# Approximation Algorithms for the Unsplittable Flow Problem on Paths and Trees

**Khaled Elbassioni[1], Naveen Garg[2], Divya Gupta[3], Amit Kumar[2], Vishal Narula[4], and Arindam Pal[2]**

1 **MPI-Informatik, Germany**
2 **IIT Delhi, India**
3 **UCLA, USA**
4 **Goldman Sachs, India**

## Abstract

We study the Unsplittable Flow Problem (UFP) and related variants, namely UFP with Bag Constraints and UFP with Rounds, on paths and trees. We provide improved constant factor approximation algorithms for all these problems under the *no bottleneck assumption* (NBA), which says that the maximum demand for any source-sink pair is at most the minimum capacity of any edge. We obtain these improved results by expressing a feasible solution to a natural LP relaxation of the UFP as a near-convex combination of feasible integral solutions.

**1998 ACM Subject Classification** F.2.2 Nonnumerical Algorithms and Problems

**Keywords and phrases** Approximation Algorithms, Integer Decomposition, Linear Programming, Scheduling, Unsplittable Flows

**Digital Object Identifier** 10.4230/LIPIcs.FSTTCS.2012.267

## 1 Introduction

In this paper, we give new results for several variants of the Unsplittable Flow Problem on paths and trees. The setting for all of these problems is as follows: we are given a graph $G = (V, E)$, where $G$ is either a path or a tree, with edge capacities $c_e$ for each edge $e \in E$, a set of demands $D_1, \ldots, D_m$, where each demand $D_i$ consists of a source-sink pair $s_i, t_i$, a bandwidth requirement $d_i$, and a profit $w_i$. In order to route a demand $D_i$, we send $d_i$ amount of flow from $s_i$ to $t_i$ along the (unique) path between them in $G$. A set of demands is said to be *feasible* if they can be simultaneously routed without violating any edge capacity. In the Max-UFP problem, we would like to find a feasible subset of demands of maximum total profit. In the Round-UFP problem, we would like to color the demands with minimum number of colors such that demands with a particular color form a feasible subset. Another interesting variant is the Bag-UFP problem, where we are given sets $\mathcal{D}_1, \ldots, \mathcal{D}_k$, where each set (or bag) $\mathcal{D}_i$ consists of a set of demands, and has an associated profit (the individual demands in each set do not have profits, though they could have different bandwidth requirements). A solution needs to pick at most one demand from each bag such that these demands are feasible. The goal is to maximize the total profit of bags from which a demand is picked.

All of the above problems are NP-Hard (even for a path), and there has been lot of recent work on obtaining constant factor approximation algorithms for them. An assumption often made in these settings is the so-called *no-bottleneck assumption*: the maximum bandwidth requirement of any demand is at most the minimum edge capacity, i.e., $\max_i d_i \leq \min_e c_e$. Obtaining constant factor approximation algorithms for the above problems without the

32nd Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012).
Editors: D. D'Souza, J. Radhakrishnan, and K. Telikepalli; pp. 267–275

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

*no-bottleneck assumption* remains a challenging task; the only exception being the recent result of Bonsma et al. [4] which gives a constant factor approximation algorithm for MAX-UFP on the line. We will assume that the *no-bottleneck assumption* holds in subsequent discussions.

MAX-UFP and BAG-UFP are weakly NP-Hard, since they contain the KNAPSACK problem as a special case, where there is just a single edge. Recently, it has been proved that the problem is strongly NP-hard, even for the restricted case where all demands are chosen from $\{1, 2, 3\}$ and all capacities are uniform [4]. However, the problem is not known to be APX-hard, so a polynomial time approximation scheme (PTAS) may still be possible. For the special case of (KNAPSACK), an FPTAS is well-known. When all capacities, demands and profits are 1, MAX-UFP specializes to MAX-EDP, the maximum edge-disjoint paths problem.

Chakrabarti et al. [7] gave the first constant approximation algorithm for MAX-UFP on paths and the approximation ratio was subsequently improved to $2 + \varepsilon$, for any constant $\varepsilon > 0$ by Chekuri et al. [8]. They also gave a constant factor approximation algorithm for MAX-UFP on trees. These algorithms are based on the idea of rounding a natural LP relaxation of the MAX-UFP problem.

