Convex Combinations of Single Source Unsplittable Flows*

Maren Martens[1,**], Fernanda Salazar[2,**], and Martin Skutella[3]

[1] University of British Columbia, Sauder School of Business,
2053 Main Mall, Vancouver, BC V6T1Z2, Canada
maren.martens@sauder.ubc.ca
[2] Escuela Politécnica Nacional, Departamento de Matemática,
Ladrón de Guevara E11-253, Quito, Ecuador
msalazar@math.epn.edu.ec
[3] Universität Dortmund, Fachbereich Mathematik,
44221 Dortmund, Germany
martin.skutella@uni-dortmund.de

Abstract. In the single source unsplittable flow problem, commodities must be routed simultaneously from a common source vertex to certain destination vertices in a given digraph. The demand of each commodity must be routed along a single path. In a groundbreaking paper Dinitz, Garg, and Goemans [4] prove that any given (splittable) flow satisfying certain demands can be turned into an unsplittable flow with the following nice property: In the unsplittable flow, the flow value on any arc exceeds the flow value on that arc in the given flow by no more than the maximum demand.

Goemans conjectures that this result even holds in the more general context with arbitrary costs on the arcs when it is required that the cost of the unsplittable flow must not exceed the cost of the given (splittable) flow. The following is an equivalent formulation of Goemans' conjecture: Any (splittable) flow can be written as a convex combination of unsplittable flows such that the unsplittable flows have the nice property mentioned above. We prove a slightly weaker version of this conjecture where each individual unsplittable flow occurring in the convex combination does not necessarily fulfill the original demands but rounded demands. Preliminary computational results based on our underlying algorithm support the strong version of the conjecture.

1 Introduction

Problem Definition and Notation. The single source unsplittable flow problem was introduced by Kleinberg [8]. We are given a digraph $D = (V, A)$, a source $s \in V$ and K sinks $t_1, \ldots, t_K \in V$. The source and sink nodes are also called *terminals*. We assume without loss of generality that the terminals are pairwise distinct. An unsplittable flow f consists of s-t_i-paths P_i, for $i = 1, \ldots, K$, together with corresponding

* This work was supported in part by the Graduate School of Production Engineering and Logistics, North Rhine-Westphalia, by the DFG Focus Program 1126, Algorithmic Aspects of Large and Complex Networks, grants SK 58/4-1 and SK 58/5-3, and by the German Academic Exchange Service (DAAD).

** Part of this work was done while the authors were at Universität Dortmund.

L. Arge and E. Welzl (Eds.): ESA 2007, LNCS 4698, pp. 395–406, 2007.

flow values $f_i \geq 0$. The flow on arc $a \in A$ is then given as $f(a) = \sum_{i:a \in P_i} f_i$. An unsplittable flow is said to satisfy demands d_i, $i = 1, \ldots, K$, if $f_i = d_i$ for all i.

We also consider flows from s to the sinks $t_1, \ldots, t_K$ that are not necessarily unsplittable. Such a flow f is given by flow values $f(a)$ for all arcs $a \in A$ such that flow conservation constraints for all non-terminal nodes are met. Moreover, the net amount of flow leaving the source s as well as the net amount of flow arriving at each sink t_i must be non-negative. In order to emphasize that a flow is not unsplittable we sometimes also call it a *splittable flow*. A splittable flow satisfies demands d_i, $i = 1, \ldots, K$, if the net amount of flow arriving at t_i equals d_i for all i. The flow traveling from the source to sink t_i is sometimes also referred to as *commodity* i.

Dinitz, Garg, and Goemans [4] present an algorithm that turns a given splittable flow f^{init} satisfying demands d_i, $i = 1, \ldots, K$, into an unsplittable flow f satisfying the same demands such that

$$f(a) \;\leq\; f^{\text{init}}(a) + d_{\max} \qquad \text{for all arcs } a \in A \tag{1}$$

where $d_{\max} := \max_{i=1,\ldots,K} d_i$. Goemans conjectures that this result can be generalized as follows.

Conjecture 1 (Goemans [6]). For any cost function $c : A \to \mathbb{R}$, a splittable flow f^{init} satisfying given demands d_i, $i = 1, \ldots, K$, can be turned into an unsplittable flow f satisfying the same demands such that property (1) holds and the cost of f is bounded by the cost of f^{init}, i.e.,

$$\sum_{a \in A} c(a) f(a) \;\leq\; \sum_{a \in A} c(a) f^{\text{init}}(a).$$

Network flows are usually considered in digraphs with arc capacities $u : A \to \mathbb{R}_0^+$. The capacity $u(a)$ of arc a is an upper bound on the amount of flow that can be sent through arc a. Moreover, for unsplittable flow problems it is often assumed that all demands are at most as large as the minimum arc capacity, i.e., $d_{\max} \leq u_{\min} := \min_{a \in A} u(a)$ such that any commodity can in principle be routed through any arc unsplittably. This condition is also known as the *balance condition*. If the balance condition is fulfilled and f^{init} obeys given arc capacities, then it follows from property (1) that the unsplittable flow f has congestion at most 2, i.e., $f(a) \leq 2u(a)$ for all arcs $a \in A$. In particular, the algorithm presented by Dinitz et al. [4] achieves performance ratio 2 for the objective to minimize congestion.

