bar{j}}/2\,.$$

Then the gambler starts playing the subgame $(\bar{j}+1, G'', \mathcal{B}'', L'')$ where G'' is defined as the subpath between e and e' (excluding e and e') and $\mathcal{B}''$ are the boxes in $\mathcal{B}\cup\mathcal{B}'$ that use at least one edge of G'' and which satisfy that $d(B) \ge \lfloor \varepsilon^{31}(1+\varepsilon)^{\bar{j}+1-1/\varepsilon^3}\rfloor$, and L'' are the tasks within $L\cup L'$ that use at least one edge of G'' without the tasks in $(L\cup L')\cap\bigcup_{j>\bar{j}-1/\varepsilon^3} L^{(j)}$. We will show later that states do not exceed the edge capacities. Note that the input tasks for the subgame $(\bar{j}+1, G'', \mathcal{B}'', L'')$ are defined from scratch, i.e., they are not necessarily identical to T_{input}.

Observe also that while playing the subgame the gambler might gain something and this profit contribute towards the total profit. It can happen that $G''=\emptyset$ in which case the subgame ends immediately and the gambler does not gain anything. Once the gambler finishes playing the subgame $(\bar{j}+1, G'', \mathcal{B}'', L'')$, she continues playing the (parent) subgame $(\bar{j}, \bar{G}, \bar{\mathcal{B}}, \bar{T}_L)$.

Independent of whether e is of type $\bar{j}$, the next state of the subgame $(\bar{j}, \bar{G}, \bar{\mathcal{B}}, \bar{L})$ is defined as follows: we define i to be the next potentially small input task; formally, let i be the input task with leftmost start vertex that satisfies $e\notin P(i)$ and $d(i)\le \mu_1(1+\varepsilon)^{\bar{j}}$. On the one hand, if the first edge of $P(i)$ lies strictly on the left of e', then the next state is $(\tilde{e}, \tilde{L}, \tilde{\mathcal{B}}, \tilde{S}^{up}, \{\tilde{S}^{up}_B\}_{B\in\tilde{\mathcal{B}}}, e', \tilde{L}', \tilde{\mathcal{B}}')$ where $\tilde{e}$ is the first edge of $P(i)$, $\tilde{L}$ are the (large) tasks in $L\cup L'$ that use $\tilde{e}$, $\tilde{L}'$ are the (large) tasks in $L\cup L'$ that use e', $\tilde{\mathcal{B}}$ are the boxes in $\mathcal{B}\cup\mathcal{B}'$ that use $\tilde{e}$, $\tilde{\mathcal{B}}'$ are the boxes in $\mathcal{B}\cup\mathcal{B}'$ that use e', $\tilde{S}$ and $\tilde{f}$ are the set $\tilde{S}^{up}$ and its partition $\{\tilde{S}^{up}_B\}_{B\in\mathcal{B}}$ are the sets S^{up} and the partition $\{S^{up}_B\}_{B\in\mathcal{B}}$ restricted to tasks i'' such that $P(i'')$ uses $\tilde{e}$, and if $d'(i)>0$ then additionally $\tilde{S}^{up}_B$ includes i where B is the box into which i was assigned before. On the other hand, if the first edge of $P(i)$ is identical to e' or lies on the right of e' then the next state is $(e', L', \mathcal{B}', (S')^{up}, \{(S')^{up}_B\}_{B\in\mathcal{B}'})$ where again $(S')^{up}$ and $\{(S')^{up}_B\}_{B\in\mathcal{B}'}$ are the set S^{up} and its partition $\{S^{up}_B\}_{B\in\mathcal{B}}$ restricted to tasks i'' such that $P(i'')$ uses e', and if $d'(i)>0$ then additionally $(S')^{up}_B$ includes i where again B is the box into which i was assigned before.

Whenever a subgame $(\bar{j}, \bar{G}, \bar{\mathcal{B}}, \bar{T}_L)$ is started, for convenience we add edges e_L, e_R on the left and on the right of $\bar{G}$ and define that they are of type $\bar{j}$. Then $(e_L, \emptyset, \emptyset, \emptyset,)$ is the first state when playing $(\bar{j}, \bar{G}, \bar{\mathcal{B}}, \bar{T}_L)$.

LEMMA 36. *The number of subgames and the number of states in each subgame is bounded by* $n^{O_\varepsilon(1)}$.

Proof. The number of subgames $(j', G', \mathcal{B}', L')$ is determined by at most n choices for j', n^2 choices for G', $n^{O_\varepsilon(1)}$ choices for $\mathcal{B}'$ for given j', e', G', and $n^{O_\varepsilon(1)}$ choices for L' for given j', e', G'.