ROUND-UFP is NP-Hard, since it contains the BIN PACKING problem as a special case, where there is just a single edge. BIN PACKING is known to be APX-hard. However, it has an asymptotic polynomial time approximation scheme (APTAS). There are also simple greedy algorithms like *first-fit* and *best-fit*, which give constant-factor approximations (see e.g. [1]). When all capacities and demands are 1, ROUND-UFP reduces to the interval coloring problem on paths, for which a simple greedy algorithm gives the optimal coloring.

The ROUND-UFP problem for paths has been well-studied in the context of on-line algorithms as well. Here the intervals arrive in arbitrary order, and we need to assign them a color on their arrival so that all intervals with one color form a feasible packing, i.e. total demand on any edge does not exceed its capacity. When all capacities and demands are 1, i.e. when no two intersecting intervals can be given the same color, the first-fit algorithm achieves a constant competitive ratio ([15, 16, 18]). Kierstead and Trotter [14] gave a different online algorithm which uses at most $3\omega - 2$ colors. They also proved that any deterministic online algorithm in the worst case will require at least $3\omega - 2$ colors. For the case of arbitrary edge capacities and demands with the no-bottleneck assumption (NBA), which is same as ROUND-UFP, Epstein et al. [12] gave a 78-competitive algorithm. Prior to our work, this was the best known result for ROUND-UFP in the off-line setting as well.

The BAG-UFP problem was introduced by Chakaravarthy et al. [6], who gave an $O\left(\log\left(\frac{c_{\max}}{c_{\min}}\right)\right)$-approximation algorithm. Here $c_{\max}$ and $c_{\min}$ are the maximum and minimum edge capacities of the path respectively. They gave the first constant factor approximation algorithm for the BAG-UFP problem on paths – the approximation ratio is 120. A related problem is the *job interval selection* problem for which Chuzhoy et al. [11] gave an $\left(\frac{e}{e-1}\right)$-approximation algorithm. See also Erlebach et al. [13] for some additional results.

## 1.1 Our Contributions

In this paper, we give a unified framework for these problems. We give a simple algorithm for ROUND-UFP on paths. We use this to give a constant factor approximation algorithm for MAX-UFP as well. The idea is to start with a natural LP relaxation for MAX-UFP. We show that using our algorithm for ROUND-UFP, one can express a fractional solution to the LP as a convex combination of integer solutions (up to a constant factor). This

idea generalizes to Bag-UFP as well. This leads to improved approximation algorithms for several of these problems. More specifically, our results are:

- We give a 24-approximation algorithm for Round-UFP on paths. This is much simpler than the 78-competitive algorithm of [12], and gives an improved approximation ratio.
- We give a 65-approximation algorithm for Bag-UFP on paths, thus improving the constant approximation factor of 120 given by Chakaravarthy et al. [5] for this problem.
- For trees, we give the first constant factor approximation algorithm for Round-UFP – the approximation factor is 64.

## 1.2 Other Related Work

Recently, Bonsma et al. [4] gave the first constant factor approximation algorithm for Max-UFP on a path without assuming NBA. They also proved that the problem is strongly NP-hard, even for the restricted case where all demands are chosen from $\{1, 2, 3\}$ and all capacities are uniform.

The round version of Bag-UFP is hard to approximate, because *scheduling jobs with interval constraints* is a special case of this problem. In the latter problem, we have a collection of jobs, where each job has a set of intervals associated with it. We can schedule a job in any of the intervals from its set. The goal is to color the jobs with minimum number of colors, such that the set of jobs with a particular color are feasible, i.e., one can pick an interval from the set associated with each job, such that these intervals are disjoint. Chuzhoy et al. [10] proved that it is NP-hard to get better than $O(\log \log n)$-approximation algorithm for this problem. In the continuous version of this problem, the intervals associated with a job form a continuous time segment, described by a release date and a deadline. Chuzhoy and Codenotti [9] gave a constant factor approximation algorithm for the continuous version.

## 1.3 Organization of the Paper

In Section 2, we define the problems considered in this paper. We give a constant factor approximation algorithm for Round-UFP on paths in Section 3. In Section 4, we use the ideas developed in Section 3 to get a constant factor approximation algorithm for Max-UFP on paths, which we then extend to Bag-UFP on paths in Section 5. In Section 6 we give a constant factor approximation algorithm for Round-UFP on trees.

