COMBINATORICA
Bolyai Society – Springer-Verlag

COMBINATORICA **19** (1) (1999) 17–41

ON THE SINGLE-SOURCE UNSPLITTABLE FLOW PROBLEM

YEFIM DINITZ[*], NAVEEN GARG[†] and MICHEL X. GOEMANS[‡]

Received July 12, 1998

Let $G=(V,E)$ be a capacitated directed graph with a source s and k terminals t_i with demands d_i, $1\le i\le k$. We would like to concurrently route every demand on a single path from s to the corresponding terminal without violating the capacities. There are several interesting and important variations of this unsplittable flow problem.

If the necessary cut condition is satisfied, we show how to compute an unsplittable flow satisfying the demands such that the total flow through any edge exceeds its capacity by at most the maximum demand. For graphs in which all capacities are at least the maximum demand, we therefore obtain an unsplittable flow with congestion at most 2, and this result is best possible. Furthermore, we show that all demands can be routed unsplittably in 5 rounds, i.e., all demands can be collectively satisfied by the union of 5 unsplittable flows. Finally, we show that 22.6% of the total demand can be satisfied unsplittably.

These results are extended to the case when the cut condition is not necessarily satisfied. We derive a 2-approximation algorithm for congestion, a 5-approximation algorithm for the number of rounds and a $4.43=1/0.226$-approximation algorithm for the maximum routable demand.

1. Introduction

Let $G=(V,E)$ be a directed graph with edge capacities $c: E\to\mathbb{R}^+$, a source s and k commodities with terminals t_i and demands $d_i\in\mathbb{R}^+$, $1\le i\le k$. A vertex may contain any number of terminals. For each i, we would like to route d_i units of commodity i along a single path from s to t_i so that the total flow through an edge e is at most its capacity $c(e)$. A routing in which each commodity flows along a single path is referred to as an *unsplittable flow*. The setting that we consider is a special case of the general unsplittable multicommodity flow where demands are requested between terminal pairs (s_i,t_i), $1\le i\le k$. In this paper, though, we consider only the single-source version.

Mathematics Subject Classification (1991): 90B10, 90C35, 05C38, 05C85.

[*] This work was done when the author was on leave at the Max-Planck Institut für Informatik, Saarbrücken, Germany from Department of Computer Science, Technion, Haifa, Israel.

[†] This research was done while the author was at the Max-Planck-Institut für Informatik, Saarbrücken and was supported by the EU ESPRIT LTR Project No. 20244 (ALCOM-IT).

[‡] On leave from MIT. The research of this author was supported in part by NSF contract 9623859-CCR.

If we had allowed the demand d_i for commodity i to be split along more than one path then this is a simple network flow problem. In particular, there exists a *fractional* solution, or flow, respecting the capacities if and only if the *cut condition* is met:

> For any set S with $s \notin S$, the total demand of terminals within S is at most the total capacity of the edges entering S.

There are several interesting questions that one may ask with regard to unsplittable flows.

Feasibility. Can one route all commodities unsplittably?

Congestion. What is the smallest $\alpha \geq 1$ such that if we multiply all capacities by α, an unsplittable flow satisfying all demands exists?

Maximization. Find a subset T of commodities which can be routed unsplittably and which maximizes $\sum_{i \in T} d_i$.

Number of rounds. How many rounds are necessary to route all commodities unsplittably?

All these versions were first considered by Kleinberg [2, 3]. As Kleinberg observes, this problem has interesting special cases. The well-known NP-hard PARTITION problem can easily be reformulated as the feasibility or the congestion versions on a graph with two parallel edges, or as the maximization or the number of rounds versions on a graph with a single edge[1]. 3-PARTITION and BIN PACKING can also be easily cast as unsplittable flow problems. This combination of routing and bin packing makes the various versions of the unsplittable flow problem particularly difficult.

Another interesting special case of the unsplittable flow problem is the following scheduling problem. We have to schedule jobs on one of m parallel machines, but job j can only be scheduled on a subset of the machines, say $M(j)$, and this subset depends on the job. The processing time of job j is p_j on any of the admissible machines, and the goal is to minimize the makespan of the schedule. This is a special case of the problem of minimizing makespan on unrelated parallel machines. To formulate this problem as an unsplittable flow problem (see Figure 1), create a vertex for each machine and for each job, and a source s, and connect the source to all machines with edges of capacity T and also connect each machine to the jobs it can process with edges of infinite capacity. Let each vertex representing a job be a terminal of a commodity with a demand equal to the processing time of the job. It is easy to see that this unsplittable flow problem is feasible iff there exists a feasible schedule of makespan at most T. Among other results, Lenstra et al. [5] and Shmoys and Tardos [6] show that, if there exists a fractional solution to this unsplittable flow problem then there exists a feasible schedule of makespan $T + \max_j p_j$. The two constructions are different and the first one assumes that the fractional solution is an extreme point of the corresponding polyhedron.

[1] This actually shows also the NP-hardness of the maximization and the number of rounds versions for fractional flows as well.

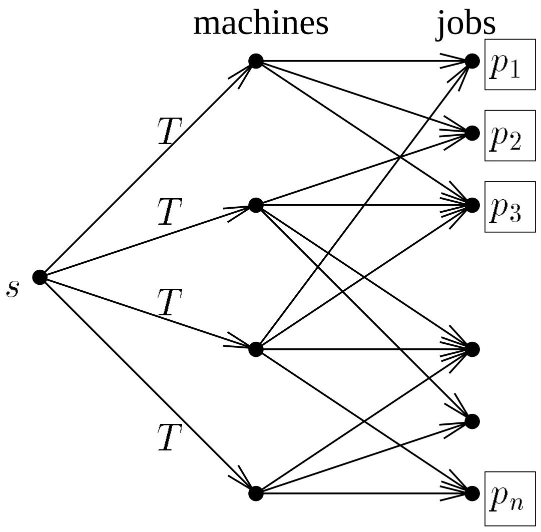

Fig. 1. A parallel machine scheduling problem as an unsplittable flow problem.

For the general single-source unsplittable flow problem, our key result is to transform any flow into an unsplittable flow satisfying the same demands while increasing the capacity of any edge by less than the maximum demand. This is therefore a generalization of the result of Lenstra et al. [5] and Shmoys and Tardos [6], but our transformation does not reduce to any of theirs in the special case of the scheduling problem. In most of the paper, as in the previous papers [2, 3, 4] on the unsplittable flow problem, we will assume that all capacities are at least the maximum demand, i.e., any commodity can be routed on any edge. Thus if the cut condition is satisfied we can find an unsplittable flow of congestion less than 2. This improves the bound of 16 of Kleinberg [2, 3] and the 3.23-approximation algorithm of Kolliopoulos and Stein [4].

Our key result has interesting implications for the other versions of the unsplittable flow problem. If the cut condition is met we show that all commodities can be routed in 5 rounds, i.e., by the union of 5 feasible unsplittable flows. The first constant bound was provided by Kleinberg [2, 3] (though the constant was not given explicitly), and the previous best result was a 13-approximation algorithm for the minimum number of rounds [4]. For the maximization problem, when the cut condition holds, we can always find a feasible unsplittable flow of value at least $0.226\ldots = 1/4.42\ldots$ times the total demand. Again the first constant bound was given in [2, 3], and Kolliopoulos and Stein show how to route a fraction 0.075 of the maximum routable demand [4].

Our results are close to best possible. We show an instance which is fractionally feasible but where any unsplittable flow violates the capacity of some edge by an amount that is arbitrarily close to the maximum demand. This same instance also shows that in going from a fractional flow to an unsplittable flow the factor 2 increase in congestion is unavoidable. For the problem of minimizing the number of rounds we show an instance which is fractionally feasible and where routing all the demands unsplittably requires 3 rounds. Finally, for the version in which we maximize the total demand routed we show an instance where all demands can be routed fractionally but only 38.45% of the total demand can be routed unsplittably.

