Improved Approximation Algorithms for Unsplittable Flow on a Path with Time Windows*

Fabrizio Grandoni, Salvatore Ingala, and Sumedha Uniyal

IDSIA, University of Lugano, Switzerland
{fabrizio,salvatore,sumedha}@idsia.ch

Abstract. In the well-studied *Unsplittable Flow on a Path* problem (UFP), we are given a path graph with edge capacities. Furthermore, we are given a collection of n tasks, each one characterized by a subpath, a weight, and a demand. Our goal is to select a maximum weight subset of tasks so that the total demand of selected tasks using each edge is upper bounded by the corresponding capacity. Chakaravarthy et al. [ESA'14] studied a generalization of UFP, bagUFP, where tasks are partitioned into bags, and we can select at most one task per bag. Intuitively, bags model jobs that can be executed at different times (with different duration, weight, and demand). They gave a $O(\log n)$ approximation for bagUFP. This is also the best known ratio in the case of uniform weights. In this paper we achieve the following main results:

- We present an LP-based $O(\log n/\log\log n)$ approximation for bagUFP. We remark that, prior to our work, the best known integrality gap (for a non-extended formulation) was $O(\log n)$ even in the special case of UFP [Chekuri et al., APPROX'09].
- We present an LP-based $O(1)$ approximation for uniform-weight bagUFP. This also generalizes the integrality gap bound for uniform-weight UFP by Anagnostopoulos et al. [IPCO'13].
- We consider a relevant special case of bagUFP, twUFP, where tasks in a bag model the possible ways in which we can schedule a job with a given processing time within a given time window. We present a QPTAS for twUFP with quasi-polynomial demands and under the Bounded Time-Window Assumption, i.e. assuming that the time window size of each job is within a constant factor from its processing time. This generalizes the QPTAS for UFP by Bansal et al. [STOC'06].

1 Introduction

In the well-studied *Unsplittable Flow on a Path* problem (UFP) we are given a path graph $G = (V, E)$, $V = \{0, 1, \ldots, m\}$, with positive integer edge capacities $\{u_e\}_{e\in E}$ and a collection T of n tasks. Each task $i \in T$ is associated with a weight $w_i \in \mathbb{N}^+$, a demand $d_i \in \mathbb{N}^+$, and a subpath P_i between nodes s_i and t_i. Let $T_e = \{i \in T : e \in P_i\}$ be the tasks *containing* edge e. Our goal is to select

* This work is partially supported by the ERC StG project NEWNET no. 279352.

a subset of tasks $T' \subseteq T$ of maximum total weight $w(T') := \sum_{i \in T'} w_i$ so that, for each edge e, the total demand $d_e(T') := \sum_{i \in T' \cap T_e} d_i$ of selected tasks using that edge is upper bounded by the corresponding capacity u_e. Intuitively, edge capacities model a given resource whose amount varies over a given time interval (in a discrete fashion), and tasks demand for some amount of that resource. In particular, the length of each subpath can be interpreted as a processing time. By standard reductions [4, 8], we can assume that $m \leq 2n$ and all edge capacities are distinct.

UFP is strongly NP-hard [14]. Anagnostopoulos et al. [2] recently gave the current best $2 + \varepsilon$ approximation for the problem[1], improving on [5, 8]. This matched a previously known [13] approximation for UFP under the No-Bottleneck Assumption (NBA), i.e. assuming that the largest demand is upper bounded by the smallest capacity. This matched also the best known approximation for the uniform-capacity case [9].

The *UFP with Bags* problem (bagUFP) is the generalization of UFP where tasks are partitioned into a set of h bags $J = \{\mathcal{B}_1, \ldots, \mathcal{B}_h\}$, and we have the extra constraint that at most one task per bag can be selected. Intuitively, bags model *jobs* that we can execute at different points of time (and at each such time one has a different demand, weight, and processing time). This problem is APX-hard even in the case of unit demands and capacities [16]. Chakaravarthy et al. [10] recently gave the current best $O(\log n)$ approximation for bagUFP. The approximation factor remains the same in the case of uniform weights. The same authors also presented a $O(1)$ approximation under NBA.

1.1 Our Contribution.

In this paper we present an improved approximation for bagUFP (see Section 3). In the special case of uniform weights, we can reduce the approximation factor down to a constant (see Section 4).

Theorem 1. *There is an expected $O(\log n / \log \log n)$ approximation for bagUFP.*

Theorem 2. *There is an $O(1)$ approximation for uniform-weight bagUFP.*

Both our results are LP-based, and exploit a refined LP for bagUFP which is inspired by the work on UFP in [1, 8]. In more detail, let us define a task *large* if it uses more than one half of the capacity of some edge along its subpath, and let T_{large} be the large tasks. Bonsma et al. [8] introduced a geometric interpretation of *large* tasks. They associate to each $i \in T_{large}$ an axis-parallel rectangle R_i in the 2D plane with top-left corner (s_i, b_i) and bottom-right corner $(t_i, b_i - d_i)$, where $b_i = \min_{e \in P_i}\{u_e\}$ is the bottleneck capacity of task i. We call this set of rectangles $\mathcal{R}$ the *top-drawn* representation of T_{large}. In [8, Lemma 13] it is shown that in any feasible UFP (hence bagUFP) solution at most 4 corresponding rectangles can overlap at a given point (intuitively, those large tasks induce