Related Results from the Literature. The single source unsplittable flow problem is a special case of the more general unsplittable flow problem (UFP) where each commodity has its own source and sink. This problem has been well studied in the literature. In the case that we are given arc capacities and demands for each commodity and look for an unsplittable flow of minimum congestion, i.e., of minimum overload of arc capacities, Raghavan and Thompson [13,12] introduce a randomized rounding technique which yields an $O(\log m / \log \log m)$-approximation algorithm provided that the balance condition[1] holds. Here, m is the number of arcs in the underlying graph. Chuzhoy

[1] Unless stated otherwise, the balance condition is always assumed to be met for the UFP.

and Naor [3] show that the directed case of the UFP is $\Omega(\log\log m)$-hard to approximate unless $NP \subseteq DTIME(n^{O(\log\log\log n)})$, where n is the number of vertices in the underlying graph. Before this result was found, only APX-hardness for the UFP was known (see, e.g., Kleinberg [8]). In the special case of unit demands and unit edge capacities (the *edge-disjoint paths problem*) Andrews and Zhang [1] prove that there is no $(\log\log m)^{1-\epsilon}$-approximation for the undirected congestion minimization problem, unless $NP \subseteq ZPTIME(n^{polylog\ n})$.

For the optimization problem to route a subset of commodities whose total sum of demands is maximal, Azar and Regev [2] present a strongly polynomial algorithm with approximation ratio $O(\sqrt{m})$. Kolman and Scheideler [10] even give a strongly polynomial $O(\sqrt{m})$-approximation algorithm for the problem without the balance condition. On the other hand, Guruswami, Khanna, Rajaraman, Shepherd, and Yannakakis [7] show that there is no approximation algorithm with performance ratio $O(m^{\frac{1}{2}-\epsilon})$ for any $\epsilon > 0$, unless $P = NP$.

It is an easy observation that already the single source unsplittable flow problem without costs contains several well-known NP-complete problems as special cases, such as, for example, PARTITION, BIN PACKING, or even scheduling parallel machines with makespan objective [11]. If we consider the problem with costs, we obtain the KNAPSACK problem as a special case. We refer to [4,8,9,14] for more details and other special cases.

Kleinberg [8], Dinitz, Garg, and Goemans [4], Kolliopoulos and Stein [9], and Skutella [14] present approximation algorithms for various optimization versions of the single source unsplittable flow problem. Du and Kolliopoulos [5] have implemented and empirically tested several of those approximation algorithms. As already mentioned above, a 2-approximation algorithm for congestion minimization is given in [4]. In [14], a 3-approximation algorithm is presented for the corresponding problem with costs. It is also shown there that the performance ratio can be decreased to 2 if the demands are multiples of each other, i.e., $d_i | d_j$ or $d_j | d_i$ for all $i, j = 1, \ldots, K$. In fact, it is shown in [14] that Conjecture 1 holds in this special case. For arbitrary demands, a weaker version of Conjecture 1 is shown where property (1) is replaced with the following less restrictive property: $f(a) \leq 2f^{\text{init}}(a) + d_{\max}$ for all arcs $a \in A$.

Contribution of this Paper. It is not difficult to observe that the following conjecture is equivalent to Conjecture 1.

Conjecture 2. A splittable flow f^{init} satisfying given demands d_i, $i = 1, \ldots, K$, can be written as a convex combination of unsplittable flows satisfying the same demands such that property (1) holds for each unsplittable flow f occurring in the convex combination.

We argue briefly that the two conjectures are indeed equivalent. It is easy to see that Conjecture 1 holds if Conjecture 2 is true. On the other hand, if Conjecture 2 is false, then there exists a splittable flow f^{init} that is not contained in the convex hull of the set of unsplittable flows f satisfying property (1) and the same demands as f^{init}. In this case there must exist a separating hyperplane whose normal vector yields a cost function c such that the cost of f^{init} with respect to c is strictly smaller than the cost of every unsplittable flow under consideration. This contradicts Conjecture 1.

Unfortunately we are not able to prove Goemans' Conjecture in this paper but we show the following slightly weaker version.

Theorem 1. *A splittable flow f^{init} satisfying given demands d_i, $i = 1, \ldots, K$, can be written as a convex combination of unsplittable flows such that property* (1) *holds for each unsplittable flow f occurring in the convex combination.*

The only difference to Conjecture 2 is that we cannot guarantee that the unsplittable flows occurring in the convex combination satisfy the same demands as the given flow f^{init}. Instead the demands satisfied by an individual unsplittable flow are the original demands rounded (up or down) by a factor of at most 2 to the next $d_{max}/2^\ell$ with $\ell \in \mathbb{N}$. Here, d_{max} is the maximum original demand.

In Section 2 we present an algorithm that, given a splittable flow f^{init}, computes unsplittable flows together with appropriate weights such that the resulting convex combination (weighted sum) is equal to the given flow f^{init}. The algorithm uses ideas of Kolliopoulos and Stein [9] and Skutella [14]. It builds a binary tree whose root node is the given splittable flow f^{init} and whose leafs are unsplittable flows fulfilling property (1). Moreover, each non-leaf node is a convex combination of its two children. As a consequence, the splittable flow f^{init} at the root is a convex combination of the unsplittable flows at the leafs.