# 2 Preliminaries

We formally define the problems considered in this paper. In all of these problems, an instance will consist of a graph $G = (V, E)$, which is either a path or a tree, with edge capacities $c_e$ for all edges $e \in E$. In case of Round-UFP and Max-UFP, we are also given a set of demands $D_1, \ldots, D_m$. Demand $D_i$ has an associated source-sink pair, $(s_i, t_i)$, a bandwidth requirement $d_i$ and a profit $w_i$. We shall use $I_i$ to denote the associated unique path between $s_i$ and $t_i$ in $G$ (in case of a path, we shall also call this an interval). A subset of demands will be called *feasible* if they can be routed without violating the edge capacities. In Max-UFP, the goal is to find a feasible subset of demands of maximum total profit. In Round-UFP, the goal is to partition the set of demands into minimum number of colors, such that demands with a particular color are feasible.

Finally, we define the Bag-UFP problem. We will consider this problem for the case of paths only. Here, we are given sets, which we will call *bags*, $\mathcal{D}^1, \ldots, \mathcal{D}^k$, where each set

$\mathcal{D}^j$ consists of a set of demands $D_1^j, \ldots, D_{n_j}^j$. As before, each demand $D_i^j$ is specified by an interval $I_i^j$ and a bandwidth requirement $d_i^j$. We are also given profits $p^j$ associated with each of the bags $\mathcal{D}^j$. A feasible solution to such an instance picks at most one demand from each of the bags – the selected demands should form a feasible set of routable demands. The profit of such a solution is the total profit of the bags from which we select a demand. The goal is to maximize the total profit.

We require our instances to satisfy the so called *no-bottleneck assumption*. This assumption states that $\max_i d_i \leq \min_e c_e$, where $i$ varies over all the demands, and $e$ varies over all the edges in $G$. We now give some definitions which will be used by all the algorithms. We will divide the set of demands into two classes – large and small demands.

▶ **Definition 1.** The *bottleneck capacity* $b_i$ of a demand $D_i$ is the smallest capacity of an edge in the (unique) path between $s_i$ and $t_i$ – such an edge is called the *bottleneck edge* for demand $D_i$. A demand $D_i$ is said to be *small* if $d_i \leq \frac{1}{4}b_i$, else it is a *large* demand.

## 3 Approximation Algorithm for Round-UFP

We consider an instance $\mathcal{I}$ of the Round-UFP problem given by a path $G$ on $n$ points, and a set of demands $D_1, \ldots, D_m$ as described in Section 2. Let $\mathcal{O}$ denote an optimal solution, and $\texttt{col}(\mathcal{O})$ denote the number of colors used by $\mathcal{O}$. We begin with a few definitions.

▶ **Definition 2.** The *congestion* of an edge $e$, $r_e$, is defined as $\left\lceil \frac{\sum_{i:e\in I_i} d_i}{c_e} \right\rceil$, i.e., the ratio of the total demand through the edge $e$ to its capacity. Let $r_{\max} = \max_e r_e$ be the *maximum congestion* on the path.

▶ **Definition 3.** The *clique size* on an edge $e$, $L_e$, is defined as the number of large demands containing the edge $e$. Let $L_{\max} = \max_e L_e$ be the *maximum clique size* on the path.

Clearly, $\texttt{col}(\mathcal{O}) \geq r_{\max}$. We give an algorithm $\mathcal{A}$ which uses $O(r_{\max})$ colors. This will give a constant factor approximation algorithm for this problem. We first consider the case of large demands.

