

Truthful Unsplittable Flow for Large Capacity Networks*

Yossi Azar[†]

Iftah Gamzu[‡]

Shai Gutner[§]

Abstract

The *unsplittable flow problem* is one of the most extensively studied optimization problems in the field of networking. An instance of it consists of an edge capacitated graph and a set of connection requests, each of which is associated with source and target vertices, a demand, and a value. The objective is to route a maximum value subset of requests subject to the edge capacities. It is a well known fact that as the capacities of the edges are larger with respect to the maximal demand among the requests, the problem can be approximated better. In particular, it is known that for sufficiently large capacities, the integrality gap of the corresponding integer linear program becomes $1 + \epsilon$, which can be matched by an algorithm that utilizes the randomized rounding technique.

In this paper, we focus our attention on the large capacities unsplittable flow problem in a game theoretic setting. In this setting, there are selfish agents, which control some of the requests characteristics, and may be dishonest about them. It is worth noting that in game theoretic settings many standard techniques, such as randomized rounding, violate certain monotonicity properties, which are imperative for truthfulness, and therefore cannot be employed. In light of this state of affairs, we design a monotone deterministic algorithm, which is based on a primal-dual machinery, which attains an approximation ratio of $\frac{e}{e-1}$, up to a disparity of ϵ away. This implies an improvement on the current best truthful mechanism, as well as an improvement on the current best combinatorial algorithm for the problem under consideration. Surprisingly, we demonstrate that any algorithm in the family of reasonable iterative path minimizing algorithms, cannot yield a better approximation ratio. Consequently, it follows that in order to achieve a monotone PTAS, if exists, one would have to exert different techniques. We also consider the large capacities *single-minded multi-unit combinatorial auction problem*. This problem is closely related to the unsplittable flow problem since one can formulate it as a special case of the integer linear program of the unsplittable flow problem. Accordingly, we obtain a comparable performance guarantee by refining the algorithm suggested for the unsplittable flow problem.

*An extended abstract of this paper appeared in *Proceedings of the 19th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 320–329, 2007.

[†]Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA and School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. Supported in part by the German-Israeli Foundation and by the Israel Science Foundation. Email: azar@post.tau.ac.il.

[‡]School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. Email: iftgam@post.tau.ac.il.

[§]School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. Email: gutner@post.tau.ac.il.

1 Introduction

The problems. We study the *unsplittable flow problem*. As input to this problem, we are given a directed or undirected graph $G = (V, E)$, such that $n = |V|$, $m = |E|$, and every edge $e \in E$ has a positive capacity c_e . An additional ingredient of the input is a set \mathcal{R} of connection requests, in which every request $r \in \mathcal{R}$ is characterized by a quadruple (s_r, t_r, d_r, v_r) such that s_r and t_r are the respective *source* and *target* vertices of the request, d_r is the positive *demand* associated with the request, and v_r is the positive *value* or profit gained as a result of allocating the request. The objective is to select a maximum value subset of requests $S \subseteq \mathcal{R}$, along with a path for each selected request, so that all the requests in S can simultaneously route their demand along the corresponding path, while preserving the capacity constraints. Denoting by $B = \min_e \{c_e\} / \max_r \{d_r\}$ the ratio between the minimal capacity of an edge and the maximal demand among the requests, the problem is referred to as the *B-bounded unsplittable flow problem*. Since one can normalize both the demands of the requests and the capacities of the edges, the B-bounded unsplittable flow problem can be equivalently defined to have $d_r \in (0, 1]$ for every request r , and $B = \min_e \{c_e\}$. Note that we shall use the latter definition throughout this paper.

We also consider the *single-minded multi-unit combinatorial auction problem*. This problem is closely related to the unsplittable flow problem since one can formulate it as a special case of the integer linear program of the unsplittable flow problem. An instance of it consists of a set U of m non-identical items, such that item $u \in U$ has a positive integer multiplicity c_u . The input also consists of a set \mathcal{R} of requests, in which every request $r \in \mathcal{R}$ is characterized by a pair (U_r, v_r) such that $U_r \subseteq U$ is an items bundle, which is the *demand* associated with the request, and v_r is the positive *value* gained as a result of allocating the bundle. The goal is to select a maximum value subset $S \subseteq \mathcal{R}$, so that every item $u \in U$ appears in at most c_u bundles of requests in S . Denoting by $B = \min_u \{c_u\}$ the minimum multiplicity of an item, the problem is referred to as the *B-bounded multi-unit combinatorial auction problem*.

The setting. In the present paper, we study the $\Omega(\ln m)$ -*bounded unsplittable flow problem*, and the $\Omega(\ln m)$ -*bounded multi-unit combinatorial auction problem* from a *mechanism design* [15] point of view. In this game-theoretic setting, some characteristics of the requests, which are henceforth referred to as the *type* of the requests, are controlled by *selfish agents*. An agent is selfish in a sense that it might declare a fallacious type in order to manipulate the algorithm in a way that will maximize its own utility. Our goal is to design mechanisms, which are referred to as *incentive compatible* or *truthful*, in which each agent's best strategy is always to reveal the true type of the request that it controls, regardless of the other requests types, and regardless of the way that the other agents decide to declare their requests types. In particular, we aim to devise *monotone* algorithms which are, roughly speaking, equivalent to truthful mechanisms. Note that in the unsplittable flow problem, the type of a request is its demand and value, whereas in the multi-unit combinatorial auction problem the type of a request is its value. Also note that the other characteristics of the request, e.g. the source and target vertices in the unsplittable flow problem, are assumed to be known and thus, the agent cannot be untruthful about them.

The motivation. One of the closely related problems to the unsplittable flow problem is the *multicommodity flow problem*. Since the multicommodity flow problem can be modelled by the relaxation of the integer linear program of the unsplittable flow problem, it may be considered as its fractional version. It is well known that the integrality gap of the integer linear program of the unsplittable flow problem becomes $1 + \epsilon$ when the ratio between the minimal capacity of an edge and the maximal demand among the requests is sufficiently large. Consequently, it is conjectured that in such case, the performance of algorithms for the fractional and integral versions would be similar. Specifically, since the multicommodity flow problem admits a monotone PTAS by combinatorial primal-dual based algorithms [10, 9], one may expect that an integral version of these monotone PTAS would yield a monotone PTAS for the unsplittable flow problem. In

the following, we refute this perception. In particular, we design a primal-dual based monotone algorithm for the $\Omega(\ln m)$ -bounded unsplittable flow problem that attains the best possible approximation ratio with respect to any *reasonable iterative path minimizing* algorithm¹, and is not a PTAS. Nevertheless, This algorithm still improves over the best previous known result [7].

1.1 Our results

Monotone deterministic algorithms. We devise a monotone deterministic algorithm based on a primal-dual approach for the $\Omega(\ln m)$ -bounded unsplittable flow problem, which obtains an approximation ratio that approaches $\frac{e}{e-1} \approx 1.58$. This result implies a corresponding incentive compatible mechanism. In addition, we show that the aforesaid algorithm can be attuned for the $\Omega(\ln m)$ -bounded multi-unit combinatorial auction problem, and attain a comparable approximation ratio, i.e. $\frac{e}{e-1}$ -approximation. These results improve over the approximation guarantee suggested by Briest et al. [7] for both problems, which approaches $e \approx 2.71$.

Deterministic lower bounds. We prove that any algorithm for the $\Omega(\ln m)$ -bounded unsplittable flow problem, which is part of the reasonable iterative path minimizing algorithms family, cannot yield an approximation guarantee that is better than $\frac{e}{e-1} - o(1)$. This implies, on the one hand, that the analysis of our algorithm is tight, and on the other hand, that to achieve a monotone deterministic PTAS, if exists, one would have to employ different techniques. Additionally, we reinforce this inapproximability result by demonstrating that even if we ease the problem setting, e.g. assume that the minimal capacity of an edge is arbitrarily large, still no reasonable iterative path minimizing algorithm can attain PTAS. Correspondingly, we also establish a lower bound of $\frac{4}{3}$ on the approximation ratio of any reasonable iterative bundle minimizing algorithm for the $\Omega(\ln m)$ -bounded multi-unit combinatorial auction problem.