[1] Unless differently stated, ε denotes an arbitrarily small positive constant parameter. Where needed, we also assume that $1/\varepsilon$ is integral and sufficiently large.

an *almost independent* set of rectangles). This insight was later used by Anagnostopoulos et al. [1]. Consider the grid induced by the horizontal and vertical lines containing the rectangle sides. Let $\mathcal{P}$ be the set of (*representative*) $O(n^2)$ middle points of the (positive area) cells of this grid. Note that any subset of rectangles that share a positive size area, will overlap on some point in $\mathcal{P}$. The authors consider the following (non-extended[2]) LP relaxation for UFP:

$$\begin{array}{lll} \max & \sum_{i \in T} w_i x_i & (LP_{UFP^+}) \\ \text{s.t.} & \sum_{i: e \in P_i} d_i x_i \leq u_e & \forall\, e \in E \qquad (1) \\ & \sum_{i \in T_{large}: p \in R_i} x_i \leq 4 & \forall\, p \in \mathcal{P} \qquad (2) \\ & x_i \geq 0 & \forall\, i \in T \end{array}$$

We call the constraints of type (1) and (2) *capacity* and *rectangle constraints*, respectively. The authors show that this relaxation has $O(1)$ integrality gap in the case of uniform weights. Note that one can preprocess the instance so that the weights range between 1 and $O(n/\varepsilon)$ while losing a factor $1+\varepsilon$ in the approximation. By partitioning tasks in $O(\log(n/\varepsilon))$ classes of almost uniform weight, one obtains that LP_{UFP^+} has $O(\log n)$ integrality gap for general weights[3].

In this paper we consider the LP relaxation LP_{bagUFP^+} for bagUFP which is obtained from LP_{UFP^+} by adding the following *bag constraints*:

$$\sum_{i \in \mathcal{B}_j} x_i \leq 1 \qquad \forall\, \mathcal{B}_j \in J.$$

The standard LP relaxation LP_{bagUFP} for bagUFP is obtained from bagUFP$^+$ by removing the rectangle constraints. We show that LP_{bagUFP^+} has constant integrality gap in the uniform weight case, and integrality gap $O(\log n / \log\log n)$ in the general case. In particular, for the uniform-weight case we can adapt the analysis in [1], while for the general case we can generalize the rounding procedure of Chan and Har-Peled [11] for the maximum independent set of rectangles problem. Note that the latter result slightly improves the best integrality gap even in the case of UFP (for a compact, non-extended LP relaxation).

We also study a relevant special case of bagUFP (and generalization of UFP), that we name *UFP with Time Windows* (twUFP). Here we are given a capacitated path graph and a collection of *jobs*, where each job j is characterized by a weight, a demand, a (positive, integer) *processing time* τ_j and a *time window* W_j (i.e. a subpath between given nodes s_j and t_j). For each possible node σ_i (*starting time*) so that $s_j \leq \sigma_i \leq t_j - \tau_j$, we define a task i with the same weight and demand as j, and whose subpath P_i has endpoint $s_i = \sigma_i$ and $t_i = \sigma_i + \tau_i$. The tasks corresponding to the same job j define a bag $\mathcal{B}_j$. Intuitively, tasks in $\mathcal{B}_j$ describe the possible ways in which we can process job j within its time window. Our goal is to compute a maximum weight solution for the resulting

[2] By non-extended we mean that it contains only decision variables for tasks. In the same paper the authors present an extended formulation with $O(1)$ integrality gap.

[3] The same gap is proved by Chekuri et al. [12]. The authors claim a $O(\log^2 n)$ gap, and then refine it to $O(\log n)$ in an unpublished manuscript.

bagUFP instance. We believe that in practice several instances of bagUFP are indeed instances of twUFP. The best-known approximation for twUFP is also $O(\log n)$, where n is the number of tasks ($O(\log n / \log\log n)$ considering Theorem 1). In particular, the approach in [10] does not seem to benefit from the special structure of twUFP, nor does the approach from Theorem 1.

We present a QPTAS[4] for twUFP under the following *Bounded Time-Window Assumption* (BTWA): for any job j, $(t_j - s_j)/\tau_j \leq C = O(1)$ (in words, the ratio between the time window size and the processing time is bounded by some constant). Our result generalizes the QPTAS for UFP by Bansal et al. [4]. Here, similarly to [4], we assume[5] that demands are quasi-polynomially bounded in n.

Theorem 3. *There is a QPTAS for twUFP under BTWA and assuming that demands are quasi-polynomially bounded in n.*

Indeed, our QPTAS generalizes to the special case of bagUFP where tasks in the same bag have the same demand and weight (under the natural generalization of BTWA, that is, under the assumption that the processing time of any task i contained in bag $\mathcal{B}_j$ is at most a constant factor shorter than the length of the bag $\max_{i\in\mathcal{B}_j} t_i - \min_{i\in\mathcal{B}_j} s_i$). Note that bagUFP is APX-hard, therefore there is not much hope for a PTAS for it. In contrast, our result provides an evidence that twUFP might be an easier problem, at least under BTWA. It is unclear to us whether the general case of twUFP is as hard to approximate as bagUFP.