The number of states $(e, L, \mathcal{B}, S^{up}, \{S^{up}_B\}_{B\in\mathcal{B}}, e', L', \mathcal{B}')$ is determined by a subgame, $O(n)$ choices for e, $n^{O_\varepsilon(1)}$ choices for L for a given e, $n^{O_\varepsilon(1)}$ choices for $\mathcal{B}$ for a given e, $n^{O_\varepsilon(1)}$ choices for S^{up}, and $O_\varepsilon(1)$ choices for $\{S^{up}_B\}_{B\in\mathcal{B}}$ for a given S^{up}. Thus the overall number of subgames and states is bounded by $n^{O_\varepsilon(1)}$. ∎

We define the *strategy* of the gambler as the (unique) decision that the gambler makes at each state of each subgame, i.e., whether or not to accept the current task i and if applicable which will be the next edge e' of type $\bar{j}$ and which are the corresponding boxes $\mathcal{B}'$ and large tasks L'. Observe that for each state this yields at most $n^{O_\varepsilon(1)}$ different choices. Since the number of subgames and the number of states in each subgame is bounded by $n^{O_\varepsilon(1)}$ we can compute the optimal strategy in polynomial time, similarly as in Lemma 6.

LEMMA 37. *In time* $n^{O_\varepsilon(1)}$ *one can compute an optimal strategy.*

Proof. Analogously to the proof of Lemma 6, we can compute an optimal strategy for each sub-game separately. Given that, we define a DAG of subgames and compute the strategy in reverse-topological order in a similar fashion. ∎

Also, similarly as in Lemma 6, we can show that there exists a strategy whose profit is at least $(1-\varepsilon)\mathrm{opt}'_{\mathrm{box}} \ge (1-O(\varepsilon))\mathrm{opt}$. Let play denote the expected profit of the optimal strategy.

LEMMA 38. *For a sufficiently small* $\varepsilon>0$ *and* $\varepsilon' \le \frac{\varepsilon^2}{3(1-\varepsilon)\ln(1/\varepsilon)}$ *it holds that* $\mathrm{play} \ge (1-\varepsilon)\mathrm{opt}_{\mathrm{box}} \ge (1-O(\varepsilon))\mathrm{opt}$.

Proof. We prove this lemma similarly as Lemma 7. Let $(\mathrm{OPT}'_L, \{\mathrm{OPT}'_{S,B}\}_{B\in\mathcal{B}^*})$ be the boxed solution defined in Lemma 34. Using Lemma 17, for each $B\in\mathcal{B}^*$ we determine a set $\mathrm{OPT}''_{S,B}\subseteq\mathrm{OPT}'_{S,B}$ such that $w(\mathrm{OPT}''_{S,B}) \ge (1-O(\varepsilon))w(\mathrm{OPT}'_{S,B})$ and on each edge $e\in P(B)$ the tasks in $\mathrm{OPT}''_{S,B}\cap T_e$ have a total demand of at most $(1-2\varepsilon)d(B)$, i.e., leave a portion of $2\varepsilon\cdot d(B)$ free. Let $\mathrm{OPT}''_S = \cup_{B\in\mathcal{B}^*}\mathrm{OPT}''_{S,B}$. We next run the gambler with the following strategy.

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

Tasks in OPT'_L are always selected and each box in $\mathcal{B}^*$ is created. For a small task $i \in \mathrm{OPT}''_{S,B}$, the gambler accepts each slice i' corresponding to i and assigns it to B whenever she can. In particular, if e is the current edge and $S^{up}_B \subseteq \mathrm{OPT}''_{S,B}$ is the current set of selected rounded up task slices assigned to B and using edge e, then i is accepted iff $|S^{up}_B| < \frac{1}{\varepsilon'}$. Observe that this gambler satisfies all the constraints.

Note that for each small task i of type j the total size of its slices is $\left\lceil \frac{d(i)}{\mu_j} \right\rceil \cdot \mu_j \leq d(i) + \mu_j$. Therefore, for each box $B \in \mathcal{B}^*$ and each edge $e \in P(B)$ the total size of the slices in $\mathrm{OPT}''_{S,B}$ whose path uses e is bounded by $d(\mathrm{OPT}''_{S,B} \cap T_e) + \mu_j |\mathrm{OPT}''_{S,B} \cap T_e| \leq (1-\epsilon) d(B)$. In the previous inequality we used the fact that $\mu_j = \frac{\varepsilon^{27}}{n}(1+\varepsilon)^j$ and $d(B) = \varepsilon^{25}(1+\varepsilon)^j$ (since B is of type j). Therefore, their total number is bounded by $(1-\epsilon)d(B)/\mu_j$. In order to prove the claim it is sufficient to show that each slice i' corresponding to a task $i \in \mathrm{OPT}''_S$ is selected with probability at least $1 - O(\varepsilon)$. With the above notation, in order for i' to be discarded it must necessarily happen that $|S^{up}_B| \geq \frac{1}{\varepsilon'}$ while the total number is at most $(1-\epsilon)d(B)/\mu_j$. Thus $E[|S^{up}_B|] \leq p_{up}(1-\epsilon)\frac{d(B)}{\mu_j} = \frac{\mu_j}{\epsilon' \cdot d(B)}(1-\epsilon)\frac{d(B)}{\mu_j} = \frac{1-\epsilon}{\epsilon'}$. Therefore by Chernoff's bound:

$$Pr[|S^{up}_{B,t}| \geq \frac{1}{\epsilon'}] \leq e^{-\frac{1-\varepsilon}{3\varepsilon'}(\frac{1}{1-\varepsilon}-1)^2} = e^{-\frac{\varepsilon^2}{3\varepsilon'(1-\varepsilon)}} \leq \varepsilon.$$

□

Notice that the boxes and the large tasks selected by the gambler respect the edge capacities, independently how the small tasks are rounded.