A deterministic $(1 + \epsilon)$ -approximation algorithm. We study the $\Omega(\ln m)$ -bounded *unsplittable flow with repetitions problem*, which is a variant of the $\Omega(\ln m)$ -bounded unsplittable flow problem in which one is allowed to satisfy every request multiple times using possibly multiple paths. In contrast with our prior findings, we demonstrate that this version admits a deterministic primal-dual based algorithm, which yields an $(1 + \epsilon)$ -approximation.

1.2 Related work

The unsplittable flow problem, and its fractional variant, the multicommodity flow problem, has been given an extensive attention in recent years, both from an algorithmic point of view and from a game-theoretic one. These fundamental optimization problems model a diverse collection of applications in network routing, parallel computing, and even in VLSI layout. Obviously, the fractional problem is easier. In particular, it is known to admit an optimal solution by linear programming, and an $(1 + \epsilon)$ -approximate solution by combinatorial algorithms, see e.g. [10, 9]. In contrast with the fractional problem, approximating the integral problem is hard. Chuzhoy et al. [8] have recently showed that the directed version of the problem is $n^{\Omega(1/B)}$ -hard to approximate unless $\text{NP} \subseteq \text{BPTIME}(n^{O(\log \log n)})$, where $B = O(\log n / \log \log n)$ is the minimal capacity of an edge. This result extended the prominent result of Guruswami et al. [11], which states that if $B = 1$, it is NP-hard to approximate the problem to within a factor of $O(n^{1/2-\epsilon})$. Respectively, when the graph is undirected, Andrews et al. [1] established an $(\log n)^{\Omega(1/B)}$ -hardness for any $B = O(\log \log n / \log \log \log n)$, under the assumption that $\text{NP} \not\subseteq \text{ZPTIME}(n^{\text{polylog}(n)})$. Nevertheless, when B is sufficiently large, e.g. $B = \Omega(\ln m)$, the integrality gap of the integer linear program of the problem becomes $1 + \epsilon$, which can be matched by an algorithm that utilizes the randomized rounding technique [17, 16, 18]. Unfortunately, this standard technique violates certain monotonicity properties, which are imperative for truthfulness and thus, cannot be directly

¹The family of *reasonable iterative path minimizing* algorithms is formally defined in Subsection 3.3.

used in the presence of selfish agents to obtain a truthful mechanism. Accordingly, until recently, the known truthful results for the $\Omega(\ln m)$ -bounded unsplittable flow problem only guaranteed approximation ratios that were at least logarithmic in the size of the graph [5, 6, 4]. Briest et al. [7] seem to have been the first to propose a constant factor approximation algorithm. Essentially, they designed a monotone primal-dual based algorithm, which was motivated by the novel work of Garg and Könemann [10] on the fractional problem, that achieves an approximation guarantee that approaches e .

The research of the single-minded multi-unit combinatorial auction problem, which is closely related to the unsplittable flow problem, yielded similar results. Bartal et al. [6] showed that approximating the problem to within a factor of $O(m^{1/(B+1)})$ is NP-hard, where B is the minimum multiplicity of an item. Yet, when $B = \Omega(\ln m)$, the integrality gap of the corresponding integer linear program becomes $1 + \epsilon$. Accordingly, Archer et al. [2], and Lavi and Swamy [12] devised truthful $(1 + \epsilon)$ -approximation mechanisms. However, these mechanisms are truthful only in a probabilistic sense and hence, the best known deterministic truthful result for the $\Omega(\ln m)$ -bounded multi-unit combinatorial auction problem is by Briest et al. [7], which attains e -approximation.

2 Preliminaries

In what follows, we present the notions of *monotonicity* and *exactness*, and then turn to describe a characterization that reduces the goal of designing truthful mechanisms to that of designing monotone and exact algorithms. Remark that the illustrated terms are presented in the context of the problems under considerations and hence, the keen reader may refer to Lehmann et al. [13] or Briest et al. [7] for more formal and comprehensive definitions of the underlying concepts.

Definition 2.1. An algorithm \mathcal{A} for the unsplittable flow problem is said to be *monotone* w.r.t. the demand and value of a request $r \in \mathcal{R}$, if it satisfies the following property: if algorithm \mathcal{A} selects r when its demand is d_r and its value is v_r then algorithm \mathcal{A} would have selected r if its demand was $\tilde{d}_r \leq d_r$, its value was $\tilde{v}_r \geq v_r$, and the demands and values of all the other requests were fixed.

Definition 2.2. An algorithm \mathcal{A} for the unsplittable flow problem is called *exact*, if it allocates the exact demand of every request selected, and does not allocate anything otherwise.

Theorem 2.3. ([13, 7]) *If algorithm \mathcal{A} for the unsplittable flow problem is monotone and exact w.r.t. the demand and value of every request then there exists a corresponding truthful mechanism. In addition, this mechanism can be efficiently computed using algorithm \mathcal{A} .*

Note that similar definitions can analogously be made for the single-minded multi-unit combinatorial auction problem. The only exception is that the monotonicity property, and the characterization theorem are only defined with respect to the value of every request, i.e. the demand terms need to be cast off.

3 Unsplittable Flow Problem

3.1 The algorithm

In this subsection, we devise a deterministic monotone algorithm for the $\Omega(\ln m)$ -bounded unsplittable flow problem, which achieves an approximation ratio that approaches $\frac{e}{e-1}$. Our algorithm is based on a primal-dual machinery. Accordingly, we present in Figure 1, the primal-dual formulation of the unsplittable flow problem. This will be later used to motivate the algorithm.

$\begin{aligned} \max \quad & \sum_{r \in \mathcal{R}} v_r \cdot \left(\sum_{s \in S_r} x_s \right) \\ \text{s.t.} \quad & \sum_{s \in S e \in s} x_s d_s \leq c_e \quad \forall e \in E \\ & \sum_{s \in S_r} x_s \leq 1 \quad \forall r \in \mathcal{R} \\ & x_s \in \{0, 1\} \quad \forall s \in S \end{aligned}$	$\begin{aligned} \min \quad & \sum_{e \in E} c_e y_e + \sum_{r \in \mathcal{R}} z_r \\ \text{s.t.} \quad & z_r + d_r \sum_{e \in S} y_e \geq v_r \quad \forall r \in \mathcal{R}, \forall s \in S_r \\ & y_e \geq 0 \quad \forall e \in E \\ & z_r \geq 0 \quad \forall r \in \mathcal{R} \end{aligned}$
---	--

Figure 1: The integer linear program of the unsplittable flow problem (left), and the dual of its relaxation (right). Note that S_r denotes the set of all the simple paths between s_r and t_r in G , $S = \bigcup_{r \in \mathcal{R}} S_r$, and d_s and v_s denote the respective demand and value of path s , i.e. if $s \in S_r$ then $d_s = d_r$ and $v_s = v_r$.

Algorithm Bounded-UFP, formally described below, is a primal-dual based algorithm for the $\Omega(\ln m)$ -bounded unsplittable flow problem. Informally, the algorithm maintains the variables of the primal and dual programs, and in each iteration selects to satisfy a request, which corresponds to the “most violated” constraint of the dual linear program. Favorably, this reduces to finding a (normalized) shortest path in the graph G , whose edge weights correspond to the set of dual variables y_e . It is worth noting that the algorithm, and part of its analysis is in the spirit of the algorithm suggested by Briest et al. [7].

Algorithm 1 Bounded-UFP(ϵ)

Input: An accuracy parameter $\epsilon \in (0, 1]$
Output: A (request, path) pairs set \mathcal{W} , which holds the requests to be allocated

- 1: Let \mathcal{L} be a list of all the requests, and let \mathcal{W} be an empty set
- 2: **for all** $r \in \mathcal{L}$ **do** $z_r = 0$ **end for**
- 3: **for all** $s \in \mathcal{S}$ **do** $x_s = 0$ **end for**
- 4: **for all** $e \in E$ **do** $y_e = \frac{1}{c_e}$ **end for**
- 5: **while** ($\mathcal{L} \neq \emptyset$ and $\sum_{e \in E} c_e y_e \leq e^{\epsilon(B-1)}$) **do**
- 6: **for all** $r \in \mathcal{L}$ **do**
- 7: Let p_r be the shortest path between s_r and t_r in G with respect to the weights y_e , and let $|p_r| = \sum_{e \in p_r} y_e$ be its length
- 8: **end for**
- 9: Let \hat{r} be the request, which minimizes $\frac{d_r}{v_r} |p_r|$ with respect to every $r \in \mathcal{L}$
- 10: **for all** $e \in p_{\hat{r}}$ **do** $y_e = y_e \cdot e^{\epsilon B d_{\hat{r}} / c_e}$ **end for**
- 11: Add $(\hat{r}, p_{\hat{r}})$ to \mathcal{W} , and remove \hat{r} from \mathcal{L}
- 12: Let $x_{p_{\hat{r}}} = 1$ and $z_{\hat{r}} = v_{\hat{r}}$
- 13: **end while**
- 14: **return** \mathcal{W}

We would like to note that since the path related variables, and the request related variables, i.e. the x_s and z_r variables respectively, play no role in the execution of the algorithm, lines 2, 3, and 12 are not regarded part of the algorithm. Nevertheless, we decided not to neglect them from the algorithm’s description since they ease the analysis presentation.