We give a detailed analysis of the algorithm proving its correctness and the existence of the requested unsplittable flows in Section 3. Finally, in Section 4 we discuss preliminary computational results trying to confirm that there also exist convex combinations of unsplittable flows of additive congestion at most d_{max} if we restrict to using the original demands instead of the rounded ones that are produced by our algorithm. We also observe in this section that a convex combination as described in Theorem 1 can be computed in polynomial time.

2 Constructing the Convex Combination

We present an algorithm that, given a splittable flow f^{init}, computes unsplittable flows with weights such that the weighted sum (convex combination) of the unsplittable flows equals f^{init}. These unsplittable flows have property (1) and they satisfy demands that are obtained by rounding the ones in f^{init}.

2.1 Preliminaries

To simplify the description of the algorithm we introduce some more vocabulary. We say that a *terminal path* is a maximal (not necessarily directed) simple path in D with endpoints in $\{s, t_1, \ldots, t_K\}$, where *maximal* means that it is not extendable at either of its endpoints. A (not necessarily directed) cycle or a terminal path is called an *augmenting structure*.

We use $d_{max} := \max_{i=1,\ldots,K} d_i$ ($d_{min} := \min_{i=1,\ldots,K} d_i$) to denote the maximum (minimum) demand satisfied by f^{init}. For a natural number ℓ, we define $r_f^\ell(a) := f(a) \mod \frac{d_{max}}{2^\ell}$, for $a \in A$. A flow is called *q-integral* for some $q \in \mathbb{R}^+$, if the flow value on each arc is an integral multiple of q. It is well known that any single source multicommodity flow that is q-integral can be decomposed into flows on paths with flow value q.

The definition $f(a) := \sum_{i:a\in P_i} f_i$, for all $a \in A$, enables us to build the sum $f+g$ of two flows f and g arcwise, even if one or both flows are unsplittable and given pathwise.

2.2 The Algorithm

In the following we shortly sketch the idea of the algorithm first and give a more detailed characterization of it later on: We want to round each demand to the two nearest (upper and lower) values of the form $d_{max}/2^{\ell}$, for some $\ell \in \mathbb{N}$, and then send these rounded demands unsplittably. A convex combination of the final unsplittable flows (with rounded demands) will yield our original flow. The largest value we need to consider for ℓ is

$$L := \left\lceil \log \frac{d_{max}}{d_{min}} \right\rceil,$$

since $d_{max}/2^L$ is the largest fraction $d_{max}/2^{\ell}$ ($\ell \in \mathbb{N}$) that is at most d_{min}. (Thus, $d_{max}/2^L$ is the smallest demand obtained from rounding all demands to $(d_{max}/2^{\ell})$-integrality, for some $\ell \in \mathbb{N}$).

We start by making the input flow $(d_{max}/2^L)$-integral. This is done by augmenting flow on cycles and terminal paths. The particular augmenting structure and the increment of flow are chosen such that after augmentation the flow on at least one additional arc is $(d_{max}/2^L)$-integral. Augmentation on a cycle does not change the demands, whereas augmentations on terminal paths yield changes of demands to $(d_{max}/2^L)$-integrality. We always augment in both directions of an augmenting structure and thus obtain two new flows in each step such that the "parent" flow is a convex combination of the two. Depending on the direction in which flow is augmented on a path, demands are rounded up or down. (See Figure 1 for an illustration of the algorithm.) Considering the two direct descendants of a "parent" flow, which result from a change of flow in either direction of an augmenting structure, we simply construct a binary tree of flows in which the descendants of a flow f can be used to form a convex combination of f. The leaves of this tree finally form the desired convex combination of unsplittable flows for the root flow f^{init}.

When a flow is $(d_{max}/2^L)$-integral, demands of value $d_{max}/2^L$ can be satisfied unsplittably. For this reason, we want to prevent such demands and corresponding flow carrying paths from further changes. Thus, for each sink t_i with demand $d_{max}/2^L$, we decrease the current flow along a corresponding flow carrying s-t_i-path by $d_{max}/2^L$. These paths with the flow values are stored for all subsequent flows. After the decrement, all demands are at least $d_{max}/2^{L-1}$ and we turn to make the (remaining) flow $(d_{max}/2^{L-1})$-integral. We proceed in this manner until the (remaining) flow is d_{max}-integral. Then all (remaining) demands equal d_{max} and are served unsplittably.

For the sake of simple presentation and analysis of the algorithm, we give a recursive description of it in Algorithm 1. The initial call to start the recursive algorithm is given by DECOMP(D,f^{init},$\emptyset$,1) where f^{init} is the single source multicommodity flow in the digraph D that we want to write as a convex combination of unsplittable flows. The third parameter is a set of paths with corresponding flow values. If (P, ϕ) is in this set, it means that all subsequent flows route ϕ units of flow along path P. The last parameter indicates the weight of the flow that is to be decomposed. For the initial flow this weight equals 1.