In order to understand our contribution, it is convenient to sketch how the QPTAS for UFP in [4] works. Let us consider the tasks OPT_{mid} in the optimum solution OPT that use the middle edge e_{mid}. The authors show how to define a *capacity profile* u_{mid}, *dominated* by the demand of OPT_{mid}, which has a quasi-polynomial number of *steps*, and such that there is a feasible solution APX_{mid} for capacities u_{mid} of cost close to OPT_{mid} and which can be computed in QPT. Thus one can *guess* u_{mid}, compute APX_{mid}, and branch on a left and right subproblem (where capacities are decreased by u_{mid}, and we consider only tasks fully contained to the left/right of e_{mid}).

This approach does not work for twUFP since a time window might be split by e_{mid}. In that case the left and right subproblems are not independent any more (in particular, we cannot select two tasks from the same bag, one from the left subproblem and the other from the right one). To circumvent this problem, we exploit the *randomized dissection* technique by Grandoni and Rothvoß for the related *Highway* problem [15]. We evenly split the path into a random constant number of intervals, and iterate the process on each such interval. With probability close to one, the time window of each job j is fully contained in some interval I of the dissection, and at the same time none of its tasks $\mathcal{B}_j$ is fully

[4] We recall that a *Quasi-Polynomial-Time Approximation Scheme* (QPTAS) is an algorithm that, given a constant parameter $\varepsilon > 0$, computes a $1+\varepsilon$ approximation in *Quasi-Polynomial Time* (QPT), i.e. in time $2^{poly\log(s)}$ where s is the input size.

[5] We remark that Batra et al. [7] recently managed to remove this assumption on the demands for UFP. Their approach does not seem to be compatible with our randomized dissection technique (at least not trivially).

contained in a subinterval of I (here we need the BTWA). Thus we can define a capacity reservation that combines a constant number of capacity profiles, and use a proper algorithm (rather different from [4], due to bag constraints) to compute a good approximation for the considered jobs. We can then branch on the subproblems induced by the subintervals.

Omitted proofs are provided in the full version of this paper.

2 A QPTAS for the Bounded Time-Window Case

In this section we present a QPTAS for twUFP under BTWA, and assuming that the largest demand D_{max} is quasi-polynomially bounded in the number of tasks n. By standard tricks, while losing only a factor $1+\varepsilon$ in the approximation factor, we can assume that weights range between 1 and $O(n/\varepsilon)$.

A *capacity reservation* r is simply a collection of edge capacities $\{r_e\}_{e\in E}$ with $r_e \leq u_e$ for all edges e. A solution is feasible w.r.t. r if it respects the capacity constraints induced by r. We say that r has k-steps if, scanning edges from left to right, the value of their capacity changes at most k times. For another capacity reservation r', we say that r *dominates* r' if $r_e \geq r'_e$ for each edge e. For a set of tasks S, we say that r *is dominated by the demand of* S if $r_e \leq \sum_{i\in S\,:\,e\in P_i} d_i$ for each edge e.

The following technical lemma is similar in spirit to results in [4].

Lemma 1. *Let S be a collection of at least $2/\varepsilon^3$ tasks using a given edge e, and with demand in $[D,(1+\varepsilon)D)$ and weight in $[W,(1+\varepsilon)W)$. Then there exists a capacity reservation r and a set of tasks $R\subseteq S$ such that: (1) r is dominated by the demand of S; (2) r has $O(1/\varepsilon^2)$ steps and its entries are integer multiples of $(1+\varepsilon)D$; (3) R is feasible for r, even if the paths of its tasks are expanded to the left/right to reach the closest edge before a change of capacity in r and their demand is increased to $(1+\varepsilon)D$; (4) $w(R)\geq(1-O(\varepsilon))w(S)$.*

Next lemma will be used to partition the input problem into a quasi-polynomial number of subproblems. For a given set of edges F, let J_F be the set of jobs j such that each task in $\mathcal{B}_j$ contains some edge in F, and let $T_F=\cup_{j\in J_F}\mathcal{B}_j$. Note that containing an edge in F is not sufficient for a task i to be in T_F. We define $T_{\bar{F}}=T\setminus T_F$.

Lemma 2. *Consider a twUFP instance with optimal solution OPT, and let F be a subset of $O(1)$ edges. There exists a QPT algorithm that generates a set $\mathcal{U}_F$ of capacity reservations r_F, and a feasible solution $APX_F\subseteq T_F$ for each such r_F such that, for at least one such pair $\{r^*,APX^*\}$, $APX:=(OPT\cap T_{\bar{F}})\cup APX^*$ is a feasible twUFP solution and $w(APX)\geq(1-O(\varepsilon))w(OPT)$.*

Proof. Let $OPT_F:=OPT\cap T_F$ and $OPT_{\bar{F}}:=OPT\cap T_{\bar{F}}$. We first show how to construct a capacity reservation r^* which is dominated by the demand of OPT_F. Let $T^{f,a,b}$ be the class of tasks $i\in T_F$ with $d_i\in[(1+\varepsilon)^a,(1+\varepsilon)^{a+1})$ and $w_i\in[(1+\varepsilon)^b,(1+\varepsilon)^{b+1})$ for $a,b\in\mathbb{N}$, and such that f is the leftmost edge in $P_i\cap F$. Observe that there are $O_\varepsilon(\log n\log D_{max})$ (non-empty) such classes. Define