When the cut condition is not satisfied, our guarantees remain the same: we have a 2-approximation algorithm for congestion, a 5-approximation algorithm for the number of rounds and a 4.43-approximation algorithm for the maximization version, all assuming that the minimum capacity is at least the maximum demand. In the process of converting the result for the maximization version when the cut condition holds into an approximation algorithm, we prove the following interesting variant of our key theorem. If there exists a flow satisfying a fraction α_i of the demand d_i for every i then there exists an unsplittable flow of total value at least $\sum_i \alpha_i d_i$ in which no capacity is violated by more than the maximum demand.

Finally, in the last section, we consider two special cases and derive improved results. The first setting we consider is when all capacities are integral and demands less than one. The other setting is when capacities are integral and all demands are powers of two less or equal to 1. This situation is quite interesting since the bin packing aspect of the problem is then almost annihilated.

2. Preliminaries

Let $c_{\min}$ represent the minimum capacity, $d_{\min}$ the minimum non-zero demand, and $d_{\max}$ the maximum demand.

To avoid any confusion, we first define the basic terminology we use. By a (fractional) *flow* in the graph G with demands d_i between a single source s and terminals t_i, we mean an assignment $f : E \to \mathbb{R}^+ \cup \{0\}$ such that the net inflow at any vertex $v \neq s$, $\sum_{e=(u,v)\in E} f(e) - \sum_{e=(v,u)\in E} f(e)$, is nonnegative and at most the sum of the demands at v. Such a flow can be easily turned to a multicommodity flow from the source to the terminals, by flow decomposition. We say that a flow f is *feasible* or *respects the capacities* if for all edges e, $f(e) \leq c(e)$. The *value* of a flow is equal to the net outflow (defined as minus the inflow) at s. A flow f *satisfies all demands* if, for every $v \in V$, the net inflow at v equals the sum of the demands at v. We will also refer to flows satisfying a subset of demands; in this case, the net inflow at v must be equal to the sum over the corresponding subset of the demands at v. We say that a flow f satisfying a (sub)set of demands is *unsplittable* if the flow for each routed commodity is sent on a single path from the source to the corresponding terminal. We should emphasize that, in our terminology, an unsplittable flow does not necessarily contain a routing for all commodities.

3. Congestion

In this section, we consider the congestion version of the unsplittable flow problem and prove our key result.

3.1. Algorithm

Our key result shows how to efficiently transform any flow into an unsplittable flow while increasing the capacities by less than the maximum demand.

Consider any flow satisfying all demands. Without loss of generality, we assume that the flow (and thus the graph) is acyclic; indeed, any directed cycle can be eliminated by flow decomposition. Let the capacity on edge e be the flow on e. Our algorithm will modify the flow, move terminals (and eliminate them when they reach the source), and eliminate edges with zero flow. One of the invariants maintained is that, at any time, we have a flow satisfying all the current demands.

In a preliminary phase, we check for each terminal t_i whether there is an incoming edge $e=(v,t_i)$ with flow greater than or equal to d_i. If so, we move t_i to v and decrease the flow on e by d_i; if the flow on e vanishes, we delete the edge. Any unsplittable flow in the resulting instance can easily be extended to one in the original instance by executing the inverse operation. We repeat this step of moving terminals as much as possible and retain only such commodities whose terminal does not coincide with the source. Observe that in the resulting instance every vertex containing a terminal has at least two incoming edges.

Our algorithm proceeds in iterations; each iteration consists of three parts. We first find an alternating cycle, then we augment flow along this cycle, and finally we move terminals according to a certain rule. See Figure 2 for an illustration. At the beginning of any iteration, certain edges are labeled as *singular*. These are the edges (u,v) such that v and all the vertices reachable from v have out-degree at most 1. In other words, the vertices reachable from v induce a directed path. These singular edges play a special role in the construction of the alternating path and in the rule for moving terminals.

A terminal t_i at vertex v will be called *regular* if its demand d_i is greater than the flow on every edge entering v; otherwise it is called *irregular*. For example, in iteration 2 of Figure 2, the terminal with demand 4 is irregular while the terminal with demand 7 is regular. Our preliminary phase guarantees that, when it terminates, there are no irregular terminals. Our rule for moving terminals ensures that at the beginning of every iteration any vertex containing terminals has at least one regular terminal, which in turn implies that the vertex has at least two incoming edges.

We begin our search for an alternating cycle from an arbitrary vertex v. From v we follow outgoing edges as long as possible, thereby constructing a *forward* path. Since the graph is acyclic, this procedure stops, and we can only stop at a terminal, t_i. We continue with constructing a *backward* path beginning from any edge entering t_i distinct from the edge that was used to reach t_i, and following singular incoming edges as far as possible. We thus stop at the first vertex, say v',

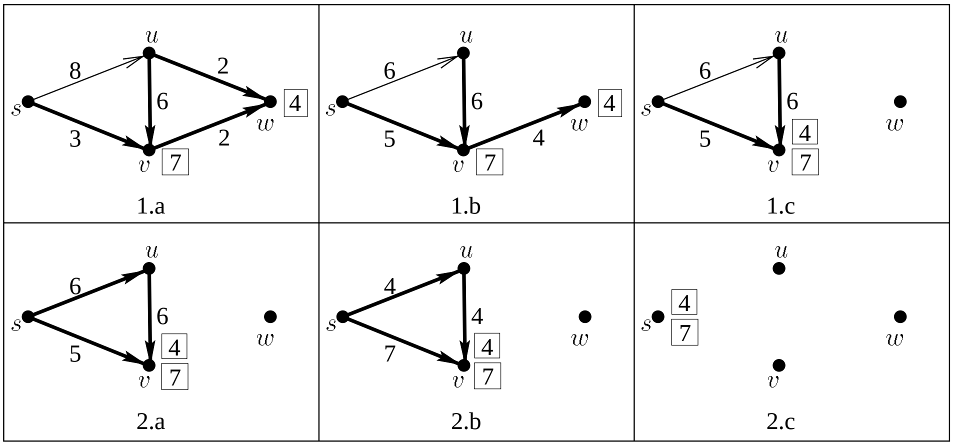

Fig. 2. An illustration of the algorithm. The two lines represent two consecutive iterations. Singular edges are emphasized. Cells 1.a and 2.a correspond to the first part of an iteration (after edges are labelled singular). Cells 1.b and 2.b correspond to after having augmented along an alternating cycle, while cells 1.c and 2.c correspond to the end of the iteration after having moved terminals. In iteration 1, the cycle $u \to w \leftarrow v \leftarrow s \to u$ was selected, $\varepsilon_1 = 2$ and $\varepsilon_2 = 2$. The terminal with demand 4 was not moved further along (s, v) or (u, v) since these edges are singular. In iteration 2, the cycle $u \to v \leftarrow s \to u$ was selected, $\varepsilon_1 = 6$, and $\varepsilon_2 = 7 - 5 = 2$. In iteration 2, there is an irregular terminal at vertex v.

which has another edge leaving it. We now continue by constructing a forward path from v'. We proceed in this manner till we reach a vertex, say w, that was already visited. This creates a cycle. If the two paths containing w in the cycle are of the same type then they both have to be forward paths, and we glue them into one forward path. Thus our cycle consists of alternating forward and backward paths (see Figure 3).

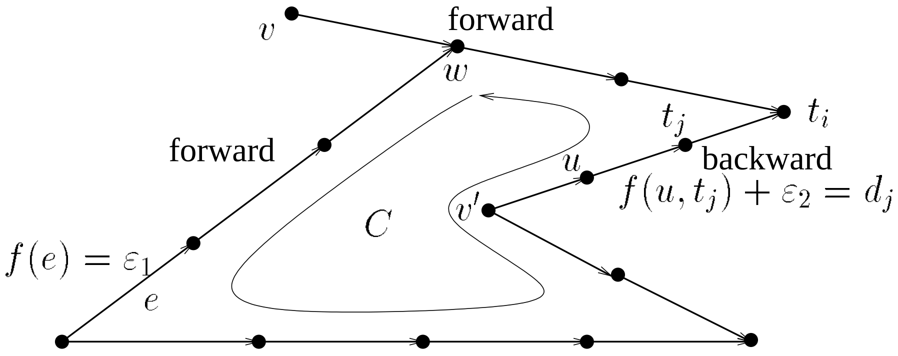

Fig. 3. An alternating cycle.