A call of DECOMP(D,f^{init},$\emptyset$,1) effects the following: In each step of the recursion, it first updates the input flow f by iteratively deleting demands $d_{max}/2^{\ell}$, for $\ell \in \mathbb{N}$, if f is $(d_{max}/2^{\ell})$-integral. The related flow carrying paths are added to the current

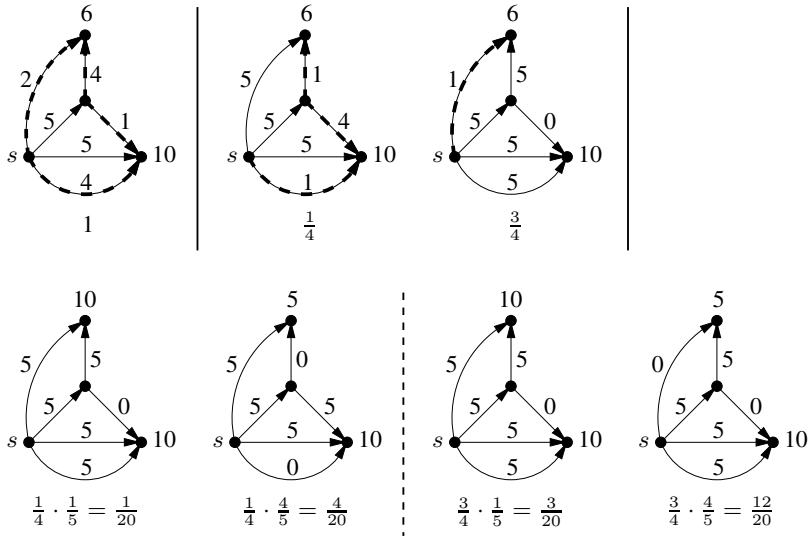

Fig. 1. Steps of the algorithm to 5-integrality. We start with the flow in the upper left corner and read from left to right first. Since $d_{max}/2^L$ equals 5, we need to make the flow 5-integral. The dashed arcs indicate the augmenting structure that is used. Since we augment flow in either direction of an augmenting structure, each non-5-integral flow produces two new flows that are separated from their "parents" by a vertical line. We consider the first flow. If we use the indicated augmenting structure clockwise, we may augment by 3. Then the arc with flow value 2 becomes 5-integral. Using the augmenting structure counterclockwise, we may augment by 1. Then the unit of flow on the arc with flow value 1 recedes. The number below each flow indicates its weight in the convex combination.

unsplittable flow given by $\mathcal{P}$. Afterwards it decomposes the (remaining) flow f into a convex combination of two new flows f_1 and f_2 that result from a single augmentation in both directions of a suitable augmenting structure. Further steps of the recursion decompose f_1 and f_2 into convex combinations of unsplittable flows.

The augmentation itself works as follows. Let us assume that the flow f that remains after the update is $(d_{max}/2^\ell)$- but not $(d_{max}/2^{\ell-1})$-integral, for some $\ell \in \{1, \ldots, L\}$. Then we consider the subgraph $\tilde{D}$ of D that consists of all arcs $a \in A$ whose flow values $f(a)$ are not $(d_{max}/2^{\ell-1})$-integral. Starting from an arbitrary arc we follow an undirected path (in either direction) until there is no more incident arc or we get to a node which has already been visited. The first criterion results in a terminal path, the second one in a cycle. (We prove this later on in Lemma 2.) For the resulting augmenting structure C we augment by the minimum δ of gaps between flow values and the next lower multiples of $d_{max}/2^{\ell-1}$ for arcs that are used by C in backward direction and of gaps between flow values and the next upper multiples of $d_{max}/2^{\ell-1}$ for arcs that are used by C in forward direction. Defining C^- as the backward arcs in C and C^+ as the forward arcs in C we can write δ as follows.

$$\delta = \min\{\min_{a \in C^-} r_f^{\ell-1}(a), \min_{a \in C^+} \frac{d_{max}}{2^{\ell-1}} - r_f^{\ell-1}(a)\}$$

Algorithm 1. DECOMP(D,f,$\mathcal{P}$,w)

Input: A digraph $D = (V, A)$, a single source multicommodity flow f in D with source s, a set $\mathcal{P}$ of paths from s to pairwise distinct nodes t in D with corresponding flow values, and a positive weight $w \in (0, 1]$. The maximum (minimum) demand, that f satisfies, is d_{max} (d_{min}).