$OPT^{f,a,b} := OPT \cap T^{f,a,b}$. Suppose that $|OPT^{f,a,b}| \geq 2/\varepsilon^3$. Then we apply Lemma 1 with $S = OPT^{f,a,b}$ and $e = f$, hence obtaining a capacity reservation $r^{f,a,b}$ and a solution $R^{f,a,b}$. Otherwise, we simply let $R^{f,a,b} = OPT^{f,a,b}$ and $r^{f,a,b}$ be the total demand of $R^{f,a,b}$. Let $r^* = \sum_{f,a,b} r^{f,a,b}$. Observe that r^* has $O_\varepsilon(|F| \log n \log D_{max})$ steps, and each entry of r^* is obtained from the total demand of $O(|F|/\varepsilon^3)$ tasks plus an integer multiple of $(1+\varepsilon)^{a+1}$ for $O_\varepsilon(\log D_{max})$ possible values of a. Therefore in QPT we can enumerate a set $\mathcal{U}_F$ of capacity reservations that includes r^*.

Our algorithm constructs (in QPT) a feasible solution for each capacity reservation in $\mathcal{U}_F$. For the sake of simplicity, we next focus on the solution APX^* corresponding to the reservation r^* described before. Note that, since r^* is dominated by the demand of OPT_F, $APX^* \cup OPT_{\bar{F}}$ has to satisfy the capacity constraints. Furthermore, the bags of tasks in $OPT_{\bar{F}}$ are disjoint from the bags of tasks in T_F by definition (hence also bag constraints are satisfied). We will later show that $w(APX^*) \geq (1 - O(\varepsilon))w(OPT_F)$. The claim follows.

Let us focus on a given pair (a, b). We guess[6] the set $F^{a,b}_{few}$ of all the edges $f \in F$ such that $|OPT^{f,a,b}| < 2/\varepsilon^3$, and the corresponding tasks $APX^{f,a,b} := OPT^{f,a,b} = R^{f,a,b}$ with demand $r^{f,a,b}$. The corresponding jobs are removed from the instance. Let $F^{a,b}_{many} := F \setminus F^{a,b}_{few}$. For any $f \in F^{a,b}_{many}$, we guess the capacity reservation $r^{f,a,b}$.[7] Note that this reservation has $O(1/\varepsilon^2)$ steps, and its entries are integer multiples of $(1+\varepsilon)D$, $D = (1+\varepsilon)^a$. We expand all the tasks in $T^{f,a,b}$ to the left/right till the closest edge before a change in the capacity of $r^{f,a,b}$ and increase their demand to $(1+\varepsilon)D$.

Next we consider the bagUFP instance induced by rounded tasks $T^{a,b}_{many} := \cup_{f \in F^{a,b}_{many}} T^{f,a,b}$, with edge capacities given by $r^{a,b}_{many} := \sum_{f \in F^{a,b}_{many}} r^{f,a,b}$. Observe that all the tasks of a remaining job are considered in the same such instance[8]. We also remark that $R^{a,b}_{many} := \cup_{f \in F^{a,b}_{many}} R^{f,a,b}$ is a feasible solution to this bagUFP instance by construction. We also remark that $r^{a,b}_{many}$ has $O(|F|/\varepsilon^2)$ steps, hence by contracting edges one obtains an equivalent bagUFP instance with a constant number of edges.

We next show how to compute the optimal solution $APX^{a,b}_{many}$ for this bagUFP instance via dynamic programming. Let us sort the considered $h^{a,b}_{many}$ jobs arbitrarily. In our dynamic program we have a table entry (h', r') for each $h' = 1, \ldots, h^{a,b}_{many}$ and for each feasible capacity reservation r' dominated by $r^{a,b}_{many}$ and whose capacities are non-negative integer multiples of $(1+\varepsilon)D$. Note that there is a polynomial number of table entries. The value $DP(h', r')$ of this entry will be set to the maximum weight of a feasible bagUFP for r' using tasks from the first h' jobs only. Table entries are filled in for increasing values of h'.

[6] Throughout this paper, by guessing we mean trying all the possibilities.
[7] In the guessing we of course guarantee that $r^* = \sum_{f,a,b} r^{f,a,b}$.
[8] Here we exploit a property of twUFP not satisfied by bagUFP.

It is easy to compute the values $DP(1, r')$ (base case). For any $h' > 1$, one has[9]

$$DP(h', r') = \max\{DP(h'-1, r'), \max_{i \in \mathcal{B}_j}\{w_j + DP(h'-1, r'_i)\}\},$$

where r'_i is obtained from r' by subtracting the demand of task i. The desired solution $APX^{a,b}_{many}$ is the one corresponding to $DP(h^{a,b}_{many}, r^{a,b}_{many})$.

