Submodular Unsplittable Flow on Trees*

Anna Adamaszek[1], Parinya Chalermsook[2], Alina Ene[3], and Andreas Wiese[2]

[1] University of Copenhagen, Denmark, `anad@di.ku.dk`
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany, `{parinya,awiese}@mpi-inf.mpg.de`
[3] University of Warwick, United Kingdom, `a.ene@dcs.warwick.ac.uk`

Abstract. We study the Unsplittable Flow problem (UFP) on trees with a submodular objective function. The input to this problem is a tree with edge capacities and a collection of tasks, each characterized by a source node, a sink node, and a demand. A subset of the tasks is feasible if the tasks can simultaneously send their demands from the source to the sink without violating the edge capacities. The goal is to select a feasible subset of the tasks that maximizes a submodular objective function.
Our main result is an $O(k \log n)$-approximation algorithm for Submodular UFP on trees where k denotes the pathwidth of the given tree. Since every tree has pathwidth $O(\log n)$, we obtain an $O(\log^2 n)$ approximation for arbitrary trees. This is the first non-trivial approximation guarantee for the problem and it matches the best approximation known for UFP on trees with a linear objective function.
Our main technical contribution is a new geometric relaxation for UFP on trees that builds on the recent work of [Bonsma et al., FOCS 2011; Anagnostopoulos et al., SODA 2014] for UFP on paths with a linear objective. Our relaxation is very structured and we can combine it with the contention resolution framework of [Chekuri et al., STOC 2011]. Our approach is robust and extends to several related problems, such as UFP with bag constraints and the Storage Allocation Problem.
Additionally, we study the special case of UFP on trees with a linear objective and upward instances where, for each task, the source node is a descendant of the sink node. Such instances generalize UFP on paths. We build on the work of [Bansal et al., STOC 2006] for UFP on paths and obtain a QPTAS for upward instances when the input data is quasi-polynomially bounded. We complement this result by showing that, unlike the path setting, upward instances are APX-hard if the input data is arbitrary.

1 Introduction

Submodular functions are a rich class of functions with many applications both in theory and in practice. On the theoretical side, submodularity is a key concept

* Partially supported by the Danish Council for Independent Research DFF-MOBILEX mobility grant.

in combinatorial optimization and economics with deep mathematical consequences. On the practical side, submodular functions arise naturally in a variety of settings such as data summarization, sensor placement, inference in graphical models, image segmentation, social networks, auctions, and exemplar clustering [22, 20, 21, 19, 6, 16, 17, 14].

One of the main reasons for the success of submodularity is that it combines a significant modeling power with a certain degree of tractability. This delicate balance between generality and tractability has made submodular functions very appealing, and there has been a significant interest in optimizing submodular functions subject to a variety of constraints.

The traditional approach to submodular maximization makes extensive use of the classical Greedy algorithm of Nemhauser, Wolsey, and Fisher [23]. The Greedy algorithm and its continuous counterparts are well-suited for constraints such as cardinality, matroids, and knapsack, but they fail to handle other types of natural constraints. Thus there is an increasing need to develop algorithms for general constraints.

A major contribution in this direction comes from the work of Chekuri *et al.* [13] which has developed a powerful framework for submodular function maximization with general constraints. Their framework leverages the power of mathematical programming relaxations coupled with structured rounding schemes called *contention resolution (CR)* schemes. In particular, it unifies several previous results for special cases (e.g., matroids or knapsack constraints) and thus captures the types of constraints for which we know how to optimize submodular functions. This has led to the following very interesting meta-question: *For which type of constraints can we provide structured relaxations that admit good CR schemes?* In this paper, we address this question in the specific case of the unsplittable flow problem (UFP). In this setting, we are given an edge-capacitated, undirected graph and a collection of tasks; each task is specified by a source vertex, a sink vertex, and a demand. The goal is to select a maximum profit subset of the tasks that can be routed unsplittably, i.e., the task's demand is routed along a single path from the source to the sink subject to the edge capacities.

The problem is well-studied, and most of the results focus on linear objectives. Despite its apparent simplicity, already UFP on paths captures several well-studied problems, including the knapsack problem (when the graph is a single edge) and resource allocation problems. UFP is quite challenging even on paths and trees, and one of the main reasons for the difficulty is the lack of LP relaxations with small integrality gaps. The natural LP relaxation for the problem has an $\Omega(n)$ integrality gap even on paths [9], and standard approaches for strengthening the LP by adding valid inequalities fail to improve the integrality gap significantly [12]. Chekuri *et al.* [12] gave a novel LP relaxation for UFP on paths that strengthens the standard LP using clique type of constraints, and they showed that it has an $O(\log n)$ integrality gap. The relaxation of [12] can also be extended to trees, and understanding this relaxation has been an interesting and challenging open question[4].