LEMMA 39. *For a sufficiently small $\varepsilon > 0$, let $\mathcal{B}'$ and T'_L denote the boxes and large tasks selected by the gambler. For each edge e it holds that $\sum_{B \in \mathcal{B}' :\, e \in P(B)} d(B) + \sum_{i \in T'_L \cap T_e} d(i) \leq u(e)$.*

Proof. By the definition of the gambler in phase j, we have

$$\sum_{B \in \mathcal{B}'^{(j')} :\, j' \geq j - 1/\varepsilon^3 \text{ and } e \in P(B)} d(B) + \sum_{j' \geq j - 1/\varepsilon^3, i \in T'^{(j')}_L \cap T_e} d(i) \leq u(e) - \varepsilon^4 (1+\varepsilon)^j/2\,.$$

We have to show that all remaining tasks fit into the slack of $\varepsilon^4(1+\varepsilon)^j/2$, i.e., the total demand of boxes and large tasks of type smaller than $j' := j - 1/\epsilon^3$ is bounded from above by $\epsilon^4(1+\varepsilon)^j/2$.

We first show that all boxes fit into half of the available slack, $\epsilon^4(1+\varepsilon)^j/4$. The total demand of these boxes is bounded from above by the demand of the small tasks, i.e., $\sum_{j'' < j'} (1+\epsilon)^{j'} = ((1+\epsilon)^{j'+1} - 1)/\epsilon$. An easy calculation shows that this value is smaller than $\epsilon^4(1+\varepsilon)^j/4$.

We now show that the large tasks fit into the remaining slack of $\epsilon^4(1+\varepsilon)^j/4$. We first claim that for each j', $|L^{(j')} \cap T_e| \leq 2/(\mu_2 \varepsilon^4)$. Let e', e'' be the first edge on the left and right of e, respectively, of type at most j'. Each task $i \in L^{(j')} \cap T_e$ has $e' \in P(i)$ or $e'' \in P(i)$ since it is in T_L and $d(i) < \mu_2(1+\varepsilon)^{j'+1}$. By Lemma 32 (property 2), there are at most $1/(\mu_2\varepsilon^4)$ such tasks and therefore the inequality follows. The total demand of the tasks of type $j'' < j'$ is bounded from above by $\sum_{j \leq j'} d(L^{(j)} \cap T_e) \leq \sum_{j \leq j'} 2/(\mu_2\varepsilon^4) \cdot \mu_2(1+\varepsilon)^{j+1} = 2/\varepsilon^5 \cdot ((1+\varepsilon)^{j'+2} - 1$. Again, an easy computation shows that this value is less than $\epsilon^4 f(j)/4$. To summarize,

$$\begin{aligned}
\sum_{B \in \mathcal{B}' :\, e \in P(B)} d(B) + \sum_{i \in T'_L \cap T_e} d(i) = & \sum_{B \in \mathcal{B}'^{(j')} :\, j' \geq j - 1/\varepsilon^3 \text{ and } e \in P(B)} d(B) + \sum_{j' \geq j - 1/\varepsilon^3, i \in T'^{(j')}_L \cap T_e} d(i) \\
& + \sum_{B \in \mathcal{B}'^{(j')} :\, j' < j - 1/\varepsilon^3 \text{ and } e \in P(B)} d(B) + \sum_{j' < j - 1/\varepsilon^3, i \in T'^{(j')}_L \cap T_e} d(i) \\
& \leq u(e) - \varepsilon^4(1+\varepsilon)^j/2 + \varepsilon^4(1+\varepsilon)^j/4 + \varepsilon^4(1+\varepsilon)^j/4 = u(e)\,.
\end{aligned}$$

□

C.3 Computing a Solution from the Player's Strategy Suppose that we are given the optimal strategy for the player (with profit play). We want to use it in order to compute a feasible solution to the given instance of UFP. To this end, we use techniques that are very similar to the ones used in Section 3.2 and Appendix B.