3.2 Analysis

In the remainder of this subsection, we will prove the following theorem.

Theorem 3.1. *Algorithm Bounded-UFP($\frac{\epsilon}{6}$) returns a feasible $((1+\epsilon)\frac{e}{e-1})$ -approximate solution for the $\Omega(\frac{\ln m}{e^2})$ -bounded unsplittable flow problem, for any $\epsilon \in (0, 1]$, runs in polynomial-time, and is monotone and exact w.r.t. the demand and value of every request.*

Corollary 3.2. *There exists a polynomial-time truthful $((1+\epsilon)\frac{e}{e-1})$ -approximation mechanism for the $\Omega(\ln m)$ -bounded unsplittable flow problem, for any $\epsilon \in (0, 1]$, where every request's demand and value is unknown.*

We begin by introducing a notation that ease the analysis presentation:

- Let x_s^i , y_e^i , and z_r^i be the respective values of the variables x_s , y_e , and z_r at the end of the i -th iteration of the algorithm, where $i \geq 0$. Mind that we regard the end of iteration 0 as the beginning of the algorithm. Additionally, we let (y^i, z^i) denote the set of dual variables at the end of the i -th iteration.
- Let $P(i) = \sum_{r \in \mathcal{R}} v_r \cdot (\sum_{s \in \mathcal{S}_r} x_s^i)$ be the value of the primal solution at the end of the i -th iteration, and let P be the value of the primal solution when the algorithm terminates. Notice that P is the sum of values of requests selected to be allocated by the algorithm, i.e. the outcome of the algorithm. In addition, we let $\Delta P(i) = P(i) - P(i-1)$ be the value in which the primal solution is incremented in the i -th iteration.
- Let $D_1(i) = \sum_{e \in E} c_e y_e^i$ and $D_2(i) = \sum_{r \in \mathcal{R}} z_r^i$ be the respective values of the first and second parts of the dual solution at the end of the i -th iteration, and let $D(i) = D_1(i) + D_2(i)$. Also, let D denote the value of the optimal solution for the dual linear program.
- Let $\alpha(i)$ denote the normalized length of the path selected after the end of the i -th iteration. Note that if path p is selected in the $(i+1)$ -th iteration then $\alpha(i) = \frac{d_p}{v_p} |p| = \frac{d_p}{v_p} \sum_{e \in p} y_e^i$.

Correctness. In what follows, we prove that the algorithm outputs a feasible solution. Namely, we demonstrate that the edges capacity constraints are not violated.

Lemma 3.3. *Algorithm Bounded-UFP outputs a feasible solution.*

Proof. Assume by contradiction that the output of the algorithm is not feasible. Let \bar{p} be the first path that induces a violation in the capacity of edge e in the ℓ -th iteration, and let \mathcal{P} be the family of paths selected before the ℓ -th iteration, which consist of e . Since \bar{p} induces a capacity violation then $\sum_{p \in \mathcal{P}} d_p + d_{\bar{p}} > c_e$. Clearly, since $d_{\bar{p}} \in (0, 1]$ it follows that $\sum_{p \in \mathcal{P}} d_p > c_e - 1$. Consequently, we get that

$$c_e y_e^{\ell-1} = c_e y_e^0 \prod_{p \in \mathcal{P}} e^{\frac{\epsilon B d_p}{c_e}} = e^{\frac{\epsilon B}{c_e} \sum_{p \in \mathcal{P}} d_p} > e^{\epsilon B \frac{c_e - 1}{c_e}} \geq e^{\epsilon B \frac{B-1}{B}} = e^{\epsilon(B-1)},$$

where the last inequality results from the fact that $\frac{x-1}{x}$ is an increasing monotonic function for all $x \geq 1$, and since $c_e \geq B \geq 1$. Inspecting the main loop stopping condition, i.e. line 5 in the algorithm, it follows that the algorithm had to exit the loop. This implies that the algorithm could not have executed the ℓ -th iteration and thus, could not have selected \bar{p} , a contradiction. ■

Approximation. We now turn to prove that the algorithm yields an approximation ratio that approaches $\frac{e}{e-1}$. We begin by establishing an algebraic claim, which will be utilized later.

Claim 3.4. *Given an increasing sequence $\{\alpha_0, \alpha_1, \dots, \alpha_{t+1}, \alpha\}$, $\sum_{i=0}^t \left(\frac{\alpha_{i+1} - \alpha_i}{\alpha - \alpha_i}\right) \leq \ln\left(\frac{\alpha - \alpha_0}{\alpha - \alpha_{t+1}}\right)$.*

Proof. For every $0 \leq i \leq t$,

$$\frac{\alpha_{i+1} - \alpha_i}{\alpha - \alpha_i} \leq \ln \left(\frac{\alpha - \alpha_i}{\alpha - \alpha_{i+1}} \right) = \ln(\alpha - \alpha_i) - \ln(\alpha - \alpha_{i+1}),$$

where the inequality follows from $\ln(1+x) \leq x$ by substituting $x = \frac{\alpha_i - \alpha_{i+1}}{\alpha - \alpha_i}$. Accordingly, this implies that

$$\sum_{i=0}^t \left(\frac{\alpha_{i+1} - \alpha_i}{\alpha - \alpha_i} \right) \leq \sum_{i=0}^t (\ln(\alpha - \alpha_i) - \ln(\alpha - \alpha_{i+1})) = \ln(\alpha - \alpha_0) - \ln(\alpha - \alpha_{t+1}) = \ln \left(\frac{\alpha - \alpha_0}{\alpha - \alpha_{t+1}} \right).$$

■

The next claim upper bounds $\alpha(i)$, the normalized length of the path selected in the $(i+1)$ -th iteration.

Claim 3.5. $\alpha(i) \leq \frac{D_1(i)}{D - D_2(i)}$, in every iteration $i \geq 0$.

Proof. Consider the $(i+1)$ -th iteration. Let p denote the path that is selected in this iteration. The path p corresponds to an unselected request such that $\frac{d_p}{v_p} \sum_{e \in p} y_e^i$ is minimal. Namely, every other path p' , which corresponds to another unselected request, satisfies

$$\frac{d_{p'}}{v_{p'}} \sum_{e \in p'} y_e^i \geq \frac{d_p}{v_p} \sum_{e \in p} y_e^i = \alpha(i), \text{ or equivalently } d_{p'} \sum_{e \in p'} \frac{y_e^i}{\alpha(i)} \geq v_{p'}.$$

This implies that if we multiply y_e^i by $\alpha(i)^{-1}$, for every $e \in E$, then all the dual linear program constraints become satisfied. Consequently, the set of variables $(y^i \alpha(i)^{-1}, z^i)$ constitutes a feasible fractional solution to the dual linear program and therefore, $D \leq D_1(i) \alpha(i)^{-1} + D_2(i)$. ■

We continue by bounding the change in $D_1(i)$, which is the first part of the dual solution at the end of the i -th iteration, and part of the stopping criteria of the algorithm.

Claim 3.6. $D_1(i+1) \leq D_1(i) + B\epsilon(1+\epsilon) \cdot \Delta P(i+1) \cdot \alpha(i)$, for every $i \geq 0$.