Our global solution is $APX^* = \cup_{a,b}(APX^{a,b}_{many} \cup (\cup_{f \in F^{a,b}_{few}} APX^{f,a,b}))$, with

$$\begin{aligned} w(APX^*) &= \textstyle\sum_{a,b,f \in F^{a,b}_{few}} w(OPT^{f,a,b}) + \sum_{a,b} w(APX^{a,b}_{many}) \\ &\geq \textstyle\sum_{a,b,f \in F^{a,b}_{few}} w(OPT^{f,a,b}) + \sum_{a,b,f \in F^{a,b}_{many}} w(R^{f,a,b}) \\ &\overset{Lem.1}{\geq} \textstyle\sum_{a,b,f \in F^{a,b}_{few}} w(OPT^{f,a,b}) + \sum_{a,b,f \in F^{a,b}_{many}} (1 - O(\varepsilon)) w(OPT^{f,a,b}) \\ &\geq (1 - O(\varepsilon)) w(OPT_F). \end{aligned}$$

□

We are now ready to describe the global algorithm, which is inspired by [15].

We first embed G into a longer random path G' as follows. Let $\gamma = (1/\varepsilon')^{1/\varepsilon'}$, where $1/\varepsilon' = \lceil 1/\min\{\varepsilon, 1/C\}) \rceil$ (in particular, $\varepsilon' \leq \min\{\varepsilon, 1/C\}$). Let m be the number of edges in the input graph. By adding dummy edges, we can assume that $m = \gamma^\ell$ for some integer ℓ. We choose integers $x \in \{1, \ldots, m\}$ and $y \in \{1, \ldots, 1/\varepsilon'\}$ uniformly at random. Next we append x dummy edges to the left of the path and $m \cdot ((1/\varepsilon')^y - 1) - x$ dummy edges to its right. All dummy edges have capacity one. Let G' be the resulting path graph, with $m' = \gamma^\ell (1/\varepsilon')^y$ edges. We remark that this step can be easily derandomized by considering all the possible values for x and y.

We next consider the following recursive dissection of G'. We split G' into γ intervals of equal length (in terms of number of edges). Each such interval is subdivided recursively in the same way, and we halt when we reach intervals of length γ or less. We let $I_1, \ldots, I_\gamma$ denote the (direct) subintervals of interval I. We remark that intervals at level $q \geq 0$ in this dissection have length $\alpha_q := m'/\gamma^q$.

We say that a job j is at level $\ell(j)$ in this dissection if its time window W_j is fully contained in an interval $I(j)$ of level $\ell(j)$, but not of level $\ell(j) + 1$. We similarly define $\ell(i)$ and $I(i)$ for a task i. For a given interval I, let $J(I)$ be the jobs whose time window if fully contained in I, but not in any one of its subintervals. Among them, we call *good* the jobs $Gd(I)$ such that all their tasks i have $I(i) = I(j)$ (i.e., they are not fully contained in a subinterval of I), and *bad* the remaining jobs $Bd(I)$[10]. We discard from the instance all the bad jobs $Bd := \cup_I Bd(I)$.

Then we apply the following recursive algorithm, that takes as input one such interval I and a *residual capacity* u' coming from earlier calls. In the root

[9] Intuitively, the first term in the outer max corresponds to the case that the best solution does not use job k, and the second term to the weight obtained by including some task $i \in \mathcal{B}_k$ in the solution.

[10] We call good the jobs of level ℓ by definition.

call we use $I = G'$ and $u' = u$. Let $F(I)$ be the set of rightmost edges of the subintervals of I. We consider the twUFP instance induced by (I, u') with jobs j such that $W_j \subseteq I$ (excluding the discarded bad jobs $Bd(I)$), and we apply to this instance Lemma 2 with $F = F(I)$. We remark that the tasks T_F in this case are precisely the tasks of good jobs $Gd(I)$.

This generates a quasi-polynomial size set of pairs $\{r(I), APX(I)\}$. For each such pair $\{r(I), APX(I)\}$ the algorithm branches by solving recursively each subproblem induced by each subinterval I_i, with capacity reservation u'_i induced by $u' - r(I)$: let $APX(I_i)$ be the resulting solution. The output of this recursive call is the maximum weight solution among the solutions of type $APX(I) \cup (\cup_i APX(I_i))$. The base case is given by intervals I of length at most γ. A QPTAS for this instance is provided by Lemma 2: just choose F to be all the edges in I.

It is not hard to see that the above recursive algorithm is QPT. Furthermore, it outputs a $1 - O(\varepsilon)$ approximation of the optimal solution, restricted to the subset of good jobs $Gd := \cup_I Gd(I)$. Next lemma shows that each given job is bad with sufficiently small probability. Theorem 3 follows.

Lemma 3. *Each job is good with probability at least* $1 - 3\varepsilon$.

Proof. Let us upper bound the probability that a job j is bad. We next assume that $\ell(j) < \ell$, otherwise j is deterministically good by definition and there is nothing to show.

We say that j is *risky* if there exists q such that $\varepsilon'\alpha_q \leq t_j - s_j \leq \frac{1}{\varepsilon'}\alpha_q$. We next bound the probability that j is risky. Consider a log-scale axis and call segment the distance corresponding to a multiplicative factor of $1/\varepsilon'$. The regions of risky time-window lengths correspond to 2 segments for each value of q, separated by $1/\varepsilon' - 2$ segments which are not risky. By the random choice of y, the risky regions are shifted randomly w.r.t. the time-window lengths. Therefore each job is risky with probability at most $2/(1/\varepsilon') = 2\varepsilon' \leq 2\varepsilon$.