Recall that before starting a subgame, the player defines a set of slices for each task i for which it is possible that i is selected as a small task in this subgame. Each slice has the same path $P(i)$ as i, profit $w(i)/k$ and essentially demand

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

$d(i)/k$, assuming that there are k slices for i. This is analogous to what we did in Section 3, where the slicing is done implicitly by Lemma 2. The player then selects some of these slices for each small task i, plus some large tasks T'_L. In the rest of this section we will show how to select slices for each small task i (and furthermore we will select T'_L). We still need to compute a solution where some small tasks, rather than their slices, are selected. To that aim we interpret the sets of selected slices for each task i as a fractional assignment of small tasks i to boxes. In particular if q such slices are assigned to some box B, then i is assigned to B by an amount of $x_{i,B} = q/k$. This fractional assignment can be converted into a feasible integral solution with a small loss in the profit via standard LP-rounding techniques. Due to this, essentially, it will be sufficient to find a profitable integral solution consisting of large tasks and slices of small tasks. We next will speak about small tasks rather than slices of small tasks for simplicity. In particular, we denote by T_S the set of slices and by T_L the set of input tasks, the intuition being that the player might select them as large tasks.

We can define a strategy DAG similarly to Section 3.1. Similarly as before, for each state of each subgame there is a vertex in this DAG. For vertices corresponding to a state v in which the player assigns a task i into a box, there are two outgoing arcs with probabilities that sum up to one. They model that the size $d'(i)$ is defined randomly. One difference to Section 3.1 is that after a state v the player might start playing some other subgame, starting at some state v_s and afterwards continue playing the current subgame in some state v'. We model this by introducing two arcs (v, v_s), (v, v') that the player traverses for sure, assuming that she arrives at state v at some point. When traversing the strategy DAG, we define that the player first traverses the arc (v, v_s), then continues on v_s, and when the recursion starting in v_s is done she traverses the arc (v, v'). Therefore, playing the game corresponds to defining a tree in the strategy DAG whose vertices are traversed in a fixed order. We define the set $\mathcal{Q}$ to be the set of all these trees where each tree $Q \in \mathcal{Q}$ has a probability $p(Q)$ to be obtained while playing the game. We sample a tree $Q^* \in \mathcal{Q}$ according to these probabilities $\{p(Q)\}_{Q\in\mathcal{Q}}$. Then Q^* corresponds to one execution of the player. In order to keep the notation more similar to Section 3, we will assume w.l.o.g. that at each state v at most one box $B(v)$ is created. This can be enforced by replacing a single state with a path of vertices that are connected via arcs that are all traversed with probability 1.

We define random variables $\bar{X}_i$ as follows:

- For each large task i, we define $\tilde{X}_i := 0$; also, we let $\bar{X}_i = X_i = 1$ if the gambler selects i and $\bar{X}_i = X_i = 0$ otherwise.
- We create each box that the gambler creates.
- We traverse the states of Q^* starting with the root of the strategy DAG. If in some state on Q^* the gambler selects a small task i and assigns i into a box $B = B(v)$ created at some (potentially different) state $v \in Q^*$, then we define $X_{v,i} = 1$. Additionally, if for each $e \in P(B)$ it holds that $d(i) + \sum_{i'\in T_e\cap T_S} \bar{X}_{v,i'} \leq (1+\varepsilon)d(B)$ (where for notational convenience we assume that $\bar{X}_{v,i'} = 0$ if $\bar{X}_{v,i'}$ has not been defined before) then we set $\bar{X}_{v,i} = 1$, otherwise $\bar{X}_{v,i} = 0$.
- Finally, we set $\bar{X}_i = \sum_{v\in Q^*} \bar{X}_{v,i}$, $X_i = \sum_{v\in Q^*} X_{v,i}$, and $\tilde{X}_i := X_i - \bar{X}_i$ for each small task i.

We define $\mathrm{play}_S = \sum_{i\in T_S} w_i\mathbb{E}[X_i]$ and $\overline{\mathrm{play}}_S = \sum_{i\in T_S} w_i\mathbb{E}[\bar{X}_i]$. Like in Section 3.2, we can show that the values $\{\mathbb{E}[X_i]\}_{i\in T_S}$ form a feasible fractional solution for the setting of small tasks only. Hence, using known techniques we can round this fractional solution to an integral solution with essentially no loss of profit.

LEMMA 40. *There is a polynomial time algorithm that computes a feasible integral solution with expected profit at least* $(1 - O(\varepsilon))\mathrm{play}_S$.

Proof. For each small task i let $x_i := \mathbb{E}[X_i]$. We can compute $\{x_i\}_{i\in T_S}$ in polynomial time since for each subgame we can compute the probability that the gambler enters this subgame, and for each small task $i \in T_S$ and each subgame we can compute the probability that the gambler selects i in this subgame, given that we enter the subgame. It holds that $\mathrm{play}_S = \sum_{i\in T_S} w(i)x_i$

Let $e \in E$; we want to argue that $\sum_{i\in T_e\cap T_S} d(i)x_i \leq u(e)$. Our goal is to prove that $\sum_{i\in T_e\cap T_S} d(i)x_i \leq \sum_{\mathcal{B}'} p(\mathcal{B}') \sum_{B\in\mathcal{B}':\, e\in P(B)} d(B)$ where for each set of boxes $\mathcal{B}'$ we denote by $p(\mathcal{B}')$ the probability that the gambler creates exactly the boxes $\mathcal{B}'$. This is sufficient since Lemma 39 implies that

$$\sum_{\mathcal{B}'} p(\mathcal{B}') \sum_{B\in\mathcal{B}':\, e\in P(B)} d(B) \leq \sum_{\mathcal{B}'} p(\mathcal{B}')u(e) = u(e).$$