Output: A set of unsplittable flows with weights that sum up to w yielding a conic combination of $w(f + f_{\mathcal{P}})$, where $f_{\mathcal{P}}$ denotes the flow that is given by $\mathcal{P}$.

for $i = \left\lceil \log \frac{d_{max}}{d_{min}} \right\rceil$ **downto** 0 **do**
  if f *is* $(d_{max}/2^i)$*-integral* **then**
    for *each sink* t *having demand* $d = d_{max}/2^i$ *in* f **do**
      Determine an arbitrary flow carrying s-t-path P in D.
      Set $f(a) := f(a) - d$ for all $a \in P$.
      Set $\mathcal{P}' := \mathcal{P} \cup \{(P, d)\}$.
    end
  end
end
if $f \equiv 0$ **then**
  return $(\mathcal{P}', w)$.
end
Set $\ell := \min\{\min\{j \in \mathbb{N} | f \text{ is } (d_{max}/2^j)\text{-integral}\}, \left\lceil \log \frac{d_{max}}{d_{min}} \right\rceil + 1\}$.
Let $C \subseteq A$ be an augmenting structure with $r_f^{\ell-1}(a) \neq 0 \; \forall \, a \in C$.
Set $C^+ := \{a \in C \mid C \text{ traverses } a \text{ in forward direction}\}$,
  $C^- := \{a \in C \mid C \text{ traverses } a \text{ in backward direction}\}$.
Set $\delta_1 := \min\{\min_{a \in C^-} r_f^{\ell-1}(a), \min_{a \in C^+} \frac{d_{max}}{2^{\ell-1}} - r_f^{\ell-1}(a)\}$,
  $\delta_2 := \min\{\min_{a \in C^+} r_f^{\ell-1}(a), \min_{a \in C^-} \frac{d_{max}}{2^{\ell-1}} - r_f^{\ell-1}(a)\}$.
For $a \in A \setminus C$ set $f_1(a) := f(a)$ and $f_2(a) := f(a)$.
For $a \in C^+$ set $f_1(a) := f(a) + \delta_1$ and $f_2(a) := f(a) - \delta_2$.
For $a \in C^-$ set $f_1(a) := f(a) - \delta_1$ and $f_2(a) := f(a) + \delta_2$.
Set $w_1 := \frac{\delta_2}{\delta_1 + \delta_2} w$ and $w_2 := \frac{\delta_1}{\delta_1 + \delta_2} w$.
return DECOMP(D,f_1,$\mathcal{P}'$,w_1) $\cup$ DECOMP(D,f_2,$\mathcal{P}'$,w_2).

Now let f_1 and f_2 be the flows resulting from augmenting along C and its "counterpart", i.e., C in the opposite direction. Let δ_1 and δ_2 be the corresponding augmentation values. Then the weight of f_i (for $i = 1, 2$) is given by the weight of f multiplied with $\delta_{3-i}/(\delta_1 + \delta_2)$.

3 Analysis of the Algorithm

In Section 3.1 we show that the flows and weights returned by DECOMP(D,f^{init},$\emptyset$,1) yield a convex combination for f^{init}. Further, the produced flows are unsplittable and all its demands are of the form $d_{max}/2^\ell$, for some $\ell \in \{0, \ldots, \lceil \log(d_{max}/d_{min}) \rceil\}$. To prove Theorem 1 we have to show as well that, on each arc $a \in A$, the final flows send at most the initial flow $f^{\text{init}}(a)$ plus an additive d_{max}. This is done in Section 3.2.

3.1 Correctness of the Algorithm

It is easy to see that the following lemma is true. A detailed proof is omitted due to space limitations.

Lemma 1. *For each DECOMP(D,f,$\mathcal{P}$,w) that is triggered by DECOMP(D,f^{init},$\emptyset$,1) it holds that*

1. *f is a single source multicommodity flow in D,*
2. *$\mathcal{P}$ is a set of paths from the source of f to pairwise distinct nodes t in D with corresponding flow values, and*
3. *$w \in (0, 1]$.*

Lemma 1 shows that the algorithm is well-defined. The following lemma is necessary to prove that our algorithm terminates. It follows immediately from flow conservation.

Lemma 2. *If a flow f in D is not $(d_{max}/2^{\ell})$-integral, for some $\ell \in \mathbb{N}$, then there exists an augmenting structure $C \subseteq A$ with $r_f^{\ell}(a) \neq 0$, for all $a \in C$.*

The definition of δ_1 and δ_2 and the augmentation rule imply that if f is not decreased in the for-loop of DECOMP(D,f,$\mathcal{P}$,w), the flows f_1 and f_2 are "more integral" than f.

Lemma 3. *For any flow f that is augmented in Algorithm 1 and its corresponding value ℓ as defined in the algorithm, it holds that f_1 and f_2 each have at least one more arc than f whose flow value is $(d_{max}/2^{\ell-1})$-integral.*

Before we turn to proving that the flows/weights returned by DECOMP(D,f,$\mathcal{P}$,w) yield a conic combination of wf, we show that the procedure indeed terminates and outputs unsplittable flows whose demands are of the form $d_{max}/2^{\ell}$, for some $\ell \in \{0, \ldots, \lceil \log(d_{max}/d_{min}) \rceil\}$.