Let us next condition on the event that job j is not risky. Then there exists a q such that $\frac{1}{\varepsilon'}\alpha_q < t_j - s_j < \varepsilon'\alpha_{q-1}$. If $\ell(j) = q - 1$, then j is good: indeed, $\tau_j \geq (t_j - s_j)/C \geq \varepsilon'(t_j - s_j) > \alpha_q$. Thus each path P_i, $i \in \mathcal{B}_j$, is strictly longer than the level q subintervals of $I(j)$.

Due to the random choice of x, the endpoints of the intervals of level $q - 1$ are randomly shifted w.r.t the time window W_j, and $\ell(j) < q - 1$ only if W_j crosses some interval of level $q - 1$. Therefore $Pr[\ell(j) < q - 1] \leq \frac{t_j - s_j}{\alpha_{q-1}} \leq \varepsilon' \leq \varepsilon$. Altogether, job j is bad with probability at most 3ε.

Remark 1. The above QPTAS extends to the special case of bagUFP where tasks in the same bag have the same demand and weight (under the natural analogue of BTWA). In particular, it is sufficient to adapt the DP from Lemma 2. However, it does not seem to extend to the case that weights and demands are arbitrary (since in that case the same bag might influence different capacity profiles $r^{a,b}_{many}$, which therefore cannot be considered separately in the DP).

3 An Improved Approximation for bagUFP

In the *Maximum Independent Set of Rectangles* problem (MISR) we are given a collection $\mathcal{R} = \{R_1, \ldots, R_n\}$ of axis-parallel rectangles in the 2D plane, where R_i has weight w_i. Our goal is to find a maximum total weight subset of rectangles which are pairwise non-overlapping[11]. We define bagMISR as the natural generalization of MISR with bags $J = \{\mathcal{B}_1, \ldots, \mathcal{B}_h\}$.

We first present a $O(\log n / \log\log n)$ approximation for bagMISR, and then show how to use it to achieve the same approximation factor for bagUFP.

Approximating bagMISR. Let $\mathcal{P}$ be the set of $O(n^2)$ *representative* points for rectangles $\mathcal{R}$ obtained with the already mentioned construction. Consider the following natural LP relaxation for bagMISR:

$$\begin{array}{lll} \max & \sum_{R_i \in \mathcal{R}} w_i y_i & (LP_{\text{bagMISR}}) \\ \text{s.t.} & \sum_{R_i \in \mathcal{R}: p \in R_i} y_i \leq 1 & \forall\, p \in \mathcal{P} \\ & \sum_{R_i \in \mathcal{B}_j} y_i \leq 1 & \forall\, \mathcal{B}_j \in J \\ & y_i \geq 0 & \forall\, R_i \in \mathcal{R} \end{array}$$

The standard LP relaxation LP_{MISR} for MISR is obtained from the above LP by removing the bag constraints. Let $\mathcal{B}(R_i)$ be the set of rectangles in the same bag of R_i, and, for an arbitrary set of rectangles $\mathcal{R}' \subseteq \mathcal{R}$, let $y(\mathcal{R}') = \sum_{i \in \mathcal{R}'} y_i$. For two overlapping, distinct rectangles R_i and R_j, we say that they *corner-intersect*, and write $R_i \otimes R_j$, if one rectangle contains at least one corner of the other rectangle. Otherwise they *cross*. For an arbitrary set $\mathcal{R}' \subseteq \mathcal{R}$ and rectangle $R_i \in \mathcal{R}$, define the *resistance* η as:

$$\eta(R_i, \mathcal{R}') = \sum_{\substack{R_j \in \mathcal{R}' \setminus \mathcal{B}(R_i), \\ R_i \otimes R_j}} y_j + \sum_{R_j \in \mathcal{R}' \cap \mathcal{B}(R_i) \setminus \{R_i\}} y_j$$

Let G_1 and G_2 be two undirected graphs with vertex set $\mathcal{R}$ constructed as follows: if two distinct rectangles R_i and R_j are *incompatible* (i.e., they overlap or are in the same bag), then the edge (R_i, R_j) is added to G_1 if $R_i \otimes R_j$ or if they are in the same bag, otherwise the edge (R_i, R_j) is added to G_2. Of course, a subset $I \subseteq \mathcal{R}$ is a feasible solution if and only if it induces an independent set of nodes in both the graphs simultaneously.

Our approximation algorithm works as follows. First, a fractional optimum solution y of LP_{bagMISR} is found. Then a permutation Π of $\mathcal{R}$ is computed in the following manner: given the first i rectangles $\Pi_i = \{\pi_1, \ldots, \pi_i\}$ in the permutation, the $(i+1)^{th}$ element π_{i+1} (breaking ties arbitrarily) is:

$$\pi_{i+1} = \arg \min_{R_j \in \mathcal{R} \setminus \Pi_i} \eta(R_j, \mathcal{R} \setminus \Pi_i)$$

[11] For our goals, it is convenient to consider two rectangles as overlapping iff they overlap on a positive value area. In particular, overlapping on rectangle boundaries is allowed.

We next compute a candidate set C and an independent set $I \subseteq C$ of G_1. Initially, C and I are empty. Then, the members of the permutation are scanned in reverse order: at iteration k, the rectangle $\pi_{n-(k-1)}$ is added to C independently with probability $y(\pi_{n-(k-1)})/10$. If $\pi_{n-(k-1)}$ is added to C and $I \cup \{\pi_{n-(k-1)}\}$ is an independent set in G_1, then $\pi_{n-(k-1)}$ is also added to I.