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

Let $\mathcal{B}$ denote the set of all boxes that the gambler potentially creates while following her strategy. For each box $B \in \mathcal{B}$ denote by $x_{i,B}$ the probability that task i is assigned to B, conditioning on the event that B is created by the gambler. Observe that if i is assigned to some box B (and in particular if $x_{i,B} > 0$) then necessarily $d(i) = \mu_{j(B)}$ where $j(B)$ is the phase in which B is created. For a small task i and a box B created in a phase j let $\hat{x}_{i,B} := x_{i,B} \cdot \frac{\mu_{j(B)}}{\varepsilon' d(B)}$ and observe that $\hat{x}_{i,B}$ denotes the probability that i is selected, assigned to box B in a subgame of phase j, *and* rounded up. Conditioning on the fact that the gambler creates box B, we have that $\sum_{i \in T_e \cap T_S} \hat{x}_{i,B} \leq \frac{1}{\epsilon'}$, since otherwise the rounded demand of the rounded up tasks using e would exceed the capacity $d(B)$. Therefore, we have that

$$\begin{aligned}
\sum_{i \in T_e \cap T_S} d(i) x_i &\leq \sum_{i \in T_e \cap T_S} \sum_{\mathcal{B}'} p(\mathcal{B}') \sum_{B \in \mathcal{B}' :\, e \in P(B)} d(i) x_{i,B} \\
&= \sum_{\mathcal{B}'} p(\mathcal{B}') \sum_{B \in \mathcal{B}' :\, e \in P(B)} \sum_{i \in T_e \cap T_S} d(i) \hat{x}_{i,B} \cdot \frac{\epsilon' d(B)}{\mu_{j(B)}} \\
&= \sum_{\mathcal{B}'} p(\mathcal{B}') \sum_{B \in \mathcal{B}' :\, e \in P(B)} \frac{1}{\epsilon'} \mu_{j(B)} \cdot \frac{\epsilon' d(B)}{\mu_{j(B)}} \\
&= \sum_{\mathcal{B}'} p(\mathcal{B}') \sum_{B \in \mathcal{B}' :\, e \in P(B)} d(B).
\end{aligned}$$

In order to compute a solution with profit play_S we need to work with the tasks in the input, rather than the task slices in T_S. To this end, for each (real) task $i \in T$ let $\mathrm{Sl}(i) \subseteq T_S$ denote the set of small task slices that correspond to i and for each phase j let $\mathrm{Sl}(i,j) \subseteq \mathrm{Sl}(i)$ denote the set of small task slices that correspond to i with demand μ_j. Based on $\{x_i\}_{i \in T_S}$ we define a solution $\{y_i\}_{i \in T'_S}$ where we define $T'_S \subseteq T$ to be all tasks $i \in T$ such that T_S contains a slice corresponding to i. For each $i \in T'_S$ and each phase j we define $y_{i,j} := \frac{1}{|\mathrm{Sl}(i',j)|} \sum_{i' \in \mathrm{Sl}(i',j)} x_{i'}$; based on this we define $y_i := \sum_j y_{i,j}$. We argue that $y_i \leq 1$. We have that

$$\begin{aligned}
y_i &= \sum_j y_{i,j} \\
&= \sum_j \frac{1}{|\mathrm{Sl}(i,j)|} \sum_{i' \in \mathrm{Sl}(i,j)} x_{i'} \\
&= \sum_j \frac{1}{|\mathrm{Sl}(i,j)|} \sum_{i' \in \mathrm{Sl}(i,j)} \mathbb{E}[X_{i'}] \\
&= \sum_j \Pr[i \text{ is of type } j] \frac{1}{|\mathrm{Sl}(i,j)|} \sum_{i' \in \mathrm{Sl}(i,j)} \mathbb{E}[X_{i'} \mid i \text{ is of type } j] \\
&\leq \sum_j \Pr[i \text{ is of type } j] \frac{1}{|\mathrm{Sl}(i,j)|} |\mathrm{Sl}(i,j)| \\
&\leq 1 \,.
\end{aligned}$$

where $\Pr[i \,\text{is of type}\, j]$ refers to the probability that according to the edge types defined by the player, i is of type j, i.e., some edge of $P(i)$ is of type j and there is no edge in $P(i)$ of some type $j' < j$.