We modify the flow along this cycle by decreasing the flow along its forward paths and increasing the flow along its backward paths by the same amount. Note that such an augmentation conserves the net outflow at all vertices. Observe that, by definition of the backward paths, we increase the flow only on singular edges.

The amount by which we augment is the minimum of two values. The first one, $\varepsilon_1 > 0$, is the minimum flow along an edge on a forward path of the cycle. The second quantity, ε_2, is equal to $\min(d_j - f(e))$ where the minimum is taken over all edges $e = (u, v)$ lying on backward paths of the cycle and over all terminals t_j at v for which $d_j > f(e)$. By definition, $\varepsilon_2 > 0$.

If the minimum is achieved for an edge on a forward path then after augmentation the flow on this edge vanishes and so the edge disappears from the graph. If the minimum is achieved for an edge (u, t_j) on a backward path, then after augmentation the flow on (u, t_j) is equal to d_j.

We now move terminals according to the following rule. We move terminal t_i of commodity i to v along the edge $e = (v, t_i)$ whenever

1. e is a singular edge and $f(e) = d_i$, or
2. e is not singular and $f(e) \geq d_i$,

with a preference for 1. If t_i is moved along e then we decrease the flow on e by d_i. Thus if we ever move a terminal along a singular edge, then we decrease the flow on that edge to zero and hence remove the edge.

In particular, when the extent of the augmentation is determined by the difference between the demand d_j of a terminal and the flow along an edge $e = (u, t_j)$ on a backward path of our augmenting cycle, we increase the flow on e to d_j. It follows from our rule for moving terminals that we will move t_j, or another terminal with the same demand as t_j, along e thereby deleting this edge.

The algorithm stops when all terminals reach the source. The flow path used by commodity i (from the source to t_i) is now defined as the reverse path to the one followed by the terminal t_i to the source during the execution of the algorithm.

3.2. Analysis

We first prove the correctness of the above algorithm. In particular, we prove our claimed invariants about irregular terminals and about the existence of two edges entering every terminal.

Lemma 3.1. *Whenever an edge becomes singular, it remains so until it is removed from the graph.*

Proof. Since we never add edges, the out-degree of any vertex cannot increase. As a result, the condition for an edge to be singular remains satisfied as long as the edge exists in the graph. ∎

Lemma 3.2. *If, at the beginning of an iteration, v contains an irregular terminal t_j then (i) the out-degree of v is 0, (ii) v contains no other irregular terminal and (iii) v contains another (regular) terminal.*

For example, vertex v in iteration 2 of Figure 2 contains an irregular terminal, but it also contains a regular terminal and has out-degree equal to 0.

Proof. Since t_j is irregular, there exists an edge $e=(u,v)$ with $d_j < f(e)$ (if d_j is equal to $f(e)$, t_j would have been moved along e in the previous iteration). The presence of the irregular terminal t_j at v can only happen if t_j was moved to v, say from w. Indeed, our rule for choosing ε_2 ensures that $f(e)$ cannot be increased from under d_j to above d_j when t_j is at v.

Consider the iteration, say i, when t_j was moved to v. Since the flow on e cannot be increased from under d_j to above d_j in a later iteration and since t_j was not moved to u during that iteration, it follows that, at the end of iteration i, the flow on e is greater than d_j and the edge (u,v) is singular. By definition of singular edges, the edge (v,w) was thus also singular at the beginning of iteration i, and this was the only edge leaving v.

Since t_j was moved along the singular edge (v,w) during iteration i, this edge disappears from the graph during this iteration. As a result, the out-degree of v is 0 at the end of iteration i, showing (i). This implies that, over the course of the entire algorithm, at most one irregular terminal can be moved to v. This proves (ii). Since $f(e) > d_j$ and v has out-degree 0, flow conservation at v implies the existence of at least one more terminal at v, proving (iii). ∎

Lemma 3.3. *At the beginning of any iteration, the in-degree of any vertex containing one or more terminals is at least 2.*

Proof. By Lemma 3.2, any vertex v containing terminals must contain a regular terminal, and hence the flow on any incoming edge must be less than the maximum demand at v. By conservation of flow, the flow coming into v must be at least the sum of the demands of terminals at v. These two facts combined imply the existence of at least two edges entering v. ∎

Lemma 3.4. *As long as all terminals have not reached the source, the algorithm always finds an alternating cycle.*

Proof. While constructing a forward path, we stop at a vertex with no outgoing edges. Such a vertex must have terminals on it and so, by Lemma 3.3, it has at least two incoming edges. Hence, we can proceed with constructing a backward path.

When constructing a backward path from a terminal t_i, we can get stuck only if we arrive at a vertex with no incoming edge (this can only be the source) before we find a vertex with more than one outgoing edge. However, by Lemma 3.3, t_i has at least one incoming edge, say e, which is distinct from the first edge of our backward path, say e' (see Figure 4). The flow on e defines another path from the source to t_i, distinct from our backward path. Since both paths begin at s, this path must reach our backward path somewhere, contradicting our rule that none of the vertices on the backward path P has outgoing edges not in P. ∎

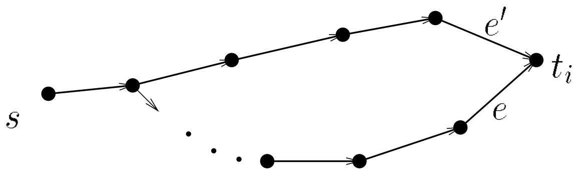

Fig. 4. End of constructing a backward path.

This shows that the algorithm does not end prematurely. We now show that the unsplittable flow constructed has the required property.

Theorem 3.5. *The algorithm described in Section* 3.1 *finds an unsplittable flow such that the total flow through any edge exceeds its capacity (or initial flow) by less than the maximum demand.*

Proof. Before an edge becomes a singular edge, the flow on it does not increase. Hence the sum of the demands of terminals that have been moved along this edge is less than or equal to the initial flow on the edge, with equality if the edge vanishes. After the edge becomes a singular edge we move at most one terminal along this edge. Thus the total unsplittable flow through this edge is less than the sum of its initial flow and the maximum demand. ∎

After we have informed Leonid Zosin on the results of this section, he achieved a slightly stronger result, as follows.

Theorem 3.6. (L. Zosin) *If the demand of an arbitrary commodity is increased to* $d_{\max}$*, there still exists an unsplittable flow violating the edge capacities by less than* $d_{\max}$.

For our applications, it will also be useful to consider the following strengthening of Theorem 3.5 which follows immediately from its proof.

Theorem 3.7. *The algorithm finds an unsplittable flow such that the sum of all but one demand using any edge* e *is less than the initial flow on* e.

Tightness. The result given in Theorem 3.5 is essentially tight as shown by the following instance, parametrized by an arbitrary integer $q \geq 2$. Its structure is very simple: all edges have unit capacity and all the $q+1$ terminals, except for one, have demand $1-\frac{1}{q}$. However, each unsplittable flow violates the capacity of some edge by $1-\frac{1}{q}$.

The construction for the case $q=4$ is described in Figure 5. The capacities of all edges are 1, the demands are shown next to the vertices. Since routing of all "small" demands is unique, we can route them first. Now the remaining capacity of each edge adjacent to s is equal to $\frac{1}{q}$. Clearly, any routing of the commodity with unit demand violates the capacity of an edge leaving the source by $1-\frac{1}{q}$. Notice also that we could replace each terminal of demand $1-\frac{1}{q}$ with a large number, say p, of terminals with demand $\frac{1}{p}-\frac{1}{pq}$; in the resulting instance, the demands are either 1 or very small.

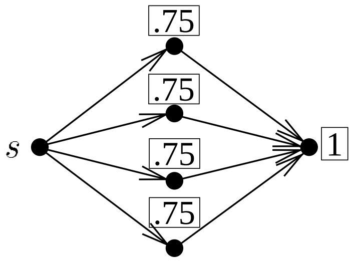

Fig. 5. A tight instance for Theorem 3.5. Demand values are indicated within rectangle boxes.