The design of good relaxations for UFP is motivated not only by the goal of obtaining better approximations for linear objectives, but also by the need of handling more general constraints and objective functions. In particular, the current approaches for submodular objectives rely on structured relaxations with good CR schemes. As a result, there is a discrepancy between the approximation guarantees for linear and submodular objectives. There has been a long line of work for UFP on paths with a linear objective that led to a constant factor approximation [5, 4]; these approaches combine the standard LP relaxation with dynamic programming techniques. Chekuri *et al.* [12] give a combinatorial greedy algorithm for UFP on trees with a linear objective that achieves an $O(\log^2 n)$ approximation. In contrast, for UFP with a submodular objective, only an $O(\log n)$ approximation is known for paths and *no non-trivial approximation was known for trees prior to our work.* Chekuri *et al.* [13] consider instances of submodular UFP on trees that satisfy a certain assumption, called the no-bottleneck assumption (NBA)[5], and they give a constant factor approximation for such instances. However, the no-bottleneck assumption is very restrictive and removing this restriction poses several technical challenges, particularly for the design of mathematical programming relaxations.

Thus, there has been an extensive work on UFP on paths but relatively fewer results on trees. Since UFP models the allocation of communication bandwidths in networks, we believe that it is worthwhile to develop a better understanding for more complex network topologies, such as trees. Also, submodular objective functions are much richer than linear objectives, and can model for instance linear objective functions with additional constraints.

Our contributions. We give the first approximation algorithm for submodular UFP on trees and the first relaxation with a matching integrality gap. Our algorithm achieves an approximation ratio of $O(k \log n)$ on trees with pathwidth k. As each tree has pathwidth $O(\log n)$, this gives an $O(\log^2 n)$-approximation for arbitrary trees, matching the best known result for linear objective functions [12]. For several special cases of the problem, such as paths, spiders, and caterpillars, our approximation ratio improves to $O(\log n)$ (since in those cases $k = O(1)$), and such a ratio was not even known for linear objectives. Thus our result generalizes and improves the best approximations known for UFP on paths with a submodular objective and UFP on trees with a linear objective.

Theorem 1. *There is a $O(k \cdot \log n)$ approximation for Submodular* UFP *on trees, where k is the pathwidth of the tree and n is the number of nodes in the tree. Additionally, there is a polynomial-sized relaxation for the problem with a matching integrality gap.*

[4] Friggstad and Zao [15] showed an $O(\log^2 n)$ upper bound on the integrality gap of the relaxation of [12] for UFP on trees with a linear objective. This upper bound is shown via a primal-dual analysis which is not suitable for designing a CR scheme.

[5] The no-bottleneck assumption states that the maximum demand of any task is at most the minimum capacity of any edge.

We obtain our result via a new geometric LP relaxation for UFP on trees that is very different from the clique-based approach of [12]. Our relaxation builds on a powerful two-dimensional geometric viewpoint developed in the context of the UFP problem on paths with a linear objective [5]. This viewpoint connects UFP to structured instances of the Maximum Independent Set of Rectangles (MISR) problem [1, 10, 11], which in turn allows one to handle instances of UFP on paths for which the standard LP relaxation fails. The geometry was exploited to obtain a combinatorial algorithm for such instances that is based on dynamic programming. A related two-dimensional visualization was used in [2], again as the basis of a dynamic program. These approaches, however, break down for submodular UFP on trees; in the two-dimensional viewpoints, an input path corresponds to a subinterval of the x-axis and this is no longer meaningful for trees. Also, dynamic programming approaches are not suitable for submodular objective functions. In contrast to previous work, the focus in this paper is to translate these geometric insights to an LP relaxation for UFP on trees. We give a CR scheme for our relaxation that can be combined with the framework of [13] to obtain approximation guarantees for submodular objectives. The core of our reasoning is that our LP-formulation not only decides which tasks to select, but also computes a drawing of them as non-overlapping rectangles on suitable subpaths of the tree. We remark that our LP is a polynomial-sized extended formulation and, to the best of our knowledge, this is the first time that an extended formulation is used in the context of CR schemes.

A very important feature of the CR scheme framework is that it allows one to combine several constraints, thus extending the applicability of our approach to two generalized settings. First, in the Submodular Bag-UFP on trees problem, the input tasks are partitioned into bags and a feasible solution is allowed to select at most one task per bag [8][6]. We obtain an $O(k \log n)$ approximation for Submodular Bag-UFP on trees of pathwidth k. Second, we obtain an $O(\log n)$ approximation for the Submodular Storage Allocation Problem on trees. This problem has the same input as UFP, with additional requirements that each selected task gets a private subinterval of width equal to the demand, contained in $[0, u_e)$ for each edge e used by the task. We require that these subintervals are disjoint for any two tasks sharing an edge of the tree. Intuitively, this models that each task gets a contiguous portion of the resource spectrum.

Finally, we round up our contributions with the following results for a special case of UFP on trees with a linear objective function. An instance of UFP on tree is an *upward instance* if the input tree is rooted and, for every task, the source node of the task is an ancestor of the sink node (or vice-versa).