Proof. Consider the $(i+1)$ -th iteration. Let p denote the path that is selected in this iteration, and let d_p and v_p denote its respective demand and value. Inspecting the algorithm, one can derive that

$$\begin{aligned} \sum_{e \in E} c_e y_e^{i+1} &= \sum_{\substack{e \in E \\ e \notin p}} c_e y_e^i + \sum_{e \in p} c_e y_e^i \cdot e^{\frac{\epsilon B d_p}{c_e}} \\ &\leq \sum_{\substack{e \in E \\ e \notin p}} c_e y_e^i + \sum_{e \in p} c_e y_e^i \cdot \left(1 + \frac{\epsilon B d_p}{c_e} + \left(\frac{\epsilon B d_p}{c_e} \right)^2 \right) \\ &\leq \sum_{e \in E} c_e y_e^i + B\epsilon(1+\epsilon) d_p \sum_{e \in p} y_e^i \\ &= \sum_{e \in E} c_e y_e^i + B\epsilon(1+\epsilon) \cdot \Delta P(i+1) \cdot \alpha(i). \end{aligned}$$

The first inequality is due to the fact that $e^a \leq 1 + a + a^2$ for any $a \in [0, 1]$, and the fact that $\frac{\epsilon B d_p}{c_e} \in (0, 1]$. The second inequality holds since

$$c_e \cdot \left(\frac{\epsilon B d_p}{c_e} + \left(\frac{\epsilon B d_p}{c_e} \right)^2 \right) \leq \epsilon B d_p + \epsilon^2 B d_p = B\epsilon(1+\epsilon) d_p, \quad (1)$$

where the inequality in (1) follows from the observations that $d_p^2 \leq d_p$, and $\frac{B}{c_e} \leq 1$. Finally, the last equality follows from the definition of $\alpha(i)$, which can be rewritten as $d_p \sum_{e \in E} c_e y_e^i = v_p \alpha(i)$, and the observation that v_p is the value in which the primal solution is incremented in the $(i+1)$ -th iteration. Recalling that $D_1(i) = \sum_{e \in E} c_e y_e^i$ completes the proof. \blacksquare

We are now ready to prove that the algorithm achieves almost $\frac{\epsilon}{e-1}$ -approximation for the problem under consideration.

Lemma 3.7. *Algorithm Bounded-UFP(ϵ) returns an $((1+6\epsilon)\frac{\epsilon}{e-1})$ -approximate solution for the $\frac{\ln m}{\epsilon^2}$ -bounded unsplittable flow problem, for any $\epsilon \in (0, \frac{1}{6}]$.*

Proof. One can easily notice, by inspecting the stopping condition of the main loop, that when the algorithm terminates, either $\mathcal{L} = \emptyset$ or $\sum_{e \in E} c_e y_e > e^{\epsilon(B-1)}$. If $\mathcal{L} = \emptyset$ then it follows that the algorithm succeeded to satisfy all the requests and thus, its output is optimal. Consequently, in the remainder of the proof, we shall assume that $\sum_{e \in E} c_e y_e > e^{\epsilon(B-1)}$. For every $i \geq 0$,

$$\begin{aligned} D_1(i+1) &\leq D_1(i) + B\epsilon(1+\epsilon) \cdot \Delta P(i+1) \cdot \alpha(i) \\ &\leq D_1(i) + B\epsilon(1+\epsilon) \cdot \Delta P(i+1) \cdot \frac{D_1(i)}{D - D_2(i)} \\ &= D_1(i) \left(1 + B\epsilon(1+\epsilon) \frac{\Delta P(i+1)}{D - D_2(i)} \right) \\ &\leq D_1(i) e^{\left(B\epsilon(1+\epsilon) \frac{\Delta P(i+1)}{D - D_2(i)} \right)}, \end{aligned}$$

where the first and second inequalities follow from Claim 3.6 and Claim 3.5, respectively, and the last inequality is due to the fact that $1 + a \leq e^a$. This implies that

$$D_1(i+1) \leq D_1(0) e^{\left(B\epsilon(1+\epsilon) \sum_{j=0}^i \frac{\Delta P(j+1)}{D - D_2(j)} \right)} \leq e^{\left(B\epsilon^2 + B\epsilon(1+\epsilon) \sum_{j=0}^i \frac{\Delta P(j+1)}{D - D_2(j)} \right)},$$

where the first inequality results from the expansion of the recursion, and the second follows from $D_1(0) \leq e^{B\epsilon^2}$, which is obtained by noticing that $D_1(0) = \sum_{e \in E} c_e y_e^0 = m$, and recalling that $B \geq \frac{\ln m}{\epsilon^2}$ by definition. Lets assume that the algorithm terminates after $t+1$ iterations. Accordingly, using our prior assumption that $\sum_{e \in E} c_e y_e > e^{\epsilon(B-1)}$, we get that

$$e^{\epsilon(B-1)} < D_1(t+1) \leq e^{\left(B\epsilon^2 + B\epsilon(1+\epsilon) \sum_{j=0}^t \frac{\Delta P(j+1)}{D - D_2(j)} \right)}.$$

One can validate that $D_2(j) = P(j)$ by inspecting the variables alterations in line 12 of the algorithm, and their affect on $D_2(j)$ and $P(j)$. Hence, we derive that $\frac{\Delta P(j+1)}{D - D_2(j)} = \frac{P(j+1) - P(j)}{D - P(j)}$. Consequently, we can apply Claim 3.4, while recalling that $P(0) = 0$ and $P(t+1) = P$, and yield

$$\epsilon(B-1) < B\epsilon^2 + B\epsilon(1+\epsilon) \ln \left(\frac{D}{D-P} \right).$$

Because $\frac{1-2\epsilon}{1+\epsilon} \leq \frac{\epsilon(B-1) - B\epsilon^2}{B\epsilon(1+\epsilon)}$, and since $1-3\epsilon \leq \frac{1-2\epsilon}{1+\epsilon}$ for any positive ϵ , we attain $1-3\epsilon \leq \ln \left(\frac{D}{D-P} \right)$. This can be simplified further to give $\frac{D}{P} \leq \frac{1}{e^{(1-3\epsilon)} - 1} + 1 \leq (1+6\epsilon)\frac{\epsilon}{e-1}$. Recall that D is the value of the optimal solution for the dual linear program and thus, using the weak LP duality completes the proof. \blacksquare

Truthfulness. The following lemma establishes the monotonicity and exactness of the algorithm.

Lemma 3.8. *Algorithm Bounded-UFP is monotone and exact w.r.t. the demand and value of every request.*

Proof. Consider a request r selected to be routed using path p_r in the ℓ -th iteration of the algorithm, which has a respective demand and value of d_r and v_r . Now, suppose that r had a demand of $\tilde{d}_r \leq d_r$, a value of $\tilde{v}_r \geq v_r$, and the demands and values of all the other requests were fixed. For the sake of monotonicity, we need to prove that the algorithm would have selected r in the latter case, i.e. when its demand and value were \tilde{d}_r and \tilde{v}_r , respectively. If r is selected by the algorithm in the first $\ell - 1$ iterations then we are done. Otherwise, let's consider the ℓ -th iteration. One can easily observe that in the first $\ell - 1$ iterations of the algorithm, the same set of requests is selected to be routed using the same set of paths whether the demand and value of r is (d_r, v_r) or $(\tilde{d}_r, \tilde{v}_r)$. Respectively, the same set of unselected requests remain. Note that $\frac{\tilde{d}_r}{\tilde{v}_r} \leq \frac{d_r}{v_r}$. Hence, since the path p_r minimizes $\frac{d_p}{v_p} \sum_{e \in p} y_e$ over any path p , which corresponds to an unselected request, when the demand and value of r is (d_r, v_r) , so it does when the demand and value of r is $(\tilde{d}_r, \tilde{v}_r)$. This implies that r must be selected by the algorithm in the ℓ -th iteration.

The exactness of the algorithm is clear, as the algorithm may route the exact demand of every request selected, and may not route anything otherwise. ■

We are now ready to prove the main theorem of this subsection.

Proof of Theorem 3.1. The correctness of the algorithm is due to Lemma 3.3, the approximation guarantee is established in Lemma 3.7, and the monotonicity and exactness are presented in Lemma 3.8. Finally, it is clear that the running time of the algorithm is polynomial, and in fact, if we denote the number of requests by $|\mathcal{R}|$, one can easily validate that the number of iterations is bounded by $|\mathcal{R}|$, and every iteration takes time proportional to $|\mathcal{R}|$ shortest path computations. ■

3.3 Inapproximability result

In the following, we introduce two input instances for the problem under consideration that lower bound the performance guarantee of any algorithm, which is part of the reasonable iterative path minimizing algorithms family. Specifically, the first input instance proves that any such algorithm cannot yield an approximation guarantee better than $\frac{e}{e-1} - o(1)$. This demonstrates that the analysis of algorithm Bounded-UFP is tight, and that it is the “best” algorithm in the aforementioned family of algorithms. The second input instance establishes a lower bound of $\frac{4}{3}$ on the approximation ratio of any such algorithm for this problem in its utmost generality, i.e. when the underlying graph is undirected and the minimal edge capacity is arbitrarily large. In particular, this suggests that even if we ease the problem setting, e.g. assume that the minimal capacity of an edge is $\Omega(m)$ instead of $\Omega(\ln m)$, no algorithm in the aforesaid family can achieve PTAS.