▶ **Lemma 4.** *We can color all large demands with at most $L_{\max}$ colors. Further, $L_{\max} \leq 8\texttt{col}(\mathcal{O})$.*

**Proof.** We will use the following result of Nomikos et al. [17].

▶ **Lemma 5.** *[17] Consider an instance of* Round-UFP *where all capacities are integers and all demands $D_i$ have bandwidth requirement $d_i = 1$. Then, one can color these demands with $r_{\max}$ colors.*

We first scale all capacties and demand requirements such that $c_{\min}$ becomes equal to 1. Now, we round all capacities down to the nearest integer, and we scale all the demand requirements $d_i$ to 1. Note that this will affect the congestion of an edge $e$ by a factor of at most 8 – since $c_e$ was at least 1, rounding it down to the nearest integer will reduce it by a factor of at most 1/2. Since all demands were of size at least 1/4 (because they are large demands), we may increase the requirement of a demand by factor of at most 4. Thus, the value of $r_{\max}$ will increase by factor of at most 8. Now, we invoke the result in Lemma 5. This proves the lemma. ◀

We now consider the more non-trivial case of small demands. We divide the edges into classes based on their capacities. We say that an edge $e$ is of *class* $l$ if $2^l \le c_e < 2^{l+1}$. We use $\texttt{cl}(e)$ to denote the class of $e$. For a demand $D_j$, let $l_j$ be the smallest class such that the interval $I_j$ contains an edge of class $l_j$. The *critical edge* of demand $D_j$ is defined as the first edge (as we go from left to right from $s_j$ to $t_j$) in $I_j$ of class $l_j$. Note that the critical edge could be different from the bottleneck edge, though both of them would be of class $l_j$.

▶ **Lemma 6.** *The small demands can be colored with at most* $16r_{\max}$ *colors.*

**Proof.** We maintain $16r_{\max}$ different solutions to the instance $\mathcal{I}$, where a solution routes a subset of the demands. We will be done if we can assign each demand to one of these solutions. Let us call these solutions $\mathcal{S}_1, \ldots, \mathcal{S}_K$, where $K = 16r_{\max}$. We first describe the routing algorithm and then show that it has the desired properties.

We arrange the demands in order of their left end-points – let this ordering be $D_1, \ldots, D_m$. Let $e_j$ be the critical edge of $D_j$. When we consider $D_j$, we send it to a solution $\mathcal{S}_l$ for which the total requirements of demands containing $e_j$ is at most $c_{e_j}/16$. At least one such solution must exist, otherwise $r_e > \frac{16r_{\max} \cdot c_{e_j}/16}{c_{e_j}} = r_{\max}$, a contradiction. This completes the description of how we assign each demand to one of the solutions. We now prove that each of the solutions $\mathcal{S}_l$ is feasible.

Fix a solution $\mathcal{S}_l$ and an edge $e$. Suppose $e$ is of class $i$. Let $\mathcal{D}(\mathcal{S}_l)$ be the demands routed in $\mathcal{S}_l$ which contain the edge $e$. Among such demands, let $D_u$ be the last demand for which the critical edge is to the left of $e$ (including $e$) – let $e'$ be such an edge. Clearly, $\texttt{cl}(e') \ge i$. For an integer $i' \le i$, let $e^{(i')}$ be the first edge of class $i'$ to the right of $e$ (so, $e^{(i)}$ is same as $e$).

First consider the demands in $\mathcal{D}(\mathcal{S}_l)$ which are considered before (and including $D_u$). All of these demands go through $e'$ (because all such demands begin before $D_u$ does and contain $e$). So, the total requirement of such demands, excluding $D_u$, is at most $c_{e'}/16$ – otherwise we would not have assigned $D_u$ to this solution. Because $D_u$ is a small demand and $\texttt{cl}(e') \ge i$, the total requirements of such demands (including $D_u$) is at most

$$\frac{2^{i+1}}{16} + \frac{c_e}{4} \le \frac{c_e}{8} + \frac{c_e}{4} = \frac{3c_e}{8}.$$

Now consider the demands in $\mathcal{D}(\mathcal{S}_l)$ whose critical edges are to the right of $e$ – note that, such an edge must be one of $e^{(i')}$ for some $i' < i$. Similar to the argument above, the total requirements of such demands is at most

$$\sum_{i'<i} \left( \frac{2^{i'+1}}{16} + \frac{2^{i'+1}}{4} \right) \le \frac{5 \cdot 2^{i+1}}{16} \le \frac{5c_e}{8}.$$

Thus, we see that the total requirements of demands in $\mathcal{D}(\mathcal{S}_l)$ is at most

$$\frac{5c_e}{8} + \frac{3c_e}{8} \le c_e.$$

Hence the solution is feasible. This proves the lemma. ◀

Combining the above two lemmas, we get the following theorem.