Theorem 2. *There is a* $(1+\epsilon)$ *approximation algorithm for upward instances of* UFP *on trees with running time* $n^{\text{poly}(\log(n/\epsilon))\log(d_{\max}/d_{\min})}$. *In particular, if the demands are quasi-polynomially bounded, this gives a* QPTAS.

[6] For linear objective functions, the bag constraints can be "glued" with the objective function, yielding an instance of Submodular UFP. It is not clear though whether this holds in general for any initial submodular objective function.

Unlike for UFP on paths [4], we show that the dependency of the running time on the term $\log(d_{\max}/d_{\min})$ can *not* be removed for upward instances of UFP on trees. In fact, assuming the *Exponential Time Hypothesis (ETH)*, the running time of our approximation scheme is essentially tight. This illustrates an inherent distinction between paths and upward instances on trees. Also, it shows that this is one of the very rare problems that allows a QPTAS on quasi-polynomially bounded input data but becomes APX-hard on general instances.

Theorem 3. *There is a universal constant ϵ_0 such that for all $\delta > 0$ any $(1+\epsilon_0)$-approximation algorithm for upward instances of* UFP *on trees runs in time of at least $n^{\mathrm{poly}(\log n)\log^{1-\delta}(d_{\max}/d_{\min})}$, unless ETH fails. Also, the problem is* APX-*hard.*

Other related work. The problem of maximizing submodular functions subject to various constraints is very well-studied and several results are known; we refer the reader to [13] for an overview. UFP with a linear objective is also extensively studied. Due to space limitation, we omit a detailed discussion of these results. The best approximation is a $(2+\varepsilon)$ approximation [2] and a QPTAS [3] for UFP on paths, and an $O(\log^2 n)$ approximation for UFP on trees [12].

Formal problem definitions. We consider the Unsplittable Flow problem on trees (UFP-tree). The input consists of an undirected tree $T = (V, E)$ with edge capacities $u_e \in \mathbb{Z}_+$, and a set of tasks $\mathcal{T}$. Each task $i \in \mathcal{T}$ is characterized by a start vertex $s_i \in V$, an end vertex $t_i \in V$, a demand $d_i \in \mathbb{Z}_+$, and a profit $w_i \in \mathbb{Z}_+$. For each task $i \in \mathcal{T}$ denote by p_i the unique path between s_i and t_i in T. A feasible solution is a subset of the tasks $\mathcal{T}' \subseteq \mathcal{T}$ satisfying the capacity constraints $\sum_{i\in\mathcal{T}':\, p_i\ni e} d_i \le u_e$ for each edge $e \in E$. The goal is to find a feasible solution maximizing $w(\mathcal{T}') := \sum_{i\in\mathcal{T}'} w_i$.

The Submodular UFP-tree problem is a generalization of UFP-tree, where instead of a linear weight function w we are given a submodular objective function $f : 2^{\mathcal{T}} \to \mathbb{R}_+$ and the goal is to select a feasible subset $\mathcal{T}' \subseteq \mathcal{T}$ maximizing $f(\mathcal{T}')$. A function $f : 2^{\mathcal{T}} \to \mathbb{R}_+$ is *submodular* if $f(A) + f(B) \ge f(A \cap B) + f(A \cup B)$ for any two subsets $A, B \subseteq \mathcal{T}$. We assume that f is given as a value oracle, i.e., we are given access to an oracle that takes as input any set S and outputs $f(S)$.

In the Bag-UFP-tree problem, in addition to the input of UFP-tree, the input tasks are partitioned into sets called *bags* and we are allowed to select at most one task from each bag. We also consider the Storage Allocation Problem (SAP-tree). The input to SAP-tree is the same as for UFP-tree, with additional requirement that for each selected task i in $\mathcal{T}' \subseteq \mathcal{T}$ we have to compute a value $h(i) \ge 0$ such that $h(i)+d_i \le u_e$ for each edge $e \in p_i$, and $[h(i), h(i)+d_i)\cap[h(i'), h(i')+d_{i'}) = \emptyset$ for any two tasks $i, i' \in \mathcal{T}'$ with $p_i \cap p_{i'} \neq \emptyset$. This corresponds to giving each task $i \in \mathcal{T}'$ the portion $[h(i), h(i) + d_i)$ of the resource spectrum.

2 Geometric Relaxation for Submodular UFP on Trees

In this section, we present our $O(k \cdot \log n)$ approximation algorithm for Submodular UFP-tree. We first describe a pseudo-polynomial sized LP relaxation for

UFP-tree with a linear objective function. In Section 2.2, we show how to reduce the size of the LP to polynomial. In Section 2.3, we extend our algorithm to a submodular objective function. We defer the description of our results for the Submodular Bag-UFP and SAP problems to the full version of this paper.

2.1 A Pseudo-polynomial Sized Relaxation

In the following, we give a geometric LP-relaxation for UFP-tree with a linear objective function. The relaxation has pseudo-polynomial size.