Prior to describing the finer details of our approach, we introduce the notion of a reasonable function, which is a key ingredient in the definition of a reasonable iterative path minimizing algorithm. Note that reasonable functions have a similar flavor to the min functions introduced by Archer and Tardos [3]. Nevertheless, they are still quite different. Let $S = \bigcup_{r \in \mathcal{R}} \{p : p \text{ is a simple path between } s_r \text{ and } t_r\}$.

Definition 3.9. Let $g : S \rightarrow \mathbb{R}$ be a function, which assigns an arbitrary priority to every path. Such a function is called *reasonable* if under the assumption that the capacities of all the edges are identical, and both the demand and value of every request are unit, it follows that $g(q) \leq g(q')$ for any valid unsplittable flow, and any two paths $q, q' \in S$ that satisfy

- q consists of k edges, q' consists of k' edges, and $k \leq k'$.
- $f_i \leq f'_i$, for every $1 \leq i \leq k$, where (f_1, f_2, \dots, f_k) and $(f'_1, f'_2, \dots, f'_{k'})$ are non-increasing vectors, which indicate the flow routed through the edges of q and q' with respect to the valid unsplittable flow.

Definition 3.10. An algorithm is referred to as *reasonable iterative path minimizing algorithm*, if it iteratively selects a path that minimizes a reasonable function over all the paths that correspond to unselected requests.

One can verify that algorithm **Bounded-UFP** minimizes the function $h(p) = \frac{d_p}{v_p} \sum_{e \in p} \frac{1}{c_e} e^{\frac{Bf_e}{c_e}}$, where f_e denotes the flow routed through edge e . This function is reasonable since when we assume that both the demand and value of every request are unit, and the capacities of all the edges are identical, say B , then it reduces to $\tilde{h}(p) = \frac{1}{B} \sum_{e \in p} e^{cf_e}$, which clearly satisfies $\tilde{h}(q) \leq \tilde{h}(q')$, for any two paths q and q' that meet the properties indicated in Definition 3.9. Consequently, algorithm **Bounded-UFP** is a reasonable iterative path minimizing algorithm. We note that reasonability captures a broad class of functions. For example, the reasonable function $h_1(p) = \ln(1 + |p|) \cdot h(p)$ is similar to the function used by algorithm **Bounded-UFP** but it is mildly biased towards paths with less edges. Another example of a reasonable function is $h_2(p) = \frac{d_p}{v_p} \prod_{e \in p} \frac{f_e}{c_e}$ although it is not clear why anyone would like to use it. We are now ready to establish the main result of this subsection.

Theorem 3.11. *The approximation ratio of any reasonable iterative path minimizing algorithm for the unsplittable flow problem when the graph is directed cannot be better than $\frac{e}{e-1} - o(1)$, for $B = o(m^{1/2})$.*

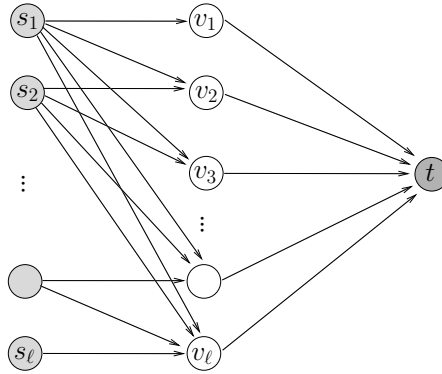


Figure 2: A directed graph in which every vertex s_i has a directed edge to every vertex v_j such that $j \geq i$. Additionally, the capacities of all the edges are identical and equal to B .

Proof. Suppose we are given the directed graph $G = (V, E)$ schematically described in Figure 2, and the set of requests is

$$\mathcal{R} = \left\{ \underbrace{(s_1, t, 1, 1)}_{B \text{ requests}}, \underbrace{(s_2, t, 1, 1)}_{B \text{ requests}}, \dots, \underbrace{(s_\ell, t, 1, 1)}_{B \text{ requests}} \right\}.$$

In order to simplify the presentation and analysis of the lower bound instance, we introduce the following assumption, which will be tackled later. We assume that when there is more than one path, which minimizes the value of the reasonable function used by the iterative path minimizing algorithm, the algorithm selects one of them arbitrarily. Accordingly, we premise that it selects a path (s_i, v_j, t) in which i is *minimal*, and j is *maximal* with respect to all the minimizing paths that their source vertex is s_i . For example, in the initial B iterations of the algorithm, all the requests that their terminal vertices are (s_1, t) are satisfied using paths that use vertices v_j such that $j = \ell, \dots, \ell - B + 1$. In the subsequent B iterations, all the requests that their terminal vertices are (s_2, t) are satisfied using paths that use vertices v_j such that $j = \ell - B, \dots, \ell - 2B + 1$, and so on. Simulating the execution of a reasonable iterative path minimizing algorithm, while ignoring integrality issues that will be resolved later, we get that

- For any integer $1 \leq q \leq B$, at the end of the first $\ell \sum_{r=1}^q (\frac{B}{B+1})^r$ iterations, all and only the requests, whose terminal vertices are (s_i, t) such that $i \leq \ell \cdot (1 - (\frac{B}{B+1})^q)$, are satisfied, and all the (v_j, t) edges such that $j > \ell \cdot (1 - (\frac{B}{B+1})^q)$ have a flow load of q or equivalently, a residual capacity of $B - q$.
- After $\ell \sum_{r=1}^B (\frac{B}{B+1})^r$ iterations, the algorithm cannot route more requests and thus, it stops².

Consequently, since all and only the requests, whose terminal vertices are (s_i, t) such that $i \leq \ell \cdot (1 - (\frac{B}{B+1})^B)$, are satisfied when the algorithm stops, it follows that the value of the solution that the algorithm outputs is at most $B\ell \cdot (1 - (\frac{B}{B+1})^B) = B\ell \cdot (1 - (1 - \frac{1}{B+1})^B) \leq B\ell \cdot (1 - \frac{1}{e})$. On the other hand, an optimal solution clearly has a value of $B\ell$, e.g. route every request of the form $(s_i, t, 1, 1)$ through the directed path (s_i, v_i, t) . Thus, we get that the approximation ratio of the algorithm cannot be better than $\frac{e}{e-1}$.

We now turn to eliminate the integrality assumption. Namely, in the above analysis, we have assumed that $\ell \sum_{r=1}^q (\frac{B}{B+1})^r$ is integral, for any integer q . This clearly may not be true. However, one can resolve this issue by applying a more careful analysis, and yield that only the requests, whose terminal vertices are (s_i, t) such that $i \leq \ell \cdot (1 - (\frac{B}{B+1})^q) + \sum_{k=0}^{q-1} (\frac{B}{B+1})^k$, may become satisfied. Since $\sum_{k=0}^{B-1} (\frac{B}{B+1})^k \leq B$, it follows that the value of the solution that the algorithm achieves might increase by no more than B^2 , in respect to $B\ell \cdot (1 - \frac{1}{e})$. Since the number of edges in the graph is $m = \ell + \sum_{k=1}^{\ell} k \leq 2\ell^2$, and $B = o(m^{1/2})$, we obtain that $B = o(\ell)$. Consequently, the analysis of the lower bound degrades by at most $o(1)$, i.e. the approximation ratio of any algorithm cannot be better than $\frac{e}{e-1} - o(1)$.

Next, we tackle the decisions assumption. Specifically, we have assumed that the decision of any algorithm between same valued minimizing paths is arbitrary and hence, one may ask if a specific tie-breaking rule can lead to better results. We can resolve this matter by constructing a similar input instance, which forces any algorithm to make similar “bad” decisions. Essentially, one way to achieve it is to replace every (s_i, v_j) edge by a directed path with $i\ell + 1 - j$ edges. The reason that an algorithm makes “bad” decisions on this instance dues to the reasonability property, i.e. any reasonable algorithm “prefers” paths with less edges. Note that this instance supports the same lower bound, but has a somewhat stricter constraint on the value of B , i.e. since $m = O(\ell^4)$, B needs to satisfy $B = o(m^{1/4})$. ■

Next, we demonstrate that even if we ease the problem setting, e.g. allow the minimal capacity of an edge to be arbitrarily large, no reasonable iterative path minimizing algorithm can achieve PTAS.