▶ **Theorem 7.** *Given an instance of* Round-UFP*, there is an algorithm for this problem which uses at most* $24 \cdot \texttt{col}(\mathcal{O})$ *colors, and hence it is a* 24*-approximation algorithm. Further, if all demands are small, then one can color the demands using at most* $16 \cdot \texttt{col}(\mathcal{O})$ *colors.*

## 4 Approximation Algorithms for Max-UFP

In this section we show how ideas from Round-UFP can be used to derive a constant factor approximation algorithm for Max-UFP. Consider an instance $\mathcal{I}$ of Max-UFP. As before, we divide the demands into small and large demands. For large demands, Chakrabarti et al. [7] showed that one can find the optimal solution by dynamic programming.

▶ **Lemma 8.** *[7] The number of $\delta$-large demands crossing any edge in a feasible solution is at most $\frac{2}{\delta}\left(\frac{1}{\delta}-1\right)$. Hence, an optimum solution can be found in $n^{O(1/\delta^2)}$ time using dynamic programming.*

Note that, according to our definition, large demands are $\frac{1}{4}$-large. Now we consider the small demands. The following lemma gives an approximation algorithm for small demands.

▶ **Lemma 9.** *If there are only small jobs, then there is a* 16*-approximation algorithm for* Max-UFP*.*

**Proof.** We write the following natural LP relaxation for this problem – a variable $x_i$ for demand $D_i$ which is 1 if we include it in our solution, and 0 otherwise.

$$\max \sum_i w_i x_i$$

$$\sum_{i:e\in I_i} d_i x_i \le c_e \quad \text{for all edges } e \tag{1}$$

$$0 \le x_i \le 1 \quad \text{for all demands } i$$

Let $x^\star$ be an optimal solution to the LP relaxation. Let $K$ be an integer such that all the variables $x_i^\star$ can be written as $\frac{\alpha_i}{K}$ for some integer $\alpha_i$. Now we construct an instance $\mathcal{I}'$ of Round-UFP as follows. For each (small) demand $D_i$ in $\mathcal{I}$, we create $\alpha_i$ copies of it. Rest of the parameters are same as those in $\mathcal{I}$. First observe that inequality (1) implies that $\sum_{i:e\in I_i} d_i\alpha_i \le Kc_e, \forall e \in E$. Thus, the congestion of each edge in $\mathcal{I}'$ is at most $K$. Using Lemma 6 for small demands, we can color the demands with at most $16K$ colors. It follows that the best solution among these $16K$ solutions will have profit at least $\frac{1}{16}\cdot\sum_i w_i x_i^\star$. ◀

Thus, we get the following theorem.

▶ **Theorem 10.** *There is a* 17*-approximation algorithm for the* Max-UFP *problem.*

**Proof.** Given an instance $\mathcal{I}$, we divide the demands into large and small demands. For large demands, we compute the optimal solution using Lemma 8, whereas for small demands we compute a solution with approximation ratio 16 using Lemma 9. Then we pick the better of the two solutions.

Consider an optimal solution $\mathcal{O}$ with profit $\texttt{profit}(\mathcal{O})$. Let $\texttt{profit}^l(\mathcal{O})$ be the profit for large demands and $\texttt{profit}^s(\mathcal{O})$ be the profit for small demands. If $\texttt{profit}^l(\mathcal{O}) \ge \frac{1}{17}\cdot\texttt{profit}(\mathcal{O})$, then our solution for large demands will also be at least $\frac{1}{17}\cdot\texttt{profit}(\mathcal{O})$. Otherwise, $\texttt{profit}^s(\mathcal{O}) \ge \frac{16}{17}\cdot\texttt{profit}(\mathcal{O})$. In this case, our solution for small demands will have value at least $\frac{1}{16}\cdot\frac{16}{17}\cdot\texttt{profit}(\mathcal{O}) = \frac{1}{17}\cdot\texttt{profit}(\mathcal{O})$. ◀

## 5 Approximation Algorithms for Bag-UFP

We now extend the above algorithm to the Bag-UFP problem. Consider an instance $\mathcal{I}$ of this problem. As before, we classify each of the demands $D_i^j$ as either large or small. For each bag, $\mathcal{D}^j$, let $\mathcal{D}^{j,\mathtt{l}}$ be the set of large demands in $\mathcal{D}^j$ and $\mathcal{D}^{j,\mathtt{s}}$ be the set of small demands in $\mathcal{D}^j$. Again, we have two different strategies for large and small demands.