Corollary 1. *DECOMP(D,f,$\mathcal{P}$,w) terminates. The output is a set of unsplittable flows whose demands are of the form $d_{max}/2^{\ell}$, for some $\ell \in \{0, \ldots, \lceil \log(d_{max}/d_{min}) \rceil\}$.*

Proof. We proved that the flows f_1 and f_2 resulting from DECOMP(D,f,$\mathcal{P}$,w) have fewer positive demands than f or are "more integral". The first property eventually results in a decrement of the input flow to the zero flow. In every recursive call of Algorithm 1 in which the number of positive demands is not decreased for the respective input, the flow value on at least one of its arcs changes to a "higher" integrality. After at most $|A|$ steps we therefore change from $(d_{max}/2^{\ell})$-integrality to $(d_{max}/2^{\ell-1})$-integrality for some $\ell \in \{1, \ldots, L\}$ (or respectively from the initial state to $(d_{max}/2^{L})$-integrality). At this point demands of value $(d_{max}/2^{\ell-1})$ and corresponding flow are deleted in the for-loop. If no such demands exist, we go to "higher" integralities and delete demands at the latest when the flow is d_{max}-integral. Therefore, at some point all demands are deleted and the algorithm terminates.

It follows from the preceding analysis that all paths in the final $\mathcal{P}'$ connect the source in f with pairwise distinct sinks. Thus, $\mathcal{P}'$ yields an unsplittable flow. It follows directly from the specification of the algorithm that all demands served by $\mathcal{P}'$ are of the form $d_{max}/2^{\ell}$, for some $\ell \in \{0, \ldots, \lceil \log(d_{max}/d_{min}) \rceil\}$.

In the following we use $f_{\mathcal{P}}$ to denote the flow that is given by some set $\mathcal{P}$ of paths with corresponding flow values. We prove the following helpful lemma in order to show that DECOMP(D,f,$\mathcal{P}$,w) returns the specified output.

Lemma 4. *Consider DECOMP(D,f,$\mathcal{P}$,w). It holds that*

$$w(f + f_{\mathcal{P}}) = w_1(f_1 + f_{\mathcal{P}'}) + w_2(f_2 + f_{\mathcal{P}'}) \tag{2}$$

and $w_1 + w_2 = w$.

Proof. The second part of the lemma follows immediately from the definition of w_1 and w_2. Equation (2) can be proven as follows. For all arcs $a \in A$ that are not in the augmenting structure C that leads from f to f_1 and f_2, it holds that $f_1(a) = f_2(a) = f(a) - (f_{\mathcal{P}'}(a) - f_{\mathcal{P}}(a))$. Since $w_1 + w_2 = w$, equation (2) follows immediately.

Now consider an arc $a \in C^+$. It holds that $f_1(a) = f(a) - (f_{\mathcal{P}'}(a) - f_{\mathcal{P}}(a)) + \delta_1$ and $f_2(a) = f(a) - (f_{\mathcal{P}'}(a) - f_{\mathcal{P}}(a)) - \delta_2$. Using $w_1 + w_2 = w$, it follows that $w_1 f_1(a) + w_2 f_2(a) = wf(a) + wf_{\mathcal{P}}(a) - w_1 f_{\mathcal{P}'}(a) - w_2 f_{\mathcal{P}'}(a)$. The proof is analogous for $a \in C^-$.

The following corollary demonstrates that the output of DECOMP(D,f,$\mathcal{P}$,w) is correct. With this result we are finished proving the correctness of the algorithm as described in Section 2. To prove our main result we still need to show that all unsplittable flows that are returned by DECOMP(D,f^{init},$\emptyset$,1) have congestion at most 2. This is done in Theorem 2 in Section 3.2.

Corollary 2. *The flows and weights returned by DECOMP(D,f,$\mathcal{P}$,w) yield a conic combination of $w(f + f_{\mathcal{P}})$ whose weights sum up to w.*

The proof of Corollary 2 uses induction on the depth of recursion and Lemma 4. It is omitted due to space limitations. The next corollary follows immediately.

Corollary 3. *The flows and weights returned by DECOMP(D,f^{init},$\emptyset$,1) yield a convex combination of f.*

3.2 Upper Bound on the Congestion

Together with the algorithm from Section 2 and its analysis in Section 3.1 the following theorem is the last component to prove Theorem 1.

Theorem 2. *For a single source multicommodity flow f^{init} in $D = (V, A)$ and any arc $a \in A$, it holds that the flow along a in any flow produced by DECOMP(D,f^{init},$\emptyset$,1) exceeds $f^{init}(a)$ by at most an additive d_{max}.*

Proof. Consider the progression of the input flow while DECOMP(D,f^{init},$\emptyset$,1) is running. Let f^0 be a flow that occurs on the way to $(d_{max}/2^L)$-integrality of the input flow. Further, let $\mathcal{P}^0$ be the current unsplittable flow while f^0 is considered in the algorithm.

By the choice of δ_1 and δ_2, it holds for all $a \in A$ that

$$f^0(a) + f_{\mathcal{P}^0}(a) \leq f^{\text{init}}(a) + \frac{d_{max}}{2^L} - \left(f^{\text{init}}(a) \mod \frac{d_{max}}{2^L} \right), \tag{3}$$

because once the flow on a is $(d_{max}/2^L)$-integral, i.e., rounded to at most the next multiple of $d_{max}/2^L$, it is not changed again on the way to $(d_{max}/2^L)$-integrality.