Theorem 3.12. *The approximation ratio of any reasonable iterative path minimizing algorithm for the unsplittable flow problem cannot be better than $\frac{4}{3}$, for any B , and even when the graph is undirected.*

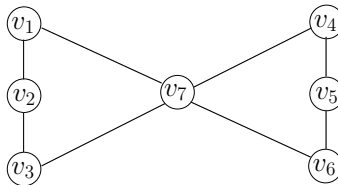


Figure 3: An undirected graph in which the capacities of all the edges are identically equal to B .

²An algorithm might stop even sooner, e.g. algorithm Bounded-UFP stops after the while condition fails. However, analyzing the case that the algorithm stops when it cannot route more requests just affirms the lower bound.

Proof. Suppose we are given the undirected graph $G = (V, E)$ schematically described in Figure 3, and the set of requests is

$$\mathcal{R} = \left\{ \underbrace{(v_1, v_3, 1, 1)}_{B \text{ requests}}, \underbrace{(v_4, v_6, 1, 1)}_{B \text{ requests}}, \underbrace{(v_1, v_6, 1, 1)}_{B \text{ requests}}, \underbrace{(v_3, v_4, 1, 1)}_{B \text{ requests}} \right\}.$$

Clearly, an optimal solution for this instance has a value of $4B$, e.g. route every request of the form $(v_1, v_3, 1, 1)$ through the path (v_1, v_2, v_3) , every request of the form $(v_4, v_6, 1, 1)$ through the path (v_4, v_5, v_6) , every request of the form $(v_1, v_6, 1, 1)$ through the path (v_1, v_7, v_6) , and every request of the form $(v_3, v_4, 1, 1)$ through the path (v_3, v_7, v_4) .

Simulating the execution of a reasonable iterative path minimizing algorithm in the initial four iterations, one can easily validate that the algorithm may select the paths $(v_1, v_7, v_3), (v_4, v_7, v_6), (v_1, v_2, v_3)$, and (v_4, v_5, v_6) since each one of these paths is one of the minimizing paths in the corresponding iteration, and by that satisfy two $(v_1, v_3, 1, 1)$ requests and two $(v_4, v_6, 1, 1)$ requests. In addition, notice that at the end of this four iterations phase, every edge has a residual capacity of $B - 1$. Arguments similar to those used in this initial four iterations phase can be applied in another $\frac{B}{2} - 1$ phases, each of four iterations, to demonstrate that the algorithm acts exactly the same. Consequently, after $\frac{B}{2}$ phases, every edge has a residual capacity of $\frac{B}{2}$, all the $(v_1, v_3, 1, 1)$ requests and the $(v_4, v_6, 1, 1)$ requests were satisfied, and the remaining requests are

$$\bar{\mathcal{R}} = \left\{ \underbrace{(v_1, v_6, 1, 1)}_{B \text{ requests}}, \underbrace{(v_3, v_4, 1, 1)}_{B \text{ requests}} \right\}.$$

In this current state, any algorithm can satisfy at most B requests from $\bar{\mathcal{R}}$. This is the result of the fact that any path from v_1 to v_6 and any path from v_3 to v_4 must use either edge (v_1, v_7) or edge (v_3, v_7) , and the fact that the total residual capacity of these edges sums to B . Thus, the solution that the algorithm outputs has value of at most $3B$. ■

Corollary 3.13. *No reasonable iterative path minimizing algorithm for the unsplittable flow problem can yield a PTAS.*

4 Single-minded Multi-unit Combinatorial Auction Problem

4.1 The algorithm

In this subsection, we design a deterministic monotone algorithm for the $\Omega(\ln m)$ -bounded multi-unit combinatorial auction problem, whose approximation ratio approaches $\frac{e}{e-1}$. Particularly, we begin by demonstrating that the single-minded multi-unit combinatorial auction problem can be formulated as a simplified special case of the integer linear program of the unsplittable flow problem, and then we turn to specialize algorithm Bounded-UFP for the problem under consideration.

The single-minded multi-unit combinatorial auction problem can be formulated as a special case of the integer linear program of the unsplittable flow problem by letting S_r to denote the singleton set of U_r , i.e. $S_r = \{U_r\}$, and replacing e, E and d_s in the integer linear program of the unsplittable flow problem with u, U and 1, respectively. Similarly, the relaxation of the integer linear program of the single-minded multi-unit combinatorial auction problem, and its dual can also be formulated as special cases of the corresponding linear programs. Consequently, the primal-dual algorithm Bounded-MUCA, formally described below, is a specialized version of algorithm Bounded-UFP, in which the path selection procedure, i.e. lines 6-8 in algorithm Bounded-UFP, was neglected, and the demand terms were omitted.

Algorithm 2 Bounded-MUCA(ϵ)

Input: An accuracy parameter $\epsilon \in (0, 1]$

Output: A set \mathcal{W} of requests to be satisfied

- 1: Let \mathcal{L} be a list of all the requests, and let \mathcal{W} be an empty set
 - 2: **for all** $u \in U$ **do** $y_u = \frac{1}{c_u}$ **end for**
 - 3: **while** ($\mathcal{L} \neq \emptyset$ and $\sum_{u \in U} c_u y_u \leq e^{\epsilon(B-1)}$) **do**
 - 4: Let \hat{r} be the request, which minimizes $\frac{1}{v_r} \sum_{u \in U_r} y_u$ with respect to every $r \in \mathcal{L}$
 - 5: **for all** $u \in U_{\hat{r}}$ **do** $y_u = y_u \cdot e^{\epsilon B/c_u}$ **end for**
 - 6: Add \hat{r} to \mathcal{W} , and remove \hat{r} from \mathcal{L}
 - 7: **end while**
 - 8: **return** \mathcal{W}
-

Theorem 4.1. *Algorithm Bounded-MUCA($\frac{\epsilon}{8}$) returns a feasible $((1+\epsilon)\frac{e}{e-1})$ -approximate solution for the $\Omega(\frac{\ln m}{\epsilon^2})$ -bounded multi-unit combinatorial auction problem, for any $\epsilon \in (0, 1]$, runs in polynomial-time, and is monotone and exact w.r.t. the value of each request.*

Proof. Since algorithm Bounded-MUCA is a simplified version of algorithm Bounded-UFP, the analysis of Theorem 3.1 also applies in this case. \blacksquare

It is worth noting that the algorithm can even be employed to a generalized version of the problem in which the demand of every request, i.e. the desired bundle of items, is part of the type of the request and therefore, agents may be dishonest about it. Note that in this case, monotonicity can be easily established by arguments similar to those used in Theorem 3.8, and the additional observation that $\sum_{u \in \tilde{U}_r} y_u \leq \sum_{u \in U_r} y_u$, for any $\tilde{U}_r \subseteq U_r$. Remark that this setting is referred to as the *unknown* single-minded case [14].

Corollary 4.2. *There exists a polynomial-time truthful $((1+\epsilon)\frac{e}{e-1})$ -approximation mechanism for the $\Omega(\ln m)$ -bounded multi-unit combinatorial auction problem among unknown single-minded agents, for any $\epsilon \in (0, 1]$.*

4.2 Inapproximability result

In what follows, we introduce an input instance for the problem under consideration that lower bounds the performance guarantee of any reasonable iterative bundle minimizing algorithm. We demonstrate that any such algorithm cannot achieve an approximation ratio better than $\frac{4}{3}$. Essentially, this suggests that no algorithm in the aforesaid family can achieve PTAS. We start by formally defining the notion of a reasonable iterative bundle minimizing algorithm. We remark that the following definitions are just a refinement of the definitions made for the unsplittable flow problem. Let $S = \bigcup_{r \in \mathcal{R}} \{U_r\}$.