Reduction to intersecting instances. First, we reduce the general case to the case in which the path of each task contains the root of the tree. We call such instances *intersecting instances.* Chekuri *et al.* [12] showed that, via a standard centroid decomposition, we can reduce an arbitrary instance to a collection of intersecting instances at a loss of $O(\log n)$ in the approximation ratio.

Lemma 1 (Chekuri *et al.* [12]). *Suppose that there is a polynomial time algorithm for* UFP-tree *that achieves an α-approximation on intersecting instances. Then there is a polynomial time $O(\alpha \cdot \log n)$ approximation algorithm for the problem on arbitrary trees. Moreover, this holds for the generalization of the problem in which the objective function is sub-additive*[7].

Partitioning into paths. In the remainder of this section, we assume that we are given an intersecting instance on a tree T of pathwidth k. Intuitively, a graph has pathwidth k if it has a tree-decomposition of width k in which the tree describing the decomposition is a path (see e.g. [18] for a formal definition). Note that every tree has pathwidth at most $O(\log n)$ [18]. Our goal is to compute a $O(k)$-approximation for such instances, so that using Lemma 1 we obtain a $O(k \log n)$-approximation for the general problem. First, we split a given tree into a collection $\mathcal{P}$ of paths such that each input task shares an edge with at most $O(k)$ paths in $\mathcal{P}$. Then, we define a new LP relaxation for the problem with a randomized rounding with alteration strategy. The relaxation will be based on a two-dimensional geometric viewpoint for each path in $\mathcal{P}$.

For our path partition $\mathcal{P}$ we require that each path $P \in \mathcal{P}$ is an *upward path*, i.e., one endpoint of the path is an ancestor in T of the other endpoint. The following observation follows from the property of an intersecting instance.

Observation 1 *For each task i and each upward path P, if i uses an edge of P then it uses the top edge of P.*

Definition 1. *Consider an intersecting instance of* UFP-tree *on a rooted tree T. Let $\mathcal{P} = \{P_1, \dots, P_\ell\}$ be a collection of paths in T. We say that $\mathcal{P}$ is a K-*nice splitting *if it has the following properties:*

– *The paths in $\mathcal{P}$ are edge-disjoint, upward paths, partitioning the edges of T.*

[7] A set function $f : 2^V \to \mathbb{R}$ is sub-additive if $f(A \cup B) \leq f(A) + f(B)$ for any two disjoint sets A and B. Note that a non-negative submodular function is sub-additive.

– *Each task uses an edge of at most K paths in $\mathcal{P}$.*

The next lemma shows the existence of a $O(k)$-nice splitting where k is the pathwidth of T.

Lemma 2. *Consider an intersecting instance I of* UFP-tree *on a rooted tree T of pathwidth k. There is a polynomial time algorithm that constructs an $O(k)$-nice splitting for I.*

Geometric viewpoint. Let $\mathcal{P} = \{P_1, \ldots, P_\ell\}$ be an $O(k)$-nice splitting of the instance that is guaranteed by Lemma 2. We use $\mathcal{P}$ to write an LP relaxation for the problem, based on the following geometric viewpoint. For a path $P \in \mathcal{P}$, let $\mathcal{T}_P$ be the set of tasks from $\mathcal{T}$ using an edge of P.

If we restrict the tasks in our instance to a path P in T, we get an instance of the Unsplittable Flow problem on paths (UFP-path) in a natural way. For each task $i \in \mathcal{T}_P$, the UFP-path instance has a corresponding task whose path is $p_i \cap P$. Notice that each task $i \in \mathcal{T}_P$ uses the top edge of P, so we can assume w.l.o.g. that when traversing the edges of P from top to bottom, their capacities are non-increasing. We call such an instance a *one-sided staircase* instance.

We claim that for such an instance of UFP-path on a path P, each feasible subset of the tasks can be represented as a collection of non-overlapping rectangles drawn underneath the capacity profile, such that each task i has a corresponding rectangle of height d_i whose projection on P is the path of i. We interpret these rectangles as open sets. We call such a drawing a *representing drawing.*

Lemma 3. *Consider an instance of* UFP-path *on a path P in which all of the tasks use the first edge of P. Any feasible subset of the tasks admits a representing drawing.*

LP relaxation. Using this geometric viewpoint, we write an LP relaxation for intersecting instances of UFP-tree as follows. Recall that we have an $O(k)$-nice splitting $\mathcal{P}$ of the tree T. We add constraints to the relaxation to enforce that there is a representing drawing for the selected tasks on each path $P \in \mathcal{P}$; we remark that these constraints will automatically enforce the capacity constraints.

Variables. The IP has the following variables. For each task i, we have a variable $x_i \in \{0, 1\}$ with the interpretation that $x_i = 1$ if task i is in the solution. For each path $P \in \mathcal{P}$, each task $i \in \mathcal{T}_P$, and each height h, we have a variable $y(i, h, P) \in \{0, 1\}$ with the interpretation that $y(i, h, P) = 1$ if the rectangle for task i is drawn at height h in the representing drawing for P. The allowed heights h are the ones satisfying $h + d_i \leq u_e$ for each edge $e \in p_i \cap P$, i.e., such that the rectangle fits under the capacity profile. We introduce variables $y(i, h, P)$ only for such heights.