▶ **Lemma 11.** *If there are only large jobs, then there is a* 48*-approximation algorithm for* Bag-UFP*.*

**Proof.** Suppose we have the further restriction that the selected intervals need to be disjoint. Lemma 8 implies that this will worsen the objective value by a factor of at most 24. However, for the latter problem, we can use the 2-approximation algorithm of Berman et al. [3] and Bar-Noy et al. [2]. This gives a 48-approximation algorithm. ◀

▶ **Lemma 12.** *If there are only small jobs, then there is a* 17*-approximation algorithm for* Bag-UFP*.*

**Proof.** As in the case of Max-UFP problem, we first write an LP relaxation, and then use an algorithm similar to the one used for the Round-UFP problem. We have a variable $x_i^j$ for demand $D_i^j$, which is 1 if we include it in our solution and 0 otherwise, and a variable $y^j$ which is 1 if we choose a demand from the bag $\mathcal{D}^j$ and 0 otherwise. The LP relaxation is as follows.

$$\max \sum_j p^j y^j$$

$$\sum_{i:e\in I_i^j} d_i^j x_i^j \le c_e \quad \text{for all edges } e \tag{2}$$

$$\sum_i x_i^j \le y^j \quad \text{for all bags } \mathcal{D}^j \tag{3}$$

$$0 \le x_i^j \le 1 \quad \text{for all demands } i$$

$$0 \le y^j \le 1 \quad \text{for all bags } \mathcal{D}^j$$

Let $x, y$ be an optimal solution to the LP above. Again, let $K$ be a large enough integer such that $y^j = \frac{\alpha^j}{K}, x_i^j = \frac{\beta_i^j}{K}$, where $\alpha_j$ and $\beta_i^j$ are integers for all $j$ and $i$. Now we consider an instance of Round-UFP where we have $\beta_i^j$ copies of the demand $D_i^j$. The only further restriction is that no two demands from the same bag can get the same color. Inequality (2) implies that $\sum_{i:e\in I_i^j} d_i^j \beta_i^j \le K c_e, \forall e \in E$. So the congestion bound is $K$. We proceed as in the proof of Lemma 6, except that now we have $17K$ different solutions. When we consider the demand $D_i^j$, we ignore the solutions which contain a demand from the bag $\mathcal{D}^j$. Inequality (3) implies that $\sum_i \beta_i^j \le \alpha^j \le K, \forall j$. Hence, there will be at most $K$ such solutions. For the remaining $16K$ solutions, we argue as in the proof of Lemma 6. ◀

▶ **Theorem 13.** *There is a* 65*-approximation algorithm for the* Bag-UFP *problem.*

**Proof.** This follows from the two previous lemmas. We argue as in the proof of Theorem 10. ◀

## 6 Approximation Algorithms for Round-UFP on Trees

We now consider the Round-UFP problem on trees. Consider an instance $\mathcal{I}$ of this problem as described in Section 2. We consider the case of large and small demands separately. Let $\mathcal{D}^{\mathsf{l}}$ be the set of large demands and $D^{\mathsf{s}}$ be the set of small demands.

▶ **Lemma 14.** *There is a* 32*-approximation algorithm for the above instance when we only have demands in* $\mathcal{D}^{\mathsf{l}}$*.*

**Proof.** Chekuri et al. [8] gave a 4-approximation algorithm for coloring a set of demands when all demands have requirement 1, and the capacities are integers. In fact, their algorithm uses at most $4r_{\max}$ colors. We can reduce our problem to this case by losing an extra factor of 8 in $r_{\max}$ – we proceed exactly as in the proof of Lemma 4. ◀

▶ **Lemma 15.** *There is a* 32*-approximation algorithm for the above instance when we only have demands in* $\mathcal{D}^{\mathbf{s}}$*.*