After $(d_{max}/2^L)$-integrality was reached, the input flow is iteratively augmented to $(d_{max}/2^{L-\ell})$-integrality for gradually increasing $\ell \in \{1, \ldots, L\}$. Let f^ℓ be a flow that occurs while DECOMP(D,f^{init},$\emptyset$,1) is running and that is $(d_{max}/2^{L-\ell})$-integral, but not $(d_{max}/2^{L-\ell-1})$-integral. Further, let $f^{\ell-1}$ be any ancestor of f^ℓ, i.e., any of the flows that (indirectly) caused the creation of f^ℓ, that is $(d_{max}/2^{L-\ell+1})$-integral. Again let $\mathcal{P}^\ell$ and $\mathcal{P}^{\ell-1}$ be the corresponding unsplittable flows.

In analogy with (3), it follows from the choice of δ_1 and δ_2 that for all $a \in A$

$$f^\ell(a) + f_{\mathcal{P}^\ell}(a) \leq f^{\ell-1}(a) + f_{\mathcal{P}^{\ell-1}}(a) + \frac{d_{max}}{2^{L-\ell}} - \frac{d_{max}}{2^{L-\ell+1}}.$$

We can prove an analogous equation if some integrality step is omitted, i.e., if there is no ancestor of f^ℓ that is $(d_{max}/2^{L-\ell+1})$-integral. Let ℓ' be the largest integer that is smaller than ℓ and for which an ancestor of f^ℓ exists that is $(d_{max}/2^{L-\ell'})$-integral. Then it holds that $f^\ell(a) + f_{\mathcal{P}^\ell}(a) \leq f^{\ell'}(a) + f_{\mathcal{P}^{\ell'}}(a) + d_{max}/2^{L-\ell} - d_{max}/2^{L-\ell'}$.

We obtain iteratively, for all $\ell \in \{0, \ldots, L\}$, that

$$f^\ell(a) + f_{\mathcal{P}^\ell}(a) \leq f^{\text{init}}(a) + \frac{d_{max}}{2^{L-\ell}} - \left(f^{\text{init}}(a) \mod \frac{d_{max}}{2^L} \right). \tag{4}$$

Now consider the point when the flow f is changed to $f' \equiv 0$ in the for-loop. Let $\mathcal{P}$ and $\mathcal{P}'$ be the corresponding unsplittable flows. Then $\mathcal{P}'$ is one of the output flows of the algorithm. Since d_{max} is the maximum demand in f, it follows that f is d_{max}-integral. With (4) we have $f_{\mathcal{P}'}(a) = f(a) + f_{\mathcal{P}}(a) \leq f^{\text{init}}(a) + d_{max} - (f^{\text{init}}(a) \mod \frac{d_{max}}{2^L}) \leq f^{\text{init}}(a) + d_{max}$.

Note that it even holds, for all $a \in A$, that $f_{\mathcal{P}'}(a) < f^{\text{init}}(a) + d_{max}$. To obtain this result, we have to regard that $f^0(a) + f_{\mathcal{P}^0}(a) \leq f^{\text{init}}(a)$, if $f^{\text{init}}(a)$ is $(d_{max}/2^L)$-integral.

If we assume f^{init} to be feasible, the next result follows immediately.

Corollary 4. *If a single source multicommodity flow f^{init} in $D = (V, A)$ obeys arc capacities $u : A \to \mathbb{R}^+$ and the balance condition is met, then all flows produced by DECOMP(D,f^{init},$\emptyset$,1) have congestion at most 2.*

We close this section with an example showing that our result is tight (see also [4] for similar results).

Lemma 5. *There exists a network and a feasible fractional single source multicommodity flow f^{init} such that in each convex combination of unsplittable flows forming f^{init} there is at least one flow with congestion arbitrarily close to 2.*

Proof. Consider a network with source s, sinks t_1, t_2 with demands 1 for both commodities, and one additional node v. The arcs in the network with their initial flow values are (s, v) with $f^{\text{init}}((s, v)) = 1 + \epsilon$, (s, t_2) with $f^{\text{init}}((s, t_2)) = 1 - \epsilon$, (v, t_1) with $f^{\text{init}}((v, t_1)) = 1$, and (v, t_2) with $f^{\text{init}}((v, t_2)) = \epsilon$. The capacity of arc a is the maximum of $f^{\text{init}}(a)$ and 1. ϵ is an arbitrary positive number smaller than 1.

Consider arc (s, v). Obviously we have to route commodity 1 on it in each unsplittable flow that participates in a convex combination forming f^{init}. But there must also be at least one unsplittable flow that routes commodity 2 on this arc. Thus, we obtain a flow value of 2 on it and a congestion of $2/(1+\epsilon)$.

4 Some Preliminary Computational Results

We have shown that it is possible to write any (splittable) single source multicommodity flow f^{init} as a convex combination of unsplittable flows obeying condition (1). The demands satisfied by the unsplittable flows that we construct for this convex combination slightly differ from the ones in the original flow.

Using our algorithm, we want to empirically confirm Conjecture 2. In principle, we would like to do the following: Consider all unsplittable flows computed by the algorithm; turn them into unsplittable flows satisfying the original demands d_i, $i = 1, \ldots, K$, by simply routing exactly d_i units of flow along the chosen s-t_i-paths (instead of the rounded demand values); omit all unsplittable flows which, after this modification, no longer obey condition (1) (notice that this can easily happen already for simple examples); check whether f^{init} is contained in the convex hull of the remaining unsplittable flows.