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

Moreover, we have that

$$\begin{aligned} d(i)y_i &= d(i)\sum_j y_{i,j} \\ &= d(i)\sum_j \frac{1}{|\mathrm{Sl}(i,j)|}\sum_{i'\in\mathrm{Sl}(i',j)} x_{i'} \\ &= \sum_j \frac{d(i)}{\lceil d(i)/\mu_j\rceil}\sum_{i'\in\mathrm{Sl}(i',j)} x_{i'} \\ &\le \sum_j \mu_j \sum_{i'\in\mathrm{Sl}(i,j)} x_{i'} \\ &= \sum_j \sum_{i'\in\mathrm{Sl}(i,j)} d(i')x_{i'} \\ &= \sum_{i'\in\mathrm{Sl}(i)} d(i')x_{i'} \end{aligned}$$

which implies that $\sum_{i\in T_e\cap T'_S} d(i)y_i \le u(e)$ for each $e\in E$. We conclude that the variables y_i form a feasible solution to the canonical LP for UFP below.

$$\begin{aligned} \max \sum_{i\in T'_S} y_i w(i) & \\ \sum_{i\in T_e\cap T'_S} d(i)y_i \le u(e) & \qquad \forall e\in E \\ y_i \le 1 & \qquad \forall i\in T'_S \\ y_i \ge 0 & \qquad \forall i\in T'_S \end{aligned}$$

Note that if $y_i > 0$ then the gambler assigns a slice of i into some box with in some subgame of some phase j, which implies that $d(i) \le \mu_1(1+\epsilon)^j \le \epsilon^2 u(e)$ for each edge $e\in P(i)$ (i.e., the task i is relatively small). Using an algorithm in [CMS07] we can round $\{y_i\}_{i\in T'_S}$ to an integral solution whose profit is at least $(1-O(\epsilon))\sum_{i\in T'_S} w(i)y_i$. ∎

Also, similarly to Section 3.2 we can compute an integral solution with profit essentially $\mathrm{play}_L + \overline{\mathrm{play}}_S$: we compute the solution $\{\bar{X}_i\}_{i\in T}$ by simulating the gambler and at the end in each box B we remove some tasks assigned to B such that the remaining tasks really fit into B (while losing only a factor $1+O(\varepsilon)$ in the profit). Afterwards, we transform the solution with the slices to a solution for the (real) input tasks via LP-rounding (which loses another factor $1+O(\epsilon)$).

LEMMA 41. *There is a polynomial-time randomized algorithm that computes a feasible boxed solution of expected profit at least* $\mathrm{play}_L + (1-O(\varepsilon))\overline{\mathrm{play}}_S$.

Proof. We simulate the gambler and compute the solution $\{\bar{X}_i\}_i$ as described above. Let T'_L denote the tasks that the gambler selected as large tasks. The excepted profit of this solution is $\mathrm{play}_L + \overline{\mathrm{play}}_S$. For each state $v\in Q^*$ in which a box $B = B(v)$ is created and each edge $e\in P(B)$ we have that $\sum_{i\in T_e\cap T_S}\bar{X}_{v,i} \le (1+\varepsilon)d(B)$. We apply Lemma 17 to the tasks i with $\bar{X}_{v,i}=1$ and obtain a set of tasks that fits into B with profit at least $(1-O(\varepsilon))\sum_{i\in T_e} w(i)\bar{X}_{v,i}$. Due to Lemma 39 this yields a boxed solution for the slices with profit at least $\mathrm{play}_L + (1-O(\varepsilon))\overline{\mathrm{play}}_S$.

We transform it to an integral solution for the (real) input tasks in T. Let T'_S denote the set of tasks $i\in T$ such that there is a slice $i'\in T_S$ corresponding to i with $\bar{X}_{i'}=1$. For each $i\in T'_S$ let $\mathrm{Sl}(i)$ denote the set of slices corresponding to i with $\bar{X}_{i'}=1$. Also note that $i\notin T'_L$. We define $y_i := \frac{1}{|\mathrm{Sl}(i)|}\sum_{i'\in\mathrm{Sl}(i)}\bar{X}_{i'}$. Since $\bar{X}_{i'}\le 1$ by definition for each slice i' we obtain that $y_i\le 1$. Also, we have that $w(i) = \sum_{i'\in\mathrm{Sl}(i)} w(i')$, using that there is only one phase j in which slices corresponding to i are selected by $\bar{X}$. We want to round the solution $\{y_i\}_{i\in T'_S}$ to an integral solution while losing only a factor $1+\epsilon$ in the profit. To this end, we observe that $\{y_i\}_{i\in T'_S}$ is a feasible solution to an LP for the given UFP instance in which we assume that the tasks in T'_L are already taken and their demand is subtracted from the edge capacities:

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

$$\max \sum_{i \in T \setminus T'_L} y_i w(i)$$

$$\sum_{i \in T_e} y_i d(i) \leq u(e) - \sum_{i \in T'_L \cap T_e} d(i) \qquad \forall e \in E$$

$$y_i \leq 1 \qquad \forall i \in T'_S$$

$$y_i \geq 0 \qquad \forall i \in T'_S$$

We remark that if $y_i > 0$ then for each edge $e \in P(i)$ it holds that

$$d(i) \leq \mu_1(1+\epsilon)^j \leq \frac{\epsilon^4}{(1+\epsilon)^{1/\epsilon^3}}(1+\epsilon)^j \leq \lfloor \epsilon^{31}(1+\epsilon)^j \rfloor \leq u(e) - \sum_{i \in T'_L \cap T_e} d(i)$$

due to Lemma 39 and the fact that if $y_i > 0$ then the gambler created at least one box of type j that uses edge e. Therefore, using an algorithm in [CMS07] in polynomial time we can compute an integral solution to the above LP with profit at least $(1 - O(\epsilon)) \sum_{i \in T \setminus T'_L} w(i) y_i$. Together with the tasks in T'_L it yields an integral solution with the claimed profit. ∎