Running Time. Since every augmentation removes at least one edge, the number of augmentations is at most m, where $m=|E|$. An augmenting cycle can be found in $O(n)$ time, where $n=|V|$, since extending the alternating path by an edge requires constant time. The time for moving terminals is $O(kn)$, where k denotes the number of terminals, since there are at most n moves for each terminal, and every move can be done in $O(1)$ time. The time to update the set of singular edges at the beginning of an iteration can be seen to be proportional to the number of edges being removed from the graph in the previous iteration plus the number of edges being labeled singular in this iteration. Thus, over all iterations, this update time is $O(m)$. In each iteration to determine ε_2, it suffices to consider only those terminals which are on a backward path of the alternating cycle and to compute the difference of the demand and the flow on the edge (of the alternating cycle) entering each such terminal. Since there are k terminals, computing ε_2 requires $O(k)$ time in each iteration. Therefore the running time of our algorithm is $O(nm+km)$. It is also possible to implement the algorithm in $O(nm+kn\log k)$ time using binary heaps to keep track of the terminals at each vertex.

Remark. Notice that the undirected version of the single-source unsplittable flow problem can be treated in exactly the same fashion (and with exactly the same bounds to come). Indeed, for a single-source undirected instance, we can bidirect every edge, and compute an initial fractional flow which uses only one of every pair of anti-parallel edges: if two opposite edges have positive flow, we can simply cancel the smaller of the two flow values.

3.3. Minimizing congestion

For a graph in which the cut condition is satisfied and for which $d_{\max} \leq c_{\min}$, we can get an unsplittable flow of congestion less than 2 by first finding a flow satisfying all demands and then applying Theorem 3.5.

If the cut condition is not satisfied, Theorem 3.5 still leads to a 2-approximation algorithm. Indeed, by solving a parametric network flow problem (e.g. by binary search on α), we can first determine the smallest $\alpha \geq 1$ such that multiplying all capacities by α satisfies the cut condition. Let f be a corresponding feasible flow. By applying Theorem 3.5 to f, we get an unsplittable flow in which the total flow on e is less than $\alpha c(e) + d_{\max} \leq (\alpha+1)c(e) \leq 2\alpha c(e)$. Since α is a lower bound on the optimum congestion, this constitutes a 2-approximation algorithm.

We would like to briefly mention that if we do not assume that $d_{\max} \leq c_{\min}$ then a 5-approximation algorithm can be obtained, improving the bound of 5.8285 of Kolliopoulos and Stein [4]. As in that paper, we begin with a flow satisfying all demands, where commodity i is sent only on edges with capacity at least d_i. We divide the demands into intervals of the form $[d/2, d]$ and treat every interval separately by our algorithm, while retaining only edges of capacity at least $d/2$.

4. Minimizing the number of rounds

In this section, we apply our main result to the problem of minimizing the number of rounds. In the remainder of this paper, we again assume that $d_{\max} \leq c_{\min}$.

In a first step, we assume that there exists a feasible flow f satisfying all the demands, i.e. that the cut condition is satisfied. We show that in this case, only 5 rounds are necessary to route all commodities unsplittably. We treat "large" and "small" demands separately.

Lemma 4.1. *All demands less than or equal to* $\left(1 - \frac{1}{q}\right) c_{\min}$ *can be routed unsplittably in* q *rounds.*

For example, this shows that when $c_{\min} = 1$, two rounds are sufficient to route all demands less than $1/2$.

Proof. We first create a new graph G' with edge capacities c' as follows. Make q separate copies of G and assign to every copy of edge e a capacity equal to $c(e)/q$. For every terminal t_i in G, whose demand d_i is less than or equal to $\left(1 - \frac{1}{q}\right) c_{\min}$, create a super-terminal T_i and connect all copies of t_i to T_i with infinite capacity edges. Similarly, create a super-source S, and connect it to all q copies of source s with infinite capacity edges. Create a commodity with a demand of d_i between the super-source S and the super-terminal T_i for every i (see Figure 6).

There is a feasible flow in G' which satisfies all demands: It is induced by q flows f/q, one in each copy of G. Theorem 3.5 then gives us an unsplittable flow

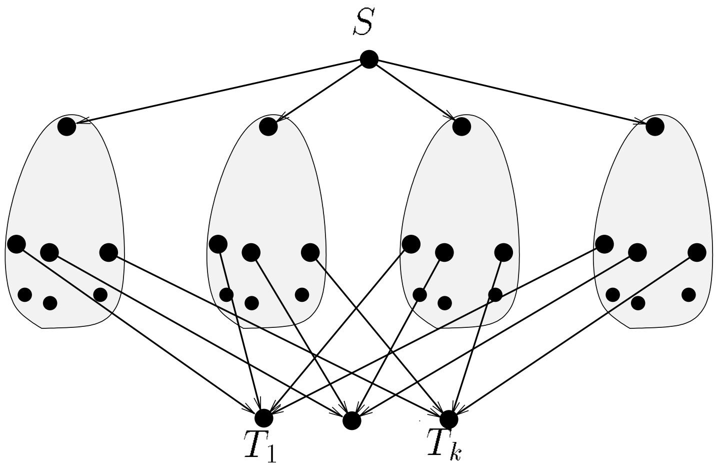

Fig. 6. Illustration for the proofs of Lemmas 4.1 and 4.2.

in G' which violates edge capacities by less than the maximum demand routed, $\left(1-\frac{1}{q}\right)c_{\min}$. Hence, in this unsplittable flow, the flow on any copy of edge e of G' exceeds its capacity of $c(e)/q$ by less than $\left(1-\frac{1}{q}\right)c(e)$. Thus the flow on any copy of e is at most $c(e)$, showing that the resulting unsplittable flow does not violate the original edge capacities. Further, any path from the super-source to a super-terminal in G' lies entirely in one of the copies of G, and hence the unsplittable flow in G' yields q unsplittable flows, one in each copy of G. These unsplittable flows satisfy all demands less than or equal to $\left(1-\frac{1}{q}\right)c_{\min}$. ∎

Lemma 4.2. *All demands in the range* $\left[\frac{d}{q-1}, d\right]$ *can be routed unsplittably in* q *rounds, where* $0 \le d \le c_{\min}$.

For example, 3 rounds are sufficient to unsplittably route all demands in the range $[0.5, 1]$. Or, if all demands are identical ($d_{\max} = d_{\min}$) and capacities are arbitrary (but $d_{\max} \le c_{\min}$) then 2 rounds are sufficient to route all commodities unsplittably.

This result is tight in the following sense. Figure 7 represents an instance of the unsplittable flow problem parametrized by n. There are $2n$ commodities with demand $1-\frac{4}{n}$ and three commodities with demand equal to 1. The instance with the capacities drawn on the edges can be seen to be feasible. However, in a single round, only $n+1$ commodities can be satisfied and, as a result, we need 3 rounds to satisfy all $2n+3$ commodities. This instance has only two distinct demand values, which can even be arbitrarily close to each other if we let n tend to infinity.