Definition 4.3. Let $g : S \rightarrow \mathbb{R}$ be a function, which assigns an arbitrary priority to every bundle. Such a function is called *reasonable* if under the assumption that the multiplicities of all the items are identical, and the value of every request is unit, it follows that $g(T) \leq g(T')$ for any valid multi-unit allocation³, and any two bundles $T, T' \in S$ that satisfy

- T consists of k items, T' consists of k' items, and $k \leq k'$.
- $f_i \leq f'_i$, for every $1 \leq i \leq k$, where (f_1, f_2, \dots, f_k) and $(f'_1, f'_2, \dots, f'_{k'})$ are non-increasing vectors, which indicate the number of allocated copies of the items of T and T' with respect to the valid multi-unit allocation.

³A *valid multi-unit allocation* can be succinctly described as an allocation of non-identical items to requests such that the number of allocated copies of every item does not exceed its multiplicity.

Definition 4.4. An algorithm is referred to as *reasonable iterative bundle minimizing algorithm*, if it iteratively selects a bundle that minimizes a reasonable function over all the bundles that correspond to unselected requests.

Note that algorithm Bounded-MUCA minimizes the function $h(s) = \frac{1}{v_s} \sum_{u \in s} \frac{1}{c_u} e^{\frac{\epsilon B f_u}{c_u}}$, where f_u denotes the number of allocated copies of item u . One can easily argue that this function is reasonable and thus, algorithm Bounded-MUCA is a reasonable iterative bundle minimizing algorithm. We are now ready to prove the core result of this subsection.

Theorem 4.5. *The approximation ratio of any reasonable iterative bundle minimizing algorithm for the single-minded multi-unit combinatorial auction problem cannot be better than $\frac{4}{3}$.*

Proof. Let m be a multiple of $p \cdot (p + 1)$, where $p \geq 3$ is a constant odd integer. Suppose we are given a set U of m items such that the multiplicities of all the items are identical and equal to B , and let $\bigcup_{i=1}^p \bigcup_{j=1}^{p+1} U_{i,j}$ be a partition of U into $p \cdot (p + 1)$ disjoint sets, each consists of $\frac{m}{p \cdot (p+1)}$ items. Additionally, suppose that \mathcal{R} consists of unit value requests of two types:

1. $\frac{B}{2}$ requests that consist of the items $U_\ell = \bigcup_{j=1}^{p+1} U_{\ell,j}$, for every $\ell = 1, \dots, p$.
2. $\frac{B}{2}$ requests that consist of the items $U_{1,2\ell-1} \cup U_{1,2\ell} \cup \bigcup_{i=2}^p U_{i,2\ell-1}$, for every $\ell = 1, \dots, \frac{p+1}{2}$, and $\frac{B}{2}$ requests that consist of the items $U_{1,2\ell-1} \cup U_{1,2\ell} \cup \bigcup_{i=2}^p U_{i,2\ell}$, for every $\ell = 1, \dots, \frac{p+1}{2}$.

Figure 4 schematically describes a concrete input instance.

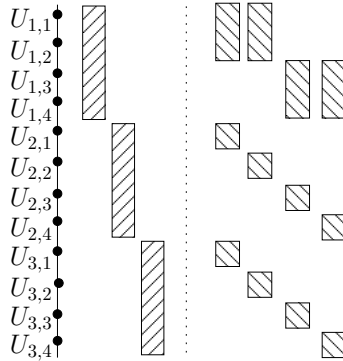


Figure 4: The input instance in the case of $p = 3$, and $m = 12$. Note that the set of items $\bigcup_{i=1}^3 \bigcup_{j=1}^4 U_{i,j}$ are represented by the black dots on the left of the figure, and every collection of $\frac{B}{2}$ requests is represented by a vertical group of rectangles such that each rectangle consists of the set of items, whose representing dots are in the left-projection zone of the rectangle. Also note that the first type of requests are located left of the dotted line, whereas the second type of requests appear on its right.

One can verify that the an optimal solution for this instance has a value of pB , for example by selecting all the requests except for the $\frac{B}{2}$ requests that consist of U_1 .

Simulating the execution of a reasonable iterative bundle minimizing algorithm, one can easily validate that the algorithm may incrementally select all the requests of the first type, e.g. it may repeatedly select a request that consists of U_1 , then a request that consists of U_2 and so on until U_p . Consequently, after $\frac{B}{2}$ phases, each of p steps, all the requests of the first type are satisfied, the value of the current solution is $\frac{p}{2}B$, and every item has a residual multiplicity of $\frac{B}{2}$. Notice that U_1 consists of $\frac{m}{p}$ items, and any request of the second type consists of $\frac{2m}{p \cdot (p+1)}$ items of U_1 . Hence, by simple counting arguments it follows that in the current state, any algorithm cannot satisfy more than $(\frac{B}{2} \cdot \frac{m}{p}) / \frac{2m}{p \cdot (p+1)} = \frac{p+1}{4}B$ requests of the second type. Therefore, we obtain that

the value of the solution that any reasonable iterative bundle minimizing algorithm outputs is no more than $\frac{3p+1}{4}B$ and thus, as p tends to infinity, the inapproximability ratio approaches $\frac{4}{3}$. ■

Corollary 4.6. *No reasonable iterative bundle minimizing algorithm for the single-minded multi-unit combinatorial auction problem yields a PTAS.*

5 Unsplittable Flow with Repetitions Problem

In this section, we study the $\Omega(\ln m)$ -bounded unsplittable flow with repetitions problem. This problem is a variant of the corresponding unsplittable flow problem in which we are allowed to satisfy every request multiple times using possibly multiple paths, and the profit gained is proportional to the number of times that every request is satisfied. In sharp contrast with our prior results, we demonstrate that this version admits a deterministic primal-dual based algorithm, which yields an $(1 + \epsilon)$ -approximation.

5.1 The algorithm

In the following, we contrive an $(1 + \epsilon)$ -approximation algorithm, named **Bounded-UFP-Repeat**, for the $\Omega(\ln m)$ -bounded unsplittable flow with repetitions problem, which is based on a primal-dual approach. Respectively, we present the primal-dual formulation of the underlying problem in Figure 5.

$\max \quad \sum_{r \in \mathcal{R}} v_r \cdot \left(\sum_{s \in S_r} x_s \right)$	$\min \quad \sum_{e \in E} c_e y_e$
$\text{s.t.} \quad \sum_{s \in S e \in s} x_s d_s \leq c_e \quad \forall e \in E$	$\text{s.t.} \quad d_r \sum_{e \in s} y_e \geq v_r \quad \forall r \in \mathcal{R}, \forall s \in S_r$
$x_s \in \mathbb{N} \quad \forall s \in S$	$y_e \geq 0 \quad \forall e \in E$

Figure 5: The integer linear program of the unsplittable flow with repetitions problem (left), and the dual of its relaxation (right). Note that S_r denotes the set of all the simple paths between s_r and t_r in G , $S = \bigcup_{r \in \mathcal{R}} S_r$, and d_s and v_s denote the respective demand and value of path s .

5.2 Analysis

In the remainder of this subsection, we will prove the following theorem.

Theorem 5.1. *Algorithm **Bounded-UFP-Repeat**($\frac{\epsilon}{6}$) is an $(1 + \epsilon)$ -approximation algorithm for the $\Omega(\frac{\ln m}{\epsilon^2})$ -bounded unsplittable flow with repetitions problem, for any $\epsilon \in (0, 1]$, with running time polynomial in m and $\frac{\max_e \{c_e\}}{\min_r \{d_r\}}$.*

For the sake of simplicity, we shall use the same notation, which was introduced in Subsection 3.1, with one exception. Namely, since there are no z -type variables in the dual linear program, we let $D(i) = \sum_{e \in E} c_e y_e^i$ denote the value of the dual solution at the end of the i -th iteration, and neglect $D_1(i)$ and $D_2(i)$. Notice that Lemma 3.3 is applicable also in this case and thus, the correctness of the algorithm follows. Consequently, in the sequel, we prove that the algorithm under consideration achieves an approximation ratio of $(1 + \epsilon)$. We begin by establishing an analogous claim to Claim 3.5, which upper bounds $\alpha(i)$.

Claim 5.2. $\alpha(i) \leq \frac{D(i)}{D}$, in every iteration $i \geq 0$.