Next, we define fractional solutions $\{X'_i\}_{i \in T_S}$, $\left\{\bar{X}'_i\right\}_{i \in T_S}$, $\left\{\tilde{X}'_i\right\}_{i \in T_S}$ in a very similar way as in Appendix B. For each box $B(v)$ created in some state v, we define $\tilde{x}_{v,i}$ as the probability that the gambler assigns task i to box $B(v)$ at v or at one node visited after v and $\tilde{X}_i = 1$, *conditioning on the event that* $v \in Q^*$. Similarly, we define $\bar{x}_{v,i}$ to be the probability that the gambler assigns task i to box $B(v)$ at v or at a node visited after v and that $\bar{X}_i = 1$, conditioning on the event $v \in Q^*$. We define $x_{v,i} = \bar{x}_{v,i} + \tilde{x}_{v,i}$. Now we define random variables X'_i, $\bar{X}'_i$, and $\tilde{X}'_i$ based on $x_{v,i}$, $\bar{x}_{v,i}$, and $\tilde{x}_{v,i}$, respectively. For each small task i we define $X'_i := \min\left\{\sum_{v \in Q^*} x_{v,i}, 1\right\}$, $\bar{X}'_i := \min\left\{\sum_{v \in Q^*} \bar{x}_{v,i}, 1\right\}$ and $\tilde{X}'_i := \min\left\{\sum_{v \in Q^*} \tilde{x}_{v,i}, 1\right\}$. Like previously, it will turn out that always $\tilde{X}'_i \leq \varepsilon$ and therefore always $\tilde{X}'_i = \sum_{v \in Q^*} \tilde{x}_{v,i}$.

LEMMA 42. *Given a path Q^* in the strategy DAG, the values $\{x_{v,i}, \bar{x}_{v,i}, \tilde{x}_{v,i}\}_{i \in T, v \in \mathcal{Q}^*}$ can be computed in polynomial time. Hence in polynomial time one can also compute the value of X'_i for each task i.*

Proof. Let v be a state in a subgame in which a box $B(v)$ is created by the gambler and let i be a task. It might be possible that after state v the gambler recurses in some other subgame; however, in this subgame she cannot assign any task to box $B(v)$. Therefore, using a similar reasoning as in Lemma 9 we can compute the values $\{x_{v,i}, \bar{x}_{v,i}, \tilde{x}_{v,i}\}_{i \in T, v \in \mathcal{Q}^*}$ and thus also the value X'_i for each task i. ∎

Our third solution is given by the solution $\{X'_i\}_{i \in T}$.

LEMMA 43. *In polynomial time we can compute a feasible integral solution with expected profit at least $\sum_{i \in T_L} w(i)\mathbb{E}[X'_i] + (1 - O(\varepsilon)) \sum_{i \in T_S} w(i)\mathbb{E}[X'_i]$.*

Proof. We simulate the gambler and define the variables X'_i as described above. The analogue of Lemma 19 holds for each box $B(v)$. We transform it now to an integral solution, using a very similar argumentation as in the proof of Lemma 41. Again, let T'_L denote the set of tasks that are selected as large tasks by the gambler. Also, similarly as before let T'_S denote the set of tasks $i \in T$ such that there is a slice $i' \in T_S$ corresponding to i with $X'_{i'} > 0$. Let $i \in T'_S$. Again we observe that $i \notin T'_L$. Let $\mathrm{Sl}(i)$ denote the set of slices corresponding to i with $X'_{i'} > 0$. We define $y_i := \frac{1}{|\mathrm{Sl}(i)|} \sum_{i' \in \mathrm{Sl}(i)} X'_{i'}$. Since $X'_{i'} \leq 1$ by definition for each slice i' we obtain that $y_i \leq 1$. Also, we have that $w(i) = \sum_{i' \in \mathrm{Sl}(i)} w(i')$, using that there is only one phase j in which slices corresponding to i are (fractionally) selected by X'. We want to round the solution $\{y_i\}_{i \in T'_S}$ to an integral solution while losing only a factor $1+\epsilon$ in the profit via LP-rounding, invoking an algorithm in [CMS07]. The remainder of our reasoning is identical to the corresponding part of the proof of Lemma 41. ∎

We want to show that the solution due to Lemma 43 yields large profit. To this end, we want to prove the following lemma.