Proof. The lemma can be proved in almost the same manner as the previous lemma, except that we need to use the slightly more precise Theorem 3.7. Construct G'

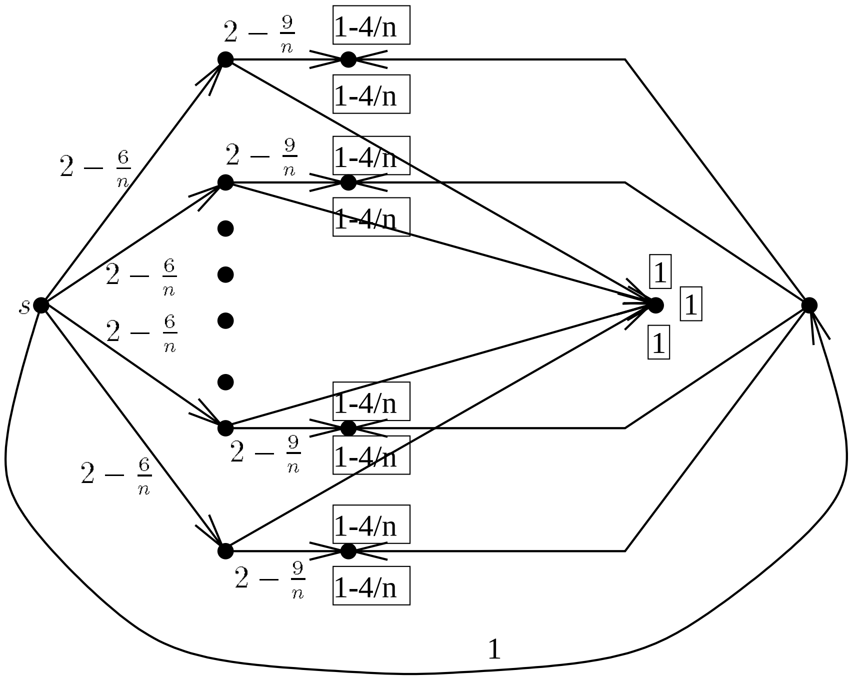

Fig. 7. 3 rounds are necessary even if the demands take only two arbitrarily close values. Demand values are indicated within rectangle boxes.

and its flow as in the proof of Lemma 4.1 by considering only those terminals whose demand is in the range $\left[\frac{d}{q-1}, d\right]$. Consider now any copy in G' of an edge e of G. Its capacity is $c(e)/q$. We consider two cases.

If $c(e) \geq \frac{q}{q-1}d$ then, as in the previous lemma, we get that the total flow on this copy of e is less than

$$\frac{c(e)}{q} + d \leq \frac{c(e)}{q} + \frac{q-1}{q}c(e) = c(e),$$

and thus the original capacity is not violated.

If $c(e) < \frac{q}{q-1}d$ then, in G', the resulting capacity is less than $\frac{d}{q-1}$. As a result, by Theorem 3.7, at most one path of our unsplittable flow uses this copy of e, and the total flow carried on it is at most $d_{\max} \leq c_{\min} \leq c(e)$.

As in the previous lemma, we have thus q unsplittable flows that collectively satisfy all demands in the range $\left[\frac{d}{q-1}, d\right]$. ∎

For both Lemma 4.1 and Lemma 4.2, the q unsplittable flows satisfying all demands can be constructed in $O(q^2mn+qkm)$ time since G' has $O(qn)$ vertices, $O(qm)$ edges and at most k commodities.

The constructions given in Lemmas 4.1 and 4.2 are similar to the construction used by Beauquier et al. [1] to show that if all demands and all capacities are equal, say to 1, then the minimum number of rounds is precisely equal to the minimum congestion.

Combining Lemma 4.1 with $q=2$ and Lemma 4.2 with $q=3$, we obtain:

Theorem 4.3. *If $d_{\max} \leq c_{\min}$ and the cut condition is satisfied, then there is a polynomial time algorithm that routes all commodities unsplittably in* 5 *rounds.*

If the cut condition is not satisfied (but $d_{\max} \leq c_{\min}$), a 5-approximation algorithm for the number of rounds can easily be devised. As for congestion, we first compute the smallest $\alpha \geq 1$ such that there exists a feasible flow satisfying all demands when the capacities are multiplied by α. Clearly, the minimum number of rounds must be at least $\lceil \alpha \rceil$ since the superposition of q rounds of unsplittable flow gives a congestion of q. Now, use the same constructions as in Lemmas 4.1 and 4.2, except that the number of copies of G is now

$$\lceil \alpha q \rceil \leq q \lceil \alpha \rceil.$$

In this case, a feasible flow in the aggregated network is induced by $\lceil \alpha q \rceil$ flows $\frac{f}{\lceil \alpha q \rceil} \leq \frac{f}{\alpha q}$. We will then be able to route in an unsplittable manner all commodities whose demands are less or equal to $c_{\min}/2$ in $\lceil 2\alpha \rceil$ rounds and all commodities whose demands are greater or equal to $c_{\min}/2$ in $\lceil 3\alpha \rceil$ rounds, giving a total of $\lceil 2\alpha \rceil + \lceil 3\alpha \rceil \leq 5\lceil \alpha \rceil$ rounds.

Theorem 4.4. *There exists a* 5*-approximation algorithm for the problem of minimizing the number of rounds when $d_{\max} \leq c_{\min}$.*

5. Maximizing the routable demand

From Theorem 4.3, we immediately get that if $d_{\max} \leq c_{\min}$ and the cut condition is satisfied then we can efficiently route unsplittably a fifth of the total demand $d_{\text{tot}} = \sum_i d_i$. However, a better bound of $0.226\ldots$ can be obtained as we show in this section. The previous best bound was a fraction 0.075 of the maximum routable demand [4].

Our improvement is based on two results. The first generalizes the constructions in Lemmas 4.1 and 4.2, while the second relies upon the fact that the unsplittable flow problem is easy when all demands are equal and all capacities are multiple of this common demand. We start with the extension of Lemmas 4.1 and 4.2.

Lemma 5.1. *Assume that the cut condition is satisfied. Then, for any $0<\beta<1$, a fraction β of the total demand can be routed unsplittably if either*

1. $d_{\max} \le \left(\frac{1}{\beta}-1\right) d_{\min}$, *or*
2. $d_{\max} \le (1-\beta)c_{\min}$.

Furthermore, if $\beta=\frac{p}{q}$ where p and q are integers then such an unsplittable flow can be obtained in time $O(q^2mn+pqkm)$.

In fact, our algorithm will only be using condition 2.

Proof. Let f be any feasible flow in G satisfying the demands. First assume that $\beta=\frac{p}{q}$ for integers p and q. Construct G' as in Lemma 4.1 or Lemma 4.2 except that we place p terminals with demand d_i at the super-terminal T_i, for every i. A feasible flow in G' can be obtained by taking in each of the q copies of G the flow βf. Observe that G' has $O(qn)$ vertices, $O(qm)$ edges and pk terminals, and thus the algorithm in Section 3.1 runs in time $O(q^2mn+pqkm)$.

We claim that the unsplittable flow returned by Theorem 3.7 induces q feasible unsplittable flows in G, which collectively satisfy every demand p times. The fact that the original capacities are respected follows from the same argument as in Lemmas 4.1 (in the case of 1) and 4.2 (in the case of 2). Furthermore, the p paths reaching the super-terminal T_i in the unsplittable flow in G' must be in different copies of G. This follows from the fact that the initial flow on the edge between a copy of t_i and T_i is equal to $\beta d_i < d_i$, and thus by Theorem 3.7 at most one path uses such an edge. This proves the claim.

Since the average demand satisfied in each copy of G is $\frac{p}{q}d_{\text{tot}}$, the unsplittable flow in one of these copies must have a value at least that large. This proves the theorem for rational values of β.

If β is irrational then let D be the largest sum of demands less than βd_{tot}. Select p and q such that $D/d_{\text{tot}} < p/q < \beta$, and this is possible since $D < \beta d_{\text{tot}}$. Using the previous construction with p and q, we get an unsplittable flow of value greater or equal to $\frac{p}{q}d_{\text{tot}} > D$ and this value must be at least βd_{tot} given the definition of p,q and D. ∎

Our second result for approximating the maximization version is based on the classical case in which all demands $d_i = d$ are equal, and all capacities are integer multiples of d. In this case, all versions (feasibility, congestion, number of rounds and maximization versions) of the unsplittable flow problem can be solved efficiently. The feasibility and maximization versions can be solved as a maximum flow problem since we can find a maximum flow in such a graph with all flows multiple of d. This remains true for weighted maximum flows where the objective is to maximize a weighted sum of the flow into each terminal (this can be proved by a reduction to a minimum cost flow problem); this is the crucial property we use.

Remark. Kleinberg [2, 3] also shows that in this setting the minimum number of rounds needed is equal to the minimum congestion rounded up. For this purpose, he uses the fact that the routable subsets of terminals form a matroid and Edmonds' theorem on covering the ground set of a matroid by bases. In the special case in which all capacities are equal to d, this has also been obtained by Beauquier et al. [1] using a construction similar to the one we used in the previous lemma and the previous section.