Constraints. For each path $P \in \mathcal{P}$ and each task $i \in \mathcal{T}_P$, we have a constraint

$$\sum_{h \text{ s.t. } \forall e \in p_i \cap P:\, h + d_i \leq u_e} y(i, h, P) = x_i \quad . \tag{1}$$

For each path $P \in \mathcal{P}$, we add constraints enforcing that in the representing drawing for P the rectangles do not overlap. This is achieved by imposing constraints modeling that any point q underneath the capacity profile is covered by at most one rectangle. Since all tasks use the first edge of P, it suffices to consider only points q on a vertical line going through the first edge of P, i.e., points $q = (x_0, h)$ where x_0 is an arbitrary x-coordinate strictly between the first and the second vertex of P and h is an integral height that is at most the capacity of the first edge of P. We use $R(i, h, P)$ to denote a rectangle representing task i on P drawn at height h, i.e., $R(i, h, P)$ is a rectangle of height d_i, with a bottom y-coordinate h, and whose projection on the x-axis equal $p_i \cap P$.

For each path $P \in \mathcal{P}$ and each point $q = (x_0, h)$ as described above we have a constraint

$$\sum_{i \in \mathcal{T}_P} \sum_{(h' :\, q \in R(i,h',P))} y(i, h', P) \leq 1. \tag{2}$$

We refer to the resulting LP relaxation as Rectangle-LP($\mathcal{P}$). It clearly has pseudo-polynomial complexity. In the following, we show an $O(k)$-approximation based on LP rounding.

Rounding. Let (x, y) be a feasible solution to Rectangle-LP($\mathcal{P}$). We use a randomized rounding with alteration strategy (as introduced in [7] to select a subset of the tasks and a representing drawing for them on each path $P \in \mathcal{P}$. We proceed in two phases. In the selection phase, we pick a subset of the tasks and determine a drawing for them. The drawing in this phase may contain overlapping rectangles. In the alteration phase, we pick a subset of the selected tasks whose corresponding rectangles do not overlap.

Selection phase. We select a (not necessarily feasible) set S of tasks. For each task i, we add i to S independently at random with probability $x_i/(c_1 \cdot k)$, where $c_1 > 1$ is a sufficiently large constant that will be determined later. We refer to the tasks in the random sample S as the *selected* tasks. Additionally, for each task $i \in S$ and each path $P \in \mathcal{P}$ such that $i \in \mathcal{T}_P$, we choose a rectangle representing the drawing of i on P, as follows. We choose a height h for the rectangle independently at random according to the probability distribution $\{y(i, h, P)/x_i\}_h$. Note that the constraints (1) ensure that the values $y(i, h, P)/x_i$ form a probability distribution over the allowed heights h.

Let $h(i, P)$ be the height chosen for task i on the path P; we use the rectangle $R(i, h(i, P), P)$ to represent task i on the path P. Let $\mathcal{R}$ denote the resulting drawing, i.e., $\mathcal{R}$ is the collection of rectangles selected for the tasks in S. Note that each rectangle $R(i, h, P)$ is in $\mathcal{R}$ with probability $x_i \cdot \frac{y(i,h,P)}{x_i} = y(i, h, P)$.

Alteration phase. In the alteration phase, we select a subset $S' \subseteq S$ of the tasks such that the rectangles $\mathcal{R}' \subseteq \mathcal{R}$ representing them on the paths are non-overlapping. Recall that we view the rectangles as open sets and thus two rectangles overlap iff they contain a common point in their interiors. We consider the paths of $\mathcal{P}$ in an arbitrary order. For each $P \in \mathcal{P}$, let $S(P) = \{i \in S : i \in \mathcal{T}_P\}$. Our goal is to choose a subset $S'(P) \subseteq S(P)$ such that the rectangles $\{R(i, h(i, P), P) \colon i \in S'(P)\}$ are non-overlapping. We choose the set of *accepted tasks* $S'(P)$ as follows.

We order the tasks in $S(P)$ in *non-increasing* order according to their demands, breaking ties arbitrarily. We consider the tasks in this order. Let i be the current task. We add i to $S'(P)$ if the rectangle $R(i, h(i, P), P)$ does not overlap with any of the rectangles $\{R(i', h(i', P), P) \colon i' \in S'(P)\}$ for the tasks we have accepted so far.

We refer to the tasks in $S'(P)$ as the tasks *accepted* on P, and we refer to the tasks in $S(P) - S'(P)$ as the tasks *rejected* on P. The following key lemma shows that each selected task $i \in S(P)$ is accepted with a constant probability. The main observation behind the lemma is that, for each task j that appears before i in the ordering, if the rectangles $R(i, h(i, P), P)$ and $R(j, h(j, P), P)$ overlap, then $R(j, h(j, P), P)$ contains the top left or the bottom left corner of $R(i, h(i, P), P)$ since $d_j \geq d_i$; this allows us to check the constraints only at two points.