LEMMA 44. *For each task $i \in T_S$ it holds that $(1 + O(\varepsilon))\mathbb{E}[X'_i] \geq (1 - \frac{1}{e})\mathbb{E}[\bar{X}_i] + \mathbb{E}[\tilde{X}_i]$.*

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited

In the remainder of this section we prove Lemma 44. For doing this, we follow a very similar argumentation as used in Appendix B.

LEMMA 45. *For a sufficiently small constant $\varepsilon' > 0$ depending on ε, $\tilde{X}'_i \leq \varepsilon$.*

Proof. Consider a state $v \in Q^*$ and let $B = B(v)$ be the box created in state v. Assume that B is of type j. First, we show that $\tilde{x}_{v,i} \leq \varepsilon^{26}$. To this end, let us consider the random set of tasks S_B that the gambler assigns to box $B = B(v)$ before task i is considered and that use the leftmost edge of i. Notice that, for task i to be discarded, it must happen that $d(i) + d(S_B) > (1+\varepsilon)d(B)$. Let also S_B^{up} be the tasks in S_B that are rounded up. Notice that deterministically $d'(S_B^{up}) \leq d(B)$. Hence an upper bound on $\tilde{x}_{v,i}$ is given by the probability of the event that $d'(S_B^{up}) \leq d(B)$ under the constraint that $d(i) + d(S_B) > (1+\varepsilon)d(B)$ and thus $1 + |S_B| > (1+\varepsilon)\frac{d(B)}{\mu_j}$. Observe however that, under the same constraint, $\mathbb{E}[|S_B^{up}|] = p_{up}|S_B| = \frac{\mu_j}{\epsilon' \cdot d(B)}|S_B| = \frac{\mu_j}{\epsilon' \cdot d(B)}((1+\epsilon)\frac{d(B)}{\mu_j} - 1) \geq \frac{\mu_j}{\epsilon' \cdot d(B)}((1+\epsilon/2)\frac{d(B)}{\mu_j}) = \frac{1+\epsilon/2}{\epsilon'}$. Hence we can use Chernoff's bound to conclude

$$\tilde{x}_{v,i} \leq \Pr[|S_B^{up}| \leq \frac{1}{\varepsilon'} \mid \mathbb{E}[|S_B^{up}|] \geq \frac{1+\varepsilon/2}{\varepsilon'}] \leq e^{-\frac{1+\varepsilon/2}{2\varepsilon'}(1-\frac{1}{1+\varepsilon/2})^2} = e^{-\frac{\varepsilon^2}{8(1+\varepsilon)\varepsilon'}} \leq \epsilon^{26}$$

if ϵ' is sufficiently small. There are at most $\frac{1}{\varepsilon^{31}}$ boxes that are created in states of Q^* that contain $P(i)$ (since each edge can be used by at most $\frac{1}{\varepsilon^{31}}$ boxes of each type using each edge e). Therefore, $\tilde{X}'_i = \sum_{v \in Q^*} \tilde{x}_{v,i} \leq \varepsilon^{26} \cdot |\{v \in Q^* : P(i) \subseteq P(B(v))\}| \leq \varepsilon$. ∎

Analogously to Lemma 29 in Appendix B, essentially $X'_i = \bar{X}'_i + \tilde{X}'_i$.

LEMMA 46. *Let i be a small task. It holds that $(1 - O(\varepsilon))(\bar{X}'_i + \tilde{X}'_i) \leq X'_i \leq \bar{X}'_i + \tilde{X}'_i$.*

Proof. This is proof is identical to the proof of Lemma 29, where we use Lemma 45 instead of Lemma 28 to upper bound $\tilde{X}'_i$. ∎

Due to Lemma 46, for proving Lemma 44 it is sufficient to show that $\mathbb{E}[\tilde{X}'_i] = \mathbb{E}[\tilde{X}_i]$ and $\mathbb{E}[\bar{X}'_i] \geq (1 - \frac{1}{e})\mathbb{E}[\bar{X}_i]$.

LEMMA 47. *Let i be a small task. We have that $\mathbb{E}[\tilde{X}'_i] = \mathbb{E}[\tilde{X}_i]$.*

Proof. This proof is identical to the proof of Lemma 30. ∎

It remains to show that $\mathbb{E}[\bar{X}'_i] \geq (1 - \frac{1}{e})\mathbb{E}[\bar{X}_i]$.

LEMMA 48. *Let i be a small task. We have that $\mathbb{E}[\bar{X}'_i] \geq (1 - \frac{1}{e})\mathbb{E}[\bar{X}_i]$.*

Proof. This proof is essentially identical to the proof of Lemma 31, which in turn is analogous to the proof of Lemma 12. The only difference is that the prefix tree T is now defined via the trees in $\mathcal{Q}$. More precisely, for each tree $Q \in \mathcal{Q}$ we consider the order in which the vertices of Q are visited which yields a vector in which each entry is a state of some subgame. Then we define T via these vectors and the remainder of the argumentation stays the same. ∎

Copyright © 20XX by SIAM
Unauthorized reproduction of this article is prohibited