The main problem with this approach is the huge size of the binary tree computed by our algorithm. Already for relatively small instances the algorithm does not terminate in reasonable time due to the exponential growth of the computed binary tree. It is therefore not realistic to try to compute all unsplittable flows corresponding to leaf nodes of that tree. Instead, we have to thin out the tree and only compute a subset of leafs of reasonable size. Of course, this subset should still have the property that f^{init} is contained in the convex hull of unsplittable flows given by the subset.

More precisely, we proceed as follows. We start to compute the binary tree in breadth-first manner. When we arrive at a layer of the tree containing "too many" nodes, we omit some of them and only maintain a subset of "reasonable size". By construction the flow f^{init} at the root node is a convex combination of the flows corresponding to the nodes of the tree in any fixed layer. It follows from Carathéodory's theorem that f^{init} can be written as a convex combination of a subset containing at most $|A| + 1$ flows. It is therefore possible to keep the width of the tree bounded by $O(|A|)$ and still maintain the property that f^{init} is a convex combination of the flows in any fixed layer. In our implementation, we simply use CPLEX to find suitable subsets of flows when we arrive at a layer of the tree that contains "too many" nodes. Since the depth of the tree is polynomially bounded, we can even find a representation of f^{init} as a convex combination of unsplittable flows in polynomial time and thus strengthen Theorem 1 as follows.

Theorem 3. *A convex combination as described in Theorem 1 can be obtained in polynomial time.*

For the purpose of our empirical study it is not advisable to reduce the width of each layer of the tree as far as possible (i.e., to width $|A| + 1$). This decreases our chance to find sufficiently many "good" unsplittable flows in the end that contain f^{init} in their convex hull. Therefore the challenge is to decide for each layer of the tree which flows to keep and which to omit in order to keep the width of the tree small. We have ex-

perimented with several different strategies but have not found the ideal strategy yet. However, for most instances that we consider our choice of flows is suitable in the sense that the resulting unsplittable flows meet the congestion requirement.

For the empirical tests we use 14 test instances that were also considered by Du and Kolliopoulos [5] for their empirical evaluation of approximation algorithms. The instances come from the following generators: noigen, satgen, rangen, and genrmf. A complete description of every generator can be found in [5]. We have tested different ways to choose the flows that we keep in our convex combinations. So far, for half of the 14 instances we were able to compute an appropriate set of unsplittable flows of additive congestion at most d_{max}. For another three instances the multiplicative congestion of the unsplittable flows does not exceed 2, while for the remaining four instances we have not found an appropriate set of unsplittable flows yet. We are currently still working on finding better heuristics to choose the flows to be kept in each layer of the tree.

References

1. Andrews, M., Zhang, L.: Hardness of the undirected congestion minimization problem. In: Proceedings of 37th Annual ACM Symposium on Theory of Computing, pp. 284–293. ACM Press, New York (2005)
2. Azar, Y., Regev, O.: Strongly polynomial algorithms for the unsplittable flow problem. In: Proceedings of the 8th Conference on Integer Programming and Combinatorial Optimization, pp. 15–29 (2001)
3. Chuzhoy, J., Naor, J.: New hardness results for congestion minimization and machine scheduling. In: Proceedings of the 36th Annual ACM Symposium on Theory of Computing, pp. 28–34. ACM Press, New York (2004)
4. Dinitz, Y., Garg, N., Goemans, M.X.: On the single source unsplittable flow problem. Combinatorica 19, 17–41 (1999)
5. Du, J., Kolliopoulos, S.: Implementing approximation algorithms for the single-source unsplittable flow problem. Journal of Experimental Algorithmics 10, 2–3 (2005)
6. Goemans, M.X.: Cited as personal communication from (January 2000) [14, Section 7]
7. Guruswami, V., Khanna, S., Rajaraman, R., Shepherd, B., Yannakakis, M.: Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. In: Proceedings of the 31st Annual ACM Symposium on Theory of Computing, pp. 19–28. ACM Press, New York (1999)
8. Kleinberg, J.M.: Approximation Algorithms for Disjoint Path Problems. PhD thesis, Massachusetts Institute of Technology (May 1996)
9. Kolliopoulos, S.G., Stein, C.: Approximation algorithms for single-source unsplittable flow. SIAM Journal on Computing 31, 919–946 (2002)
10. Kolman, P., Scheideler, C.: Improved bounds for the unsplittable flow problem. In: Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 184–193. ACM Press, New York (2002)
11. Lenstra, J.K., Shmoys, D.B., Tardos, É.: Approximation algorithms for scheduling unrelated parallel machines. Mathematical Programming 46, 259–271 (1990)
12. Raghavan, P.: Probabilistic construction of deterministic algorithms: approximating packing integer programs. Journal of Computer and System Sciences 37, 130–143 (1988)
13. Raghavan, P., Thompson, C.D.: Randomized rounding: A technique for provably good algorithms and algorithmic proofs. Combinatorica 7, 365–374 (1987)
14. Skutella, M.: Approximating the single source unsplittable min-cost flow problem. Mathematical Programming 91, 493–514 (2002)