Lemma 4. *For any path P and task $i \in \mathcal{T}_P$, $\mathbf{Pr}[i \notin S'(P) \mid i \in S(P)] \leq 2/(c_1 \cdot k)$.*

Finally, we use the sets $\{S'(P) \colon P \in \mathcal{P}\}$ to select a subset $S' \subseteq S$ such that the rectangles $\mathcal{R}' \subseteq \mathcal{R}$ representing S' on each path of $\mathcal{P}$ are non-overlapping. We set $S' = \{i \in S : \forall_{P \in \mathcal{P} : i \in \mathcal{T}_P}\ i \in S'(P)\}$, i.e., a task is accepted if it was accepted for all paths. It follows from Lemma 4 and the union bound that each selected task is rejected with probability at most $|\{P \in \mathcal{P} : i \in \mathcal{T}_P\}| \cdot \frac{2}{c_1 k} \leq 1/2$ if c_1 is sufficiently large[8].

We summarize the rounding step in the following lemma.

Lemma 5. *Consider an instance of* UFP-tree*. Suppose that the instance has a K-nice splitting $\mathcal{P}$ and let (x, y) be a feasible solution to* Rectangle-LP($\mathcal{P}$)*. Let S be a random sample of the tasks such that each task i is in S independently at random with probability $x_i/(4K)$. There is a polynomial-time algorithm that constructs a feasible solution $S' \subseteq S$ such that, for each task i, $\mathbf{Pr}[i \in S' \mid i \in S] \geq 1/2$.*

For linear objective functions, this yields a pseudo-polynomial LP-based $O(k)$-approximation for intersecting instances of UFP-tree and, with Lemma 1, a $O(k \log n)$-approximation for arbitrary instances of UFP-tree.

2.2 A Polynomial-sized Relaxation

In this section, we show how to turn a pseudo-polynomial sized LP in the previous section to a polynomial sized one. Notice that the pseudo-polynomial running time is caused by the fact that the rectangles for the tasks in $\mathcal{T}$ can be drawn at pseudo-polynomially many heights. We show that restricting to a polynomial sized set of heights incurs only an $O(1)$ factor loss in the approximation ratio.

Task classification. For a path $P \in \mathcal{P}$ and a task $i \in \mathcal{T}_P$, let $b_P(i) := \min_{e \in p_i \cap P} u_e$ be the *bottleneck capacity of i on P*. We say that a task $i \in \mathcal{T}_P$ is *big on P* if $d_i > \frac{1}{16} \cdot b_P(i)$. Otherwise we say that i is *small on P*.

[8] More precisely, if $\mathcal{P}$ is ck-nice, then this happens when $c_1 \geq 4c$.

Allowed heights. For each path $P \in \mathcal{P}$ and task $i \in \mathcal{T}_P$, we will now construct a set $\mathcal{H}(i,P)$ of *allowed heights* for drawing the rectangle corresponding to i on P. If i is big on P, we set $\mathcal{H}(i,P) = \{b_P(i) - d_i\}$, i.e., the only allowed height is obtained by drawing the rectangle for i as high as possible underneath the capacity profile. If i is small on P, for the integer j such that $b_P(i) \in [2^j, 2^{j+1})$, we set $\mathcal{H}(i,P) = \bigcup_{r\in\mathbb{N}_0\colon\, r\lceil 2^{j-3}/n\rceil \le 2^{j-1}} \{2^{j-1} + r\lceil 2^{j-3}/n\rceil\}$. We have $|\mathcal{H}(i,P)| \le 8n$. Let $\mathcal{H}$ be the union of all sets $\mathcal{H}(i,P)$. By construction, $\mathcal{H}$ has polynomial size.

Restricted LP. Denote by Restricted-Rectangle-LP($\mathcal{P}$, $\mathcal{H}$) the LP relaxation where we introduce variables $y(i,h,P)$ and the constraints (1) and (2) only for the heights $h \in \mathcal{H}$. As $\mathcal{H}$ has polynomial size, the size of Restricted-Rectangle-LP($\mathcal{P}$, $\mathcal{H}$) is also polynomial. The following lemma argues that the LP restricted to these heights still admit a good fractional solution. Combining it with Lemma 5 yields the desired polynomial time approximation algorithm for linear UFP-tree.

Lemma 6. *For each feasible integral solution $\mathcal{T}' \subseteq \mathcal{T}$, there is a feasible fractional solution (x,y) for* Restricted-Rectangle-LP($\mathcal{P}$, $\mathcal{H}$) *s.t.* $(\forall i \in \mathcal{T}')x_i = \frac{1}{64}$.

2.3 Submodular Objective via the CR Scheme Framework

In this section, we extend our results to submodular objectives by combining the results from the previous section with the framework from [13].