Note that I might not be an independent set due to crossing intersections. Let Δ be the maximum clique-size of the rectangles in $G_2[I]$, that is, the maximum number of rectangle overlapping on the same point. Since the rectangles in I only have crossing intersections, $G_2[I]$ can be colored in polynomial time using Δ colors as shown in [3]. The algorithm then returns the color subclass I' of I that has the largest total weight. Clearly, I' is an independent set in both G_1 and G_2, and thus it is a feasible solution. We can bound the approximation ratio similarly to [11].

Lemma 4. *There is an expected $O(\log n/\log\log n)$ approximation for bagMISR.*

Approximating bagUFP. Our algorithm works as follows. We first compute an approximate solution APX_{small} associated to *small* tasks $T_{small} = T \setminus T_{large}$ using the algorithm in [10]. We recall that this algorithm computes a constant approximation of the optimal fractional solution of LP_{bagUFP} restricted to small tasks [10, Lemma 1]. Next we focus on large tasks, and on the corresponding set of top-drawn rectangles $\mathcal{R}$. We consider the bagMISR instance induced by $\mathcal{R}$. We compute a solution $\mathcal{R}'$ for this instance using the algorithm from Lemma 4. Let APX_{large} be the tasks corresponding to $\mathcal{R}'$ (observe that APX_{large} is a feasible bagUFP solution). We finally return the best solution APX between APX_{small} and APX_{large}.

Proof. (of Theorem 1) Consider the above algorithm. We prove that APX is a $O(\log n/\log\log n)$ approximation with respect to the cost *opt* of the optimal fractional solution x to LP_{bagUFP^+}. Let x^{small} be the restriction of x to small tasks, and opt^{smal} be the corresponding weight. We define x^{large} and opt^{large} analogously for large tasks.

If $opt^{small} \geq opt/2$, then APX_{small} has the desired properties by [10]. Indeed, x^{small} if a feasible solution to LP_{bagUFP}.

Otherwise, let $y^{large} = x^{large}/4$. Observe that y^{large} is a feasible solution for $LP_{bagMISR}$. Therefore, APX_{large} provides a $O(\log n/\log\log n)$ approximation of the weight of y^{large} (hence of opt^{large}). The claim follows. □

4 A $O(1)$-Approximation for Uniform Profits

In this section we present our $O(1)$ approximation for bagUFP with uniform weights, which also upper bounds the integrality gap of LP_{bagUFP^+} in the same case. By scaling, we can assume w.l.o.g. that weights are exactly one.

By the same argument as in the proof of Theorem 1, it is sufficient to provide a $O(1)$ approximation for the (uniform-weight) bagMISR instance induced by the top-drawn rectangles $\mathcal{R}$ corresponding to large tasks. We use as a black box

the following result proved (implicitly) in [1]. We recall that in the *Maximum Independent Set of Intervals* problem (MISI) we are given a collection $\mathcal{I}$ of intervals along a line, each one with an associated weight, and our goal is to compute a maximum weight subset of intervals $\mathcal{I}'$ so that the intervals in $\mathcal{I}'$ are pairwise non-overlapping. By bagMISI we denote the natural generalization of MISI with bag constraints. We let LP_{MISI} be the standard LP for MISI, which is defined analogously to LP_{MISR}.

Lemma 5. *Let $\mathcal{R}'$ be a set of top-drawn rectangles corresponding to a subset of large tasks in an UFP instance, and let $\mathcal{R}_{max}$ be any maximal independent set of rectangles in $\mathcal{R}'$. There is a polynomial-time algorithm that computes up to 10 points $\mathcal{P}_i$ in the plane for each $R_i \in \mathcal{R}_{max}$, and four subsets $\mathcal{R}_{point}, \mathcal{R}_{top}, \mathcal{R}_{left}, \mathcal{R}_{right} \subseteq \mathcal{R}'$ that cover $\mathcal{R}'$ so that:*

1. *$\mathcal{R}_{point} = \{R_i \in \mathcal{R}' : R_i \cap \mathcal{P} \neq \emptyset\}$ with $\mathcal{P} = \cup_{R_i \in \mathcal{R}_{max}} \mathcal{P}_i$.*
2. *For each $x \in \{top, left, right\}$, there exists a bijection between $\mathcal{R}_x$ and a collection $\mathcal{I}_x$ of intervals along a line, so that the corresponding set of feasible fractional solutions to LP_{MISR} and LP_{MISI}, respectively, is the same.*

Our approximation algorithm for uniform-weight bagMISR works as follows. We compute any maximal feasible solution APX_{max} for the bagMISR instance induced by $\mathcal{R}$. Consider the rectangles $\mathcal{R}_{bag} \subseteq \mathcal{R} \setminus APX_{max}$ such that at least one (indeed, precisely one) task in the same bag is contained in APX_{max}.