Lemma 5.2. *Assume that the cut condition is satisfied and let* $0<\beta<c_{\min}$. *Then an unsplittable flow of value greater than*

$$\sum_{i:d_i\le\beta}\frac{d_i}{2\beta}d_i$$

can be found in polynomial time.

Proof. We modify the demands and the capacities of G as follows. The demand d_i is replaced by $d'_i=0$ if $d_i>\beta$ and by $d'_i=\beta$ if $d_i\le\beta$. The new capacity $c'(e)$ of edge e is obtained by rounding down $c(e)$ to an integer multiple of β. Let G' denote the resulting instance. Observe that, by construction, any unsplittable flow in G' automatically leads to an unsplittable flow in G. In the new instance, we define the weight of terminal t_i to be d_i, and we consider a flow maximizing $\sum_i d_i x_i$ where x_i denotes the flow sent to t_i. By the discussion preceding the lemma, we know that this maximum flow can be assumed to be unsplittable with $x_i\in\{0,\beta\}$ for every i. Given the construction of G', this unsplittable flow is also feasible for G. We claim that it has the required property. Let T be the set of commodities routed in this unsplittable flow, i.e. let $T=\{i:x_i=\beta\}$.

Since $c'(e)>c(e)/2$, any feasible flow in G (whose existence is guaranteed by the cut condition) can be transformed into a feasible flow in G' by multiplying each flow value by a factor greater than half and disregarding the flow sent to commodities for which $d_i>\beta$. This produces a flow with $x_i>\frac{d_i}{2}$ for all i with $d_i\le\beta$. Its value with regard to the objective of the sum of the flow into each terminal weighted by d_i is thus greater than $\sum_i\frac{d_i^2}{2}$. Comparing this value to the one for our unsplittable flow satisfying the demands in T, we derive that

$$\beta\sum_{i\in T}d_i>\sum_i\frac{d_i^2}{2},$$

implying the required expression. Finally, the construction described in this proof can be carried easily in polynomial time. ∎

In fact, the fraction $\frac{1}{2}$ in the expression in Lemma 5.2 can be improved to $1-\frac{1}{l}$ for an integer l if $\beta l\le 1$ (since, in this case, $c'(e)>(1-\frac{1}{l})c(e)$), but we will not be using this result.

We are now ready to describe our algorithm for approximating the maximum unsplittably routable demand. For simplicity, let $c_{\min}=1$. Fix any integer q. For any integer $p \leq q$, let A_p denote the unsplittable flow obtained by applying (part 2 of) Lemma 5.1 to the demands less or equal to $\frac{p}{q}$ with $\beta = 1 - \frac{p}{q}$. Similarly, let B_p denote the unsplittable flow obtained by applying Lemma 5.2 with $\beta = \frac{p}{q}$. Among all these unsplittable flows, our algorithm returns the one of maximum value.

Theorem 5.3. *For any integer q, the algorithm described above produces an unsplittable flow of value γd_{tot} where*

$$\frac{1}{\gamma} < \frac{2q}{q-1} + \sum_{i=r-1}^{q-2} \frac{2}{i} + \frac{q}{r-1} \sum_{j=q-r+1}^{q-1} \frac{1}{j} < 2\frac{q}{q-1} + 2\ln\frac{q-2}{r-2} + \frac{q}{r-1}\ln\frac{q-1}{q-r},$$

for any integer r with $2 < r < q$.

Proof. Let x_i, for $1 \leq i \leq q$ denote the sum of all demands greater than $\frac{i-1}{q}$ and less or equal to $\frac{i}{q}$. Let a_p (resp. b_p) denote the total demand satisfied by A_p (resp. B_p). From Lemma 5.1, we have that

$$a_p \geq \frac{q-p}{q}\left(x_1 + x_2 + \cdots + x_p\right).$$

Similarly, from Lemma 5.2, we derive that

$$b_p > \sum_{i=1}^{p} \frac{i-1}{2p} x_i.$$

In order to obtain a lower bound on the worst performance of our algorithm, we therefore need to solve the following linear program:

$$\begin{array}{lll}
\text{Minimize} & z & \\
(1) & \text{subject to:} \quad z \geq \frac{q-p}{q}\left(\sum_{i=1}^{p} x_i\right) & p \geq 1, \\
(2) & \qquad\qquad\quad z \geq \sum_{i=1}^{p} \frac{i-1}{2p} x_i & p \leq q, \\
 & \qquad\qquad\quad \sum_{i=1}^{q} x_i = 1, &
\end{array}$$

the last constraint normalizing d_{tot} to be 1. To prove a lower bound on the optimum value of this linear program, we implicitly construct a dual solution. Choose any integer r, $2 < r < q$, and let

$$s_p = \begin{cases} \frac{q}{(r-1)(q-p)} & 1 \leq p \leq r-1 \\ 0 & p \geq r, \end{cases}$$

$$t_p = \begin{cases} 2q/(q-1) & p = q \\ 2/(p-1) & r \leq p < q\ . \\ 0 & p < r. \end{cases}$$

Multiplying (1) by s_p and (2) by t_p and summing them up, we derive after some manipulations that

$$\left(\sum_p s_p + \sum_p t_p\right) z \geq \sum_{i=1}^{q} x_i = 1.$$

Because of the strict inequality for b_p, we derive that

$$\frac{1}{\gamma} < \left(\sum_p s_p + \sum_p t_p\right).$$

Rearranging this expression gives the first inequality valid for any r. To derive the second inequality, we simply use the fact that $\sum_{i=a}^{b} \frac{1}{i} < \int_{a-1}^{b} \frac{1}{x} dx = \ln b - \ln(a-1)$. ∎

In particular, for $q = 10$, Theorem 5.3 gives $\gamma > 0.214$ and $\frac{1}{\gamma} < 4.6565$ (for $r = 8$), while for $q = 1010$, it gives $\gamma > 0.226$ and $\frac{1}{\gamma} < 4.4248$. As q tends to infinity and r/q tends to x, the expression given in the theorem tends to $2 + 2\ln\frac{1}{x} + \frac{1}{x}\ln\frac{1}{1-x}$ which is minimized for $x = 0.7378\ldots$ with value < 4.4226.

Corollary 5.4. *The algorithm preceding Theorem 5.3 with* $q = 1010$ *outputs an unsplittable flow satisfying more than* 0.226 *of the total demand whenever the cut condition is satisfied and* $d_{\max} \leq c_{\min}$.

One can show that our bound of 0.226... cannot be improved if we rely only upon Lemmas 5.1 and 5.2. This is quite technical and omitted here.

When the cut condition is not satisfied, we can nevertheless approximate the maximum unsplittably routable demand within a factor less than 4.43. Removing the cut condition assumption is however less straightforward than for the congestion and number of rounds cases. The algorithm goes as follows. First compute a maximum flow in G, and let $\alpha_i d_i$ ($\alpha_i \leq 1$) be the amount of flow routed to t_i. Lemma 5.2 easily extends to show that an unsplittable flow of value

$$\sum_{i:d_i \leq \beta} \frac{d_i}{2\beta} \alpha_i d_i$$

can be found in polynomial time. We can also extend Lemma 5.1 to show for example that an unsplittable flow of value $\beta \sum_{i:d_i \leq 1-\beta} \alpha_i d_i$ exists, and that a $(1-\epsilon)$ fraction of this quantity can be found in polynomial time. Once these extensions are proved, the result follows from the proof of Theorem 5.3; the only difference is that in the definition of x_i each demand is multiplied by its corresponding factor α_i.

To prove the extension of Lemma 5.1, we need to further modify our construction of G', to account for the fact that the fraction of demand satisfied depends on the commodity. This is done in more general terms in the following theorem.