Let N be a finite ground set. Let $\mathcal{I} \subseteq 2^N$ be a family of subsets of N, and $\mathbf{P}_{\mathcal{I}}$ a convex relaxation for the constraints imposed by $\mathcal{I}$, such that $\mathbf{P}_{\mathcal{I}}$ is down-monotone and solvable.[9] Let $x \in \mathbf{P}_{\mathcal{I}}$ and let $\text{support}(x) = \{i \in N\colon x_i > 0\}$. For any $b \in [0,1]$, let $b\cdot\mathbf{P}_{\mathcal{I}} = \{bx\colon x \in \mathbf{P}_{\mathcal{I}}\}$. Let $\mathbf{R}(x)$ be a random sample of N such that each element $i \in N$ is in $\mathbf{R}(x)$ independently at random with probability x_i. For a set function $f : 2^N \to \mathbb{R}_+$ let $F : [0,1]^N \to \mathbb{R}_+$ denote the *multilinear extension* of f, which is defined as $F(x) := \mathbb{E}[f(\mathbf{R}(x))]$.

Definition 2 ([13]). *For $b, c \in [0,1]$, a (b,c)-balanced CR scheme π for a polytope $\mathbf{P}_{\mathcal{I}}$ is a procedure that for every $x \in b\cdot\mathbf{P}_{\mathcal{I}}$ and $A \subseteq N$ returns a random set $\pi_x(A)$ satisfying*
(i) $\pi_x(A) \subseteq \text{support}(x) \cap A$ *and* $\pi_x(A) \in \mathcal{I}$ *with probability* 1*, and*
(ii) for all $i \in \text{support}(x)$, $\mathbf{Pr}[i \in \pi_x(\mathbf{R}(x)) \mid i \in \mathbf{R}(x)] \ge c$.

We use the CR schemes as in [13]: first, we compute a vector x^* with $F(x^*) \ge \Omega(\max\{F(x')\colon x' \in \mathbf{P}_{\mathcal{I}}\})$. Then, we compute a random sample $\mathbf{R}(x)$ with $x := b\cdot x^*$. We apply the CR scheme π and obtain the set $\pi_x(\mathbf{R}(x))$. We know that for each element i we have that $\mathbf{Pr}[i \in \mathbf{R}(x)] = b\cdot x_i^*$ and $\mathbf{Pr}[i \in \pi_x(\mathbf{R}(x)) \mid i \in \mathbf{R}(x)] \ge c$. Thus, $\mathbf{Pr}[i \in \pi_x(\mathbf{R}(x))] \ge bc\cdot x_i^*$ which can be used to show that $\mathbb{E}[f(\pi_x(\mathbf{R}(x)))] \ge \Theta(bc)\cdot\max\{F(x')\colon x' \in \mathbf{P}_{\mathcal{I}}\}$.

[9] We call a polytope $\mathbf{P} \subseteq [0,1]^N$ *down-monotone* if for all $\mathbf{z}, \mathbf{z}' \in [0,1]^N$ we have that $\mathbf{z} \le \mathbf{z}'$ and $\mathbf{z}' \in \mathbf{P}$ implies that $\mathbf{z} \in \mathbf{P}$. The polytope is *solvable* if one can optimize any linear function over $\mathbf{P}$ in polynomial time.

Theorem 4 ([13]). *Let $f : 2^N \to \mathbb{R}_+$ be a submodular function. Let $\mathcal{I} \subseteq 2^N$ be a family of feasible solutions and let $\mathbf{P}_{\mathcal{I}} \subseteq [0,1]^N$ be a convex relaxation for $\mathcal{I}$ that is down-monotone and solvable. Suppose that there is a (b,c)-balanced CR scheme for $\mathbf{P}_{\mathcal{I}}$. Then there is a polynomial time randomized algorithm that constructs a solution $I \in \mathcal{I}$ such that*

$$\mathbb{E}[f(I)] \geq \Theta(bc) \cdot \max\{F(x) \colon x \in \mathbf{P}_{\mathcal{I}}\}.$$

To apply the above framework, let $\mathbf{P}$ denote the set of points x for which there exists a vector y such that (x,y) is contained in the polytope defined by Restricted-Rectangle-LP($\mathcal{P}$,$\mathcal{H}$). Clearly, $\mathbf{P}$ is down-monotone and solvable. Similarly as in the case of linear objective functions, $\mathbf{P}$ contains a fractional point with large profit according to F: Let $\mathcal{T}^*$ be an optimal integral solution. By Lemma 6, $\frac{1}{64} \cdot \mathbf{1}_{\mathcal{T}^*} \in \mathbf{P}$. Moreover, it is straightforward to verify that $F\left(\frac{1}{64} \cdot \mathbf{1}_{\mathcal{T}^*}\right) \geq \frac{1}{64} f(\mathcal{T}^*)$. So $\max\{F(x) : x \in \mathbf{P}_{\mathcal{I}}\} = \Omega(\mathrm{OPT})$.

By Lemma 5, there is a $(1/\Theta(k), 1/2)$-balanced CR scheme for $\mathbf{P}$. Therefore we can apply Theorem 4 to obtain our main result for Submodular UFP-tree.