We apply the algorithm from Lemma 5 with $\mathcal{R}_{max} = APX_{max}$ and $\mathcal{R}' = \mathcal{R} \setminus \mathcal{R}_{bag}$ [12]. This way we obtain the sets $\mathcal{R}_{point}, \mathcal{R}_{top}, \mathcal{R}_{left}, \mathcal{R}_{right}$. For any $x \in \{top, left, right\}$, we consider the instance of (uniform weight) bagMISI induced by $\mathcal{I}_x$ (where the bags are defined by the corresponding bijection). We apply the LP-based 2-approximation algorithm for bagMISI in [6] to this instance, hence obtaining an approximate solution APX_x. Finally, we output the best solution APX among the solutions APX_{max}, APX_{top}, APX_{left}, and APX_{right}.

Lemma 6. *The above algorithm is a 17 approximation for the bagMISR instances induced by large tasks of a bagUFP instance.*

Proof. Let y be the optimal fractional solution to $LP_{bagMISR}$ with weight opt. For $x \in \{top, left, right, bag, point\}$, let y_x be the restriction of y to rectangles $\mathcal{R}_x$, and let opt_x be the corresponding fractional weight.

Suppose that $opt_x \geq 2opt/17$ for some $x \in \{top, left, right\}$. Since y_x is feasible for $LP_{bagMISI}$ on intervals $\mathcal{I}_x$, then $|APX_x| \geq opt_x/2 \geq opt/17$.

Suppose next that $opt_{bag} \geq opt/17$. Let $\mathcal{B}_j$ be a bag corresponding to some task $i \in APX_{max}$. The total weight in y_{bag} for this bag is at most 1 by the bag constraints. Therefore $|APX_{max}| \geq opt_{bag} \geq opt/17$.

Finally, assume $opt_{point} \geq 10opt/17$. Next let $\mathcal{R}_p \subseteq \mathcal{R}_{point}$ be the rectangles containing some point $p \in \mathcal{P}_i$ for some rectangle $R_i \in APX_{max}$. The optimal

[12] Observe that, by construction, APX_{max} is a maximal independent set w.r.t $\mathcal{R}'$. This might not be the case w.r.t. $\mathcal{R}$ since bag constraints might prevent some non-overlapping rectangle to be included in the maximal solution.

fractional weight associated to $\mathcal{R}_p$ is at most 1 due to the LP constraints. Therefore, $opt_{point} \leq \sum_{R_i \in APX_{max}} \sum_{p \in \mathcal{P}_i} 1 \leq 10 \sum_{R_i \in APX_{max}} 1 = 10|APX_{max}|$. Thus $|APX_{max}| \geq opt_{point}/10 \geq opt/17$. □

Theorem 2 follows from Lemma 6 and the above discussion.

Acknowledgements. The authors wish to thank Andreas Wiese for very helpful discussions about UFP and related problems.

References

1. A. Anagnostopoulos, F. Grandoni, S. Leonardi, and A. Wiese. Constant integrality gap LP formulations of unsplittable flow on a path. In *IPCO*, pages 25–36, 2013.
2. A. Anagnostopoulos, F. Grandoni, S. Leonardi, and A. Wiese. A mazing 2+ε approximation for unsplittable flow on a path. In *SODA*, pages 26–41, 2014.
3. E. Asplund and B. Grünbaum. On a coloring problem. *Math. Scand*, 8:181–188, 1960.
4. N. Bansal, A. Chakrabarti, A. Epstein, and B. Schieber. A quasi-PTAS for unsplittable flow on line graphs. In *STOC*, pages 721–729, 2006.
5. N. Bansal, Z. Friggstad, R. Khandekar, and R. Salavatipour. A logarithmic approximation for unsplittable flow on line graphs. In *SODA*, pages 702–709, 2009.
6. A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Approximating the throughput of multiple machines in real-time scheduling. *SIAM J. Comput.*, 31(2):331–352, 2001.
7. J. Batra, N. Garg, A. Kumar, T. Mömke, and A. Wiese. New approximation schemes for unsplittable flow on a path. In *SODA*, pages 47–58, 2015.
8. P. Bonsma, J. Schulz, and A. Wiese. A constant factor approximation algorithm for unsplittable flow on paths. In *FOCS*, pages 47–56, 2011.
9. G. Calinescu, A. Chakrabarti, H. Karloff, and Y. Rabani. Improved approximation algorithms for resource allocation. In *IPCO*, pages 401–414, 2002.
10. V. T. Chakaravarthy, A. R. Choudhury, S. Gupta, S. Roy, and Y. Sabharwal. Improved algorithms for resource allocation under varying capacity. In *ESA*, pages 222–234, 2014.
11. T. M. Chan and S. Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete Comput. Geom.*, 48(2):373–392, 2012.
12. C. Chekuri, A. Ene, and N. Korula. Unsplittable flow in paths and trees and column-restricted packing integer programs. In *APPROX*, pages 42–55. 2009.
13. C. Chekuri, M. Mydlarz, and F. Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Transactions on Algorithms*, 3, 2007.
14. A. Darmann, U. Pferschy, and J. Schauer. Resource allocation with time intervals. *Theoretical Computer Science*, 411:4217–4234, 2010.
15. F. Grandoni and T. Rothvoß. Pricing on paths: A PTAS for the highway problem. In *SODA*, pages 675–684, 2011.
16. F. Spieksma. On the approximability of an interval scheduling problem. *Journal of Scheduling*, 2(5):215–227, 1999.