**Proof.** The proof is very similar to that of Lemma 6. We maintain $16r_{\max}$ solutions. For a demand $D_i$, let $a_i$ denote the least common ancestor of $s_i$ and $t_i$. We consider the demands in a bottom-up order of $a_i$. For a demand $D_i$, we define two critical edges: the $s_i$-critical edge is the critical edge on the $a_i - s_i$ path, and the $t_i$-critical edge is the critical edge on the $a_i - t_i$-path. We send $D_i$ to the solution in which both these critical edges have been used till $\frac{1}{16}$ of their total capacity only. Again it is easy to check that such a solution will exist. The rest of the argument now follows as in the proof of Lemma 6. ◀

Combining the above two lemmas, we get

▶ **Theorem 16.** *There is a* 64*-approximation algorithm for the* Round-UFP *problem on trees.*

## 7 Conclusion and Open Problems

In this paper, we studied the Unsplittable Flow Problem and some of its variants, such as UFP with Bag Constraints and UFP with Rounds. We gave improved constant factor approximation algorithms for all these problems under the *no bottleneck assumption*. One important open question is, can we improve the approximation factors further? A related question is, are there lower bounds (hardness results, bad examples or integrality gap examples) for these problems matching these upper bounds? Another important open problem is the approximability of these problems without NBA. For Max-UFP on paths, a $(7 + \varepsilon)$-approximation is known, but for the other problems the question is not settled.

## References


1 *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

2 Amotz Bar-Noy, Sudipto Guha, Joseph Naor, and Baruch Schieber. Approximating the throughput of multiple machines under real-time scheduling. In *STOC*, pages 622–631, 1999.

3 Piotr Berman and Bhaskar DasGupta. Improvements in throughout maximization for real-time scheduling. In *STOC*, pages 680–687, 2000.

4 Paul Bonsma, Jens Schulz, and Andreas Wiese. A constant factor approximation algorithm for unsplittable flow on paths. In *FOCS*, pages 47–56, 2011.

5 Venkatesan T. Chakaravarthy, Anamitra R. Choudhury, and Yogish Sabharwal. A near-linear time constant factor algorithm for unsplittable flow problem on line with bag constraints. In *FSTTCS*, pages 181–191, 2010.

6 Venkatesan T. Chakaravarthy, Vinayaka Pandit, Yogish Sabharwal, and Deva P. Seetharam. Varying bandwidth resource allocation problem with bag constraints. In *IPDPS*, pages 1–10, 2010.

7 Amit Chakrabarti, Chandra Chekuri, Anupam Gupta, and Amit Kumar. Approximation algorithms for the unsplittable flow problem. *Algorithmica*, 47(1):53–78, 2007.

8 Chandra Chekuri, Marcelo Mydlarz, and F. Bruce Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Transactions on Algorithms*, 3(3), 2007.

9 Julia Chuzhoy and Paolo Codenotti. Resource minimization job scheduling. In *APPROX-RANDOM*, pages 70–83, 2009.

10 Julia Chuzhoy and Joseph Naor. New hardness results for congestion minimization and machine scheduling. *J. ACM*, 53(5):707–721, 2006.

11 Julia Chuzhoy, Rafail Ostrovsky, and Yuval Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. *Math. Oper. Res.*, 31(4):730–738, 2006.

12 Leah Epstein, Thomas Erlebach, and Asaf Levin. Online capacitated interval coloring. *SIAM J. Discrete Math.*, 23(2):822–841, 2009.

13 Thomas Erlebach and Frits C. R. Spieksma. Interval selection: Applications, algorithms, and lower bounds. *J. Algorithms*, 46(1):27–53, 2003.

14 H.A. Kierstead and W.T. Trotter. An extremal problem in recursive combinatorics. *Congressus Numerantium*, 33:143–153, 1981.

15 Hal A. Kierstead. The linearity of first-fit coloring of interval graphs. *SIAM J. Discrete Math.*, 1(4):526–530, 1988.

16 Hal A. Kierstead and Jun Qin. Coloring interval graphs with first-fit. *Discrete Mathematics*, 144(1-3):47–57, 1995.

17 Christos Nomikos, Aris Pagourtzis, and Stathis Zachos. Routing and path multicoloring. *Inf. Process. Lett.*, 80(5):249–256, 2001.

18 Sriram V. Pemmaraju, Rajiv Raman, and Kasturi R. Varadarajan. Max-coloring and online coloring with bandwidths on interval graphs. *ACM Transactions on Algorithms*, 7(3):35, 2011.