Algorithm 3 Bounded-UFP-Repeat(ϵ)

Input: An accuracy parameter $\epsilon \in (0, 1]$

Output: A (request, path) pairs multiset \mathcal{W} , which holds the requests to be allocated

- 1: Let \mathcal{L} be a list of all the requests, and let \mathcal{W} be an empty multiset
 - 2: **for all** $e \in E$ **do** $y_e = \frac{1}{c_e}$ **end for**
 - 3: **while** $(\sum_{e \in E} c_e y_e \leq e^{\epsilon(B-1)})$ **do**
 - 4: **for every** $r \in \mathcal{L}$ **do**
 - 5: Let p_r be the shortest path between s_r and t_r in G with respect to the weights y_e , and let $|p_r| = \sum_{e \in p_r} y_e$ be its length
 - 6: **end for**
 - 7: Let \hat{r} be the request, which minimizes $\frac{d_r}{v_r} |p_r|$ with respect to every $r \in \mathcal{L}$
 - 8: **for all** $e \in p_{\hat{r}}$ **do** $y_e = y_e \cdot e^{\epsilon B d_{\hat{r}} / c_e}$ **end for**
 - 9: Add $(\hat{r}, p_{\hat{r}})$ to \mathcal{W}
 - 10: **end while**
 - 11: **return** \mathcal{W}
-

Proof. Consider the $(i + 1)$ -th iteration, and let p denote the path, which is selected in this iteration. The path p corresponds to a request such that $\frac{d_p}{v_p} \sum_{e \in p} y_e^i$ is minimal. Namely, every other path p' , which corresponds to a request, satisfies

$$\frac{d_{p'}}{v_{p'}} \sum_{e \in p'} y_e^i \geq \frac{d_p}{v_p} \sum_{e \in p} y_e^i = \alpha(i), \text{ or equivalently } d_{p'} \sum_{e \in p'} \frac{y_e^i}{\alpha(i)} \geq v_{p'}.$$

This implies that if we multiply y_e^i by $\alpha(i)^{-1}$, for every $e \in E$, then all the dual linear program constraints become satisfied, i.e. the modified variables constitute a feasible fractional solution to the dual linear program. Hence, $D \leq D(i)\alpha(i)^{-1}$. \blacksquare

Next, we prove that the algorithm achieves $(1 + \epsilon)$ -approximation for the problem under consideration.

Lemma 5.3. *Algorithm Bounded-UFP-Repeat(ϵ) returns an $(1 + 6\epsilon)$ -approximate solution for the $\frac{\ln m}{\epsilon^2}$ -bounded unsplittable flow with repetitions problem, for any $\epsilon \in (0, \frac{1}{6}]$.*

Proof. For every $i \geq 0$, one can derive that

$$\begin{aligned} D(i+1) &\leq D(i) + B\epsilon(1+\epsilon) \cdot \Delta P(i+1) \cdot \alpha(i) \\ &\leq D(i) + B\epsilon(1+\epsilon) \cdot \Delta P(i+1) \cdot \frac{D(i)}{D} \\ &= D(i) \left(1 + B\epsilon(1+\epsilon) \frac{\Delta P(i+1)}{D} \right) \\ &\leq D(i) e^{(B\epsilon(1+\epsilon) \frac{\Delta P(i+1)}{D})}. \end{aligned}$$

The first inequality follows from Claim 3.6, while noticing that $D(i)$ may replace $D_1(i)$. The second inequality is due to Claim 5.2. Finally, the last inequality results from the fact that $1 + a \leq e^a$. This implies that

$$D_1(i+1) \leq D_1(0) e^{(B\epsilon(1+\epsilon) \sum_{j=0}^i \frac{\Delta P(j+1)}{D})} \leq e^{(B\epsilon^2 + B\epsilon(1+\epsilon) \sum_{j=0}^i \frac{\Delta P(j+1)}{D})},$$

where the first inequality results from the expansion of the recursion, and the second follows from $D_1(0) \leq e^{B\epsilon^2}$, which dues to $D_1(0) = m$, and $B \geq \frac{\ln m}{\epsilon^2}$. Lets assume that the algorithm

terminates after $t + 1$ iterations. Accordingly, inspecting the stopping condition of the main loop, we attain that

$$e^{\epsilon(B-1)} < D_1(t+1) \leq e^{\left(B\epsilon^2 + B\epsilon(1+\epsilon) \sum_{j=0}^t \frac{\Delta P(j+1)}{D}\right)}.$$

This can be simplified to $\frac{\epsilon(B-1)-B\epsilon^2}{B\epsilon(1+\epsilon)} < \sum_{j=0}^t \frac{\Delta P(j+1)}{D}$. Notice that $\sum_{j=0}^t \Delta P(j+1)$ is a telescopic sum that is equal to $P(t+1) - P(0) = P$, where the equality dues to $P(t+1) = P$ and $P(0) = 0$. In addition, notice that $1 - 3\epsilon \leq \frac{\epsilon(B-1)-B\epsilon^2}{B\epsilon(1+\epsilon)}$ for any positive ϵ . Consequently, we obtain $1 - 3\epsilon \leq \frac{P}{D}$. Simplifying this expression even further yields $\frac{D}{P} \leq \frac{1}{1-3\epsilon} \leq 1 + 6\epsilon$, which by the weak LP duality establishes the lemma. ■

We are now ready to prove the main theorem of this subsection.

Proof of Theorem 5.1. As noted before, the correctness of the algorithm directly follows from Lemma 3.3. In addition, the approximation guarantee is proved in Lemma 5.3. We now turn to argue that the running time of the algorithm is polynomial in m and $\frac{c_{\max}}{d_{\min}}$, where $c_{\max} = \max_e \{c_e\}$, and $d_{\min} = \min_r \{d_r\}$. Consider some edge $e \in E$. Recall that $y_e^0 = \frac{1}{c_e}$ and $y_e^t \leq \frac{1}{c_e} e^{\epsilon B}$, where t denotes the index of the last iteration of the algorithm. In addition, notice that every time that the algorithm increments y_e , it is by a multiplicative factor of at least $e^{\epsilon B d_{\min}/c_{\max}}$. Consequently, the number of iterations in which y_e is incremented is at most $\frac{c_{\max}}{d_{\min}}$. This implies that the running time of the algorithm is bounded by $m \frac{c_{\max}}{d_{\min}}$, as there are m edges. ■

References

- [1] M. Andrews, J. Chuzhoy, S. Khanna, and L. Zhang. Hardness of the undirected edge-disjoint paths problem with congestion. In *Proceedings 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 226–244, 2005.
- [2] A. Archer, C. H. Papadimitriou, K. Talwar, and É. Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *Proceedings 14th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 205–214, 2003.
- [3] A. Archer and É. Tardos. Frugal path mechanisms. *ACM Transactions on Algorithms*, 3(1), 2007.
- [4] B. Awerbuch, Y. Azar, and A. Meyerson. Reducing truth-telling online mechanisms to online optimization. In *Proceedings 35th Annual ACM Symposium on Theory of Computing*, pages 503–510, 2003.
- [5] Y. Azar and O. Regev. Combinatorial algorithms for the unsplittable flow problem. *Algorithmica*, 44(1):49–66, 2006.
- [6] Y. Bartal, R. Gonen, and N. Nisan. Incentive compatible multi unit combinatorial auctions. In *Proceedings 9th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 72–87, 2003.
- [7] P. Briest, P. Krysta, and B. Vöcking. Approximation techniques for utilitarian mechanism design. In *Proceedings 37th ACM Symposium on Theory of Computing*, pages 39–48, 2005.
- [8] J. Chuzhoy, V. Guruswami, S. Khanna, and K. Talwar. Hardness of routing with congestion in directed graphs. In *Proceedings 39th Annual ACM Symposium on Theory of Computing*, pages 165–178, 2007.
- [9] L. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal on Discrete Mathematics*, 13(4):505–520, 2000.

- [10] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *Proceedings 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 300–309, 1998.
- [11] V. Guruswami, S. Khanna, R. Rajaraman, F. B. Shepherd, and M. Yannakakis. Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. *Journal of Computer and System Sciences*, 67(3):473–496, 2003.
- [12] R. Lavi and C. Swamy. Truthful and near-optimal mechanism design via linear programming. In *Proceedings 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 595–604, 2005.
- [13] D. J. Lehmann, L. O’Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.
- [14] A. Mu’alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *Proceedings 18th National Conference on Artificial Intelligence*, pages 379–384, 2002.
- [15] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.
- [16] P. Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *Journal of Computer and System Sciences*, 37(2):130–143, 1988.
- [17] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.
- [18] A. Srinivasan. Improved approximation guarantees for packing and covering integer programs. *SIAM Journal on Computing*, 29(2):648–670, 1999.