Theorem 5.5. *Consider any flow f in G and let $\alpha_i d_i$ (where $\alpha_i \leq 1$) denote the amount of flow routed to t_i. Then there exists an unsplittable flow satisfying at least a total demand of $\sum_i \alpha_i d_i$ such that the sum of all but one demand using any edge e is less than $f(e)$. Furthermore, such an unsplittable flow of value at least $(1-\epsilon)\sum_i \alpha_i d_i$ can be obtained in polynomial time for any $\epsilon > 0$.*

Proof. Let q be any integer, and let $p_i = \lfloor \alpha_i q \rfloor$. Let $g \leq f$ be a flow in G such that $\frac{p_i}{q} d_i$ units are routed to t_i. Construct G' as in Lemma 5.1, except that we place p_i terminals at T_i, each with a demand of d_i. A flow in G' can be obtained from q copies of flow g. Now apply the algorithm of Section 3.1 and Theorem 3.7. As in the proof of Lemma 5.1, each request at T_i will be satisfied in a different copy of G and therefore one of these copies must contain an unsplittable flow of value at least $\sum_i \frac{p_i}{q} d_i$. It also follows that the sum of all but one demand using any edge e is less than $g(e) \leq f(e) \leq c(e)$.

The value of this unsplittable flow differs from our target of $\sum_i \alpha_i d_i$ by at most d_{tot}/q. When q is sufficiently large (as in Lemma 5.1), the flow value is guaranteed to be at least our target value.

For a polynomial running time, we need to choose q to be polynomially bounded. If we let $q = k/\epsilon$ where k denotes the number of terminals, then $d_{\text{tot}}/q = \epsilon d_{\text{tot}}/k \leq \epsilon d_{\max}$. If $\sum_i \alpha_i d_i \geq d_{\max}$ then the unsplittable flow produced has value at least $(1-\epsilon)\sum_i \alpha_i d_i$; otherwise, simply outputting a path to one of the terminals of demand $d_{\max}$ gives the desired property. ∎

Theorem 5.6. *There exists a 4.43-approximation algorithm for the problem of maximizing the total demand that can be sent unsplittably when $d_{\max} \leq c_{\min}$.*

5.1. A bad instance

In this section, we describe a family of instances showing that, even if the cut condition is satisfied, there exist instances for which no fraction greater than $\frac{1}{2} - \frac{\ln 2}{6} + \epsilon < 0.3845$ of the total demand can be satisfied unsplittably.

These instances, parametrized by p and q, are described in Figure 8. They can be obtained by the superposition of q isomorphic graphs, and only one such graph is described in Figure 8. There are $2p+2$ vertices common to all q graphs, denoted by s (the source), u, a_i and t_i for $i = 1, \cdots, p$. There are also $p+1$ vertices unique to each graph, denoted in the jth graph ($j = 1, \cdots, q$) by r_j, v_{ij} for $i = 1, \cdots, p$. There are $2p+2q$ commodities: Two terminals at each vertex r_j ($j = 1, \cdots, q$) with demand $0.5 + \epsilon$, where $\epsilon = \frac{1}{2p+2q}$, and two terminals at each vertex t_i with demand $d_i + \epsilon$, where

$$d_i = \left(1 - \frac{2}{q}\right)^{i-1}$$

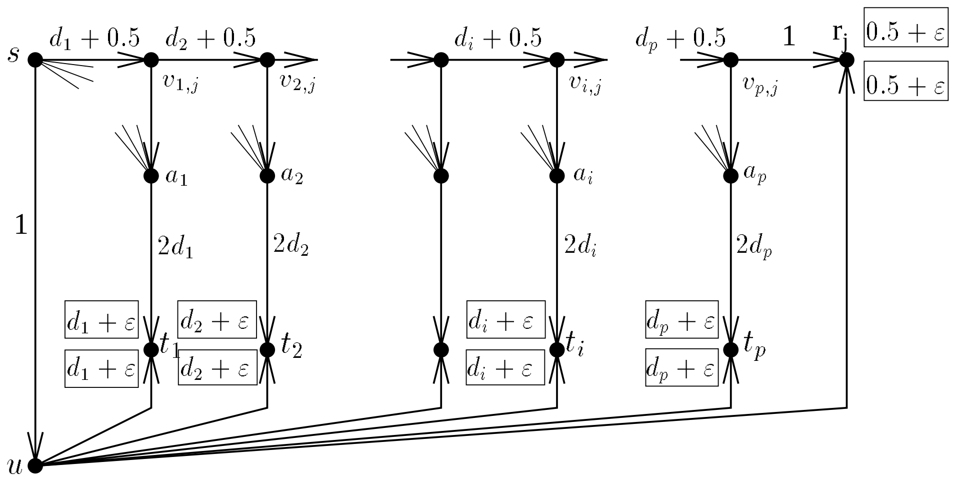

Fig. 8. A bad instance for the maximum routable demand. Only part j of the graph is represented. Demand values are indicated within rectangle boxes.

for $i = 1, \cdots, p$. We choose p so that $d_{p+1} \geq 0.5 \geq d_{p+2}$, i.e. $p = \frac{\ln 2}{2}(q + o(q))$. The capacities of the edges, if finite, are indicated on the figure (and are the same in all q graphs). Observe that $d_{\max} \leq c_{\min}$.

One can show that the instance can be satisfied fractionally by the following flow which saturates every edge of finite capacity: send ϵ units of every demand through u, and send $d_i - d_{i+1} = d_i(1-(1-2/q)) = \frac{2d_i}{q}$ units on the edge (v_{ij}, a_i) which adds up to $2d_i$ on the edge (a_i, t_i) (since there are q copies of the graph depicted in the Figure). Our condition on p guarantees that there exists enough capacity to send 1 unit directly to r_j through the v_{ij}'s.

The total demand in this instance is equal to:

$$q + 2\sum_{i=1}^{p} d_i + (2p+2q)\epsilon = q + 2\sum_{i=1}^{p}\left(1 - \frac{2}{q}\right)^{i-1} + 1 = q + 1 + 2\frac{q}{4} + o(q) = \frac{6}{4}q + o(q),$$

where we have used the fact that

$$\sum_{i=1}^{p} d_i = \sum_{i=1}^{p}\left(1 - \frac{2}{q}\right)^{i-1} = \frac{1-(1-2/q)^p}{1-(1-2/q)} = \frac{q}{2}\left(\frac{1}{2} + o(1)\right) = \frac{q}{4} + o(q). \tag{3}$$

In one round, we claim that we can satisfy unsplittably only $q+1$ commodities, one through u and one for the jth copy of the graph ($j = 1, \cdots, q$). Moreover, we cannot simultaneously satisfy the two commodities sharing the same terminal t_i for $i = 1, \cdots, p$, except for one such terminal (through u). Hence, the maximum demand one can possibly satisfy in one round is (using the fact that $p<q$ and (3)):

$$d_1+\sum_{i=1}^{p}(d_i+\epsilon)+(q-p)(0.5+\epsilon)=\frac{q}{4}+\frac{q-p}{2}+o(q)=\frac{3-\ln 2}{4}q+o(q).$$

As a result, the maximal fraction that can be routed unsplittably is equal to $\frac{3-\ln 2}{6}+o(1)<0.3845$.

6. Special cases

In this last section we consider two special cases of the single-source unsplittable flow problem.

6.1. Integral capacities

When capacities are integral and the cut condition is satisfied, the situation can be dealt with easily using Theorem 3.7.

Lemma 6.1. *If all capacities are integer multiples of c_0, $c_0\geq d_{\max}$ and $d_{\min}\geq c_0/q$ for $q>1$ then all commodities can be routed unsplittably in q rounds.*

Proof. The construction is identical to the ones in Lemma 4.1 and 4.2. An edge e of capacity c_0 in G (if any) has a capacity $c_0/q\leq d_{\min}$ in G', implying that at most one path of our unsplittable flow will be using this edge by Theorem 3.7. If edge e has capacity pc_0 for an integer $p\geq 2$, then the total demand routed through a copy of e will be $\frac{pc_0}{q}+d_{\max}\leq\frac{pc_0}{q}+c_0=pc_0(\frac{1}{q}+\frac{1}{p})\leq pc_0$ (since $p,q\geq 2$), and hence the resulting q unsplittable flows are feasible in G and cover the entire demand. ∎

Lemma 6.1 can also be easily (or more easily) proved by rounding up all demands to $c_{\min}$. This lemma together with Lemma 4.1 gives the following corollary.