Theorem 5. *There is a polynomial time $O(k)$ approximation algorithm for* Submodular UFP-tree *on intersecting instances and, therefore, an $O(k \log n)$ approximation for arbitrary instances, where k is the pathwidth of the tree.*

References

1. Adamaszek, A., Wiese, A.: Approximation schemes for maximum weight independent set of rectangles. In: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013. pp. 400–409. IEEE Computer Society (2013)
2. Anagnostopoulos, A., Grandoni, F., Leonardi, S., Wiese, A.: A mazing 2+ϵ approximation for unsplittable flow on a path. In: Chekuri, C. (ed.) Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014. pp. 26–41. SIAM (2014)
3. Bansal, N., Chakrabarti, A., Epstein, A., Schieber, B.: A quasi-PTAS for unsplittable flow on line graphs. In: Kleinberg, J.M. (ed.) Proceedings of the 38th Annual ACM Symposium on Theory of Computing, 2006. pp. 721–729. ACM (2006)
4. Batra, J., Garg, N., Kumar, A., Mömke, T., Wiese, A.: New approximation schemes for unsplittable flow on a path. In: Indyk, P. (ed.) Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015. pp. 47–58. SIAM (2015)
5. Bonsma, P.S., Schulz, J., Wiese, A.: A constant-factor approximation algorithm for unsplittable flow on paths. SIAM J. Comput. 43(2), 767–799 (2014)
6. Boykov, Y., Jolly, M.: Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In: ICCV. pp. 105–112 (2001)
7. Călinescu, G., Chakrabarti, A., Karloff, H.J., Rabani, Y.: An improved approximation algorithm for resource allocation. ACM Transactions on Algorithms 7(4), 48 (2011)
8. Chakaravarthy, V.T., Choudhury, A.R., Gupta, S., Roy, S., Sabharwal, Y.: Improved algorithms for resource allocation under varying capacity. In: Schulz, A.S., Wagner, D. (eds.) Algorithms - ESA 2014 - 22th Annual European Symposium, Proceedings. Lecture Notes in Computer Science, vol. 8737, pp. 222–234. Springer (2014)

9. Chakrabarti, A., Chekuri, C., Gupta, A., Kumar, A.: Approximation algorithms for the unsplittable flow problem. Algorithmica 47(1), 53–78 (2007)
10. Chalermsook, P., Chuzhoy, J.: Maximum independent set of rectangles. In: Mathieu, C. (ed.) Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009. pp. 892–901. SIAM (2009)
11. Chan, T.M., Har-Peled, S.: Approximation algorithms for maximum independent set of pseudo-disks. Discrete & Computational Geometry 48(2), 373–392 (2012)
12. Chekuri, C., Ene, A., Korula, N.: Unsplittable flow in paths and trees and column-restricted packing integer programs. In: Dinur, I., Jansen, K., Naor, J., Rolim, J.D.P. (eds.) Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Proceedings. Lecture Notes in Computer Science, vol. 5687, pp. 42–55. Springer (2009)
13. Chekuri, C., Vondrák, J., Zenklusen, R.: Submodular function maximization via the multilinear relaxation and contention resolution schemes. SIAM J. Comput. 43(6), 1831–1879 (2014)
14. Dueck, D., Frey, B.J.: Non-metric affinity propagation for unsupervised image categorization. In: IEEE 11th International Conference on Computer Vision, ICCV 2007. pp. 1–8. IEEE (2007)
15. Friggstad, Z., Gao, Z.: On linear programming relaxations for unsplittable flow in trees. In: Garg, N., Jansen, K., Rao, A., Rolim, J.D.P. (eds.) Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015. LIPIcs, vol. 40, pp. 265–283. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2015)
16. Jegelka, S., Bilmes, J.A.: Submodularity beyond submodular energies: Coupling edges in graph cuts. In: The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011. pp. 1897–1904. IEEE Computer Society (2011)
17. Kempe, D., Kleinberg, J.M., Tardos, É.: Maximizing the spread of influence through a social network. Theory of Computing 11, 105–147 (2015)
18. Korach, E., Solel, N.: Tree-width, path-width, and cutwidth. Discrete Applied Mathematics 43(1), 97–101 (1993)
19. Krause, A., Guestrin, C.: Near-optimal observation selection using submodular functions. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, 2007. pp. 1650–1654. AAAI Press (2007)
20. Krause, A., Guestrin, C.: Submodularity and its applications in optimized information gathering. ACM TIST 2(4), 32 (2011)
21. Krause, A., Singh, A.P., Guestrin, C.: Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. Journal of Machine Learning Research 9, 235–284 (2008)
22. Lin, H., Bilmes, J.A.: A class of submodular functions for document summarization. In: Lin, D., Matsumoto, Y., Mihalcea, R. (eds.) The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 2011. pp. 510–520. The Association for Computer Linguistics (2011)
23. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions–I. Mathematical Programming 14(1), 265–294 (1978)