Theorem 6.2. *If all capacities are integral and $d_{\max}\leq 1$ then we can unsplittably route*

1. *in 2 rounds all demands greater or equal to $1/2$ or all demands less or equal to 1/2,*
2. *in 3 rounds all demands greater or equal to $1/3$, and*
3. *in 4 rounds all demands.*

6.2. Integral capacities and demands power of 2

We now consider the case in which all demands are of the form 2^{-p} for $p\in N$ and all capacities are integral.

If we were considering the Bin Packing problem, the restriction to weights of size 2^{-p} for $p\in N$ (and unit-sized bins) would make the problem easy. Indeed, by considering the weights in non-increasing order, we are guaranteed that the unused part of the current bin is a multiple of the current item; as a result, if the bin is not full, then the next item will always fit in the current bin. This implies that algorithms such as FIRST-FIT-DECREASING or NEXT-FIT-DECREASING lead to an optimum bin packing in this case (and also whenever, for any two weights, one is a multiple of the other). This can be extended to the context of single-source multicommodity flows.

Theorem 6.3. *Assume all capacities are integral and all demands are of the form* 2^{-p} *with* $p \in N$. *Then there exists a maximum flow such that the flow sent to* t_i *is either 0 or* d_i, *for every* i.

Proof. In order to find a maximum flow, use an augmenting path algorithm that considers the commodities in non-increasing order of demands and, for each commodity, pushes as much flow as possible. We make the inductive hypothesis that, when a commodity i with demand d_i is considered, the residual capacities are all multiples of d_i. As a result, either the full demand of commodity i will be satisfied (if t_i can be reached from s in the residual graph), or none of it (if t_i is unreachable from s in the residual graph). Furthermore, this implies that the inductive hypothesis continues to hold after augmentation. This proves the desired result since the resulting flow is maximum and has the desired properties. ∎

Although the flow augmentation is carried on a single path in the previous proof, the resulting flow is not necessarily unsplittable (since augmenting flow in the residual graph might mean canceling flow of a commodity with larger demand in the original graph). In fact, the problem remains hard; for example, the congestion version of this problem cannot be approximated within a factor better than 3/2 unless P=NP [5] (see also [4]). We can nevertheless improve somewhat the results of the previous sections.

Theorem 6.4. *Assume all capacities are integral and all demands are of the form* 2^{-p} *with* $p \in N$. *If the cut condition is satisfied then all commodities can be routed with congestion* $1 + d_{\max} - d_{\min}$.

This theorem was proved by Kolliopoulos and Stein [4] when all $d_i \in \{0.5, 1\}$.

Proof. The theorem follows immediately from Theorem 3.5. Indeed, the corresponding unsplittable flow violates the capacities by less than $d_{\max}$. However, since the flow on any edge is a multiple of $d_{\min}$, no capacity is violated by more than $d_{\max} - d_{\min}$. ∎

The same approach also leads to an approximation algorithm for congestion with a performance guarantee of $1+d_{\max}-d_{\min}$ when all capacities are integral and all demands are of the form 2^{-p} with $p \in N$. This is best possible whenever $d_{\max} = 2d_{\min}$ since Kolliopoulos (personal communication) has extended a construction of Lenstra et al. [5] to show that no approximation algorithm for congestion (in this setting) has a performance guarantee better than $1 + d_{\min}$ unless P=NP.

For the number of rounds version of the problem, we obtain the following result.

Theorem 6.5. *Assume all capacities are integral and all demands are of the form 2^{-p} with $p \in N$. If the cut condition is satisfied then all commodities can be routed in 3 rounds.*

Proof. Because of integrality, all commodities with unit demand can be routed unsplittably in one round. The other commodities can be routed unsplittably in two rounds by Lemma 4.1 since their maximum demand is at most half. ∎

To obtain an improvement for the maximization version, we first need to derive a strengthening of Lemma 5.2 for the setting considered here.

Lemma 6.6. *Consider an instance with integral capacities and demands of the form 2^{-p} where $p \in N$. Assume that the cut condition is satisfied and let r be any nonnegative integer. Then an unsplittable flow of value at least*

$$\sum_{i:d_i \le 2^{-r}} \frac{d_i}{2^{-r}} d_i$$

can be found in polynomial time.

Proof. The proof is identical to the proof of Lemma 5.2, except that we do not need to modify the capacities of the edges thereby saving a factor of 2 in the expression for the flow. ∎

Theorem 6.7. *Assume all capacities are integral and all demands are of the form 2^{-p} with $p \in N$. If the cut condition is satisfied, $1/2.6067\ldots = 0.3836\ldots$ of the total demand can be routed unsplittably, where $2.6067\ldots = 1 + \sum_{p=1}^{\infty} \frac{1}{2^p - 1}$.*

Proof. The proof is similar to the proof of Theorem 5.3. From Lemma 6.6 with $r = 0$, we know that we can find an unsplittable flow of value

$$z \ge D_1 + \frac{1}{2} D_{1/2} + \frac{1}{4} D_{1/4} + \cdots + \frac{1}{2^p} D_{1/2^p} + \cdots, \tag{4}$$

where $D_{1/2^p}$ denotes the total demand of commodities having demand $1/2^p$.

Moreover, by Lemma 5.1, we can also find an unsplittable flow of value

$$z \ge (1 - \beta) \sum_{d \le \beta} D_d, \tag{5}$$

for any β. Adding a $\frac{\beta}{1-\beta}$ multiple of inequality (5) for $\beta = 2^{-p}$ with $p \ge 1$ to inequality (4), we obtain that:

$$\left(1 + \sum_{p=1} \frac{2^{-p}}{1 - 2^{-p}} \right) z \ge D_1 + D_{1/2} + D_{1/4} + \cdots$$

or,

$$\left(1+\sum_{p=1}^{\infty}\frac{1}{2^p-1}\right)z \geq d_{\text{tot}}. \qquad \blacksquare$$

The worst instance we are aware of (satisfying the assumptions of Theorem 6.7) allows at most a 0.75 fraction of the maximum flow to be routed unsplittably, see Figure 9. Notice also that the demand of this instance can be satisfied unsplittably in two rounds while Theorem 6.5 guarantees 3 rounds.

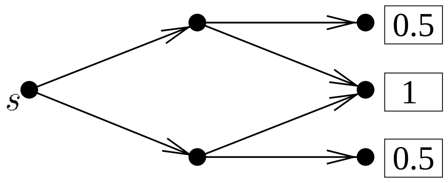

Fig. 9. A 0.75 fraction instance (all edge capacities are 1).

Acknowledgements. We would like to thank Stavros Kolliopoulos for pointing out the hardness of approximation result of Shmoys, Tardos and Lenstra and its extensions for the case considered in Section 6.2.

References

[1] B. Beauquier, P. Hell and S. Perennes: Optimal wavelength-routed multicasting, *Discrete Applied Mathematics*, **84** (1998), 15–20.

[2] J. M. Kleinberg: Single-source unsplittable flow, in: *Proc. 37th Annual Symposium on Foundation of Computer Science (FOCS'96)*, (1996), 68–77.

[3] J. M. Kleinberg: Approximation algorithms for disjoint paths problems, Ph.D. dissertation, M.I.T., 1996.

[4] S. G. Kolliopoulos and C. Stein: Improved approximation algorithms for unsplittable flow problems, in: *Proc. 38th Annual Symposium on Foundation of Computer Science (FOCS'97)*, (1997), 426–435.

[5] J. K. Lenstra, D. B. Shmoys and É. Tardos: Approximation algorithms for scheduling unrelated parallel machines, *Mathematical Programming*, **46** (1990), 259–271.

[6] D. B. Shmoys and É. Tardos: An approximation algorithm for the generalized assignment problem, *Mathematical Programming*, **62** (1993), 461–474.

Yefim Dinitz (E. A. Dinic)

Department of Computer Science,
Ben-Gurion University of the Negev,
Beer-Sheva, Israel.
dinitz@cs.bgu.ac.il

Naveen Garg

Department of Computer Science
and Engineering
Indian Institute of Technology,
New Delhi
naveen@cse.iitd.ernet.in

Michel X. Goemans

CORE, 34 Voie du Roman Pays,
B-1348 Louvain-La-Neuve, Belgium
goemans@core.ucl.ac.be