Improved approximation algorithms for unsplittable flow problems

(Extended Abstract)

Stavros G. Kolliopoulos[1]   Clifford Stein[1]

Abstract

In the single-source unsplittable flow *problem we are given a graph G, a source vertex s and a set of sinks $t_1, \ldots, t_k$ with associated demands. We seek a single s-t_i flow path for each commodity i so that the demands are satisfied and the total flow routed across any edge e is bounded by its capacity c_e. The problem is an NP-hard variant of max flow and a generalization of single-source edge-disjoint paths with applications to scheduling, load balancing and virtual-circuit routing problems. In a significant development, Kleinberg gave recently constant-factor approximation algorithms for several natural optimization versions of the problem [18]. In this paper we give a generic framework that yields simpler algorithms and significant improvements upon the constant factors. Our framework, with appropriate subroutines, applies to all optimization versions previously considered and treats in a unified manner directed and undirected graphs.*

To give a flavor of our results, consider minimizing relative congestion, *i.e. the maximum ratio over all edges e of the flow through e divided by the capacity c_e. This metric was a primary testbed for randomized rounding techniques and has been studied extensively. We give a simple $(3.23 + o(1))$-approximation algorithm for both directed and undirected graphs. The previously known bounds were 16 for the directed and 8.25 for the undirected case. Our approach also gives the first constant-factor approximation for minimum-cost unsplittable flow on directed graphs and improves considerably upon the approximation ratio for the minimum cost version on undirected graphs.*

The algorithmic techniques we introduce are quite general and apply to related problems as well. For example we use them to give a constructive proof of the following fact. If there exists an algorithm $\mathcal{A}$ for (multisource) edge-disjoint paths such that $\mathcal{A}$ outputs an approximation for relative congestion that is ρ times the fractional optimum, then a corresponding $O(\rho)$-approximation algorithm exists for multisource unsplittable flow with arbitrary demands and capacities. On the negative side we show that for the problem with two sources, no ρ-approximation with $\rho < 2$ can be achieved, unless P=NP.

We also examine connections with parallel machine scheduling. A lower bound of $3/2$ exists for the approximability of minimum makespan on parallel machines when a job j can be processed only on a subset $M(j)$ of the available machines. This scheduling problem is known to have an approximation-preserving reduction to single-source unsplittable flow. The lower bound of $3/2$ actually holds when processing times (or demands) take the values p or $2p$ for some $p > 0$. We give a best possible, unless P=NP, $3/2$-approximation for this restricted unsplittable flow problem and generalizations to other restricted sets of demands.

[1]Dartmouth College, Department of Computer Science, Hanover, NH 03755-3510. E-mail: {`stavros, cliff`}`@cs.dartmouth.edu`. Research partly supported by NSF Award CCR-9308701 and NSF Career Award CCR-9624828.

1 Introduction

In the *single-source unsplittable flow* problem, we are given a network G, a source vertex s and a set of sinks $t_1, \ldots, t_k$ with associated demands $\rho_1, \ldots, \rho_k$. We seek a single s-t_i flow path for each commodity i so that the demands are satisfied and the total flow routed across any edge e is bounded by the capacity c_e. The minimum edge capacity is assumed to have value at least $\max_i \rho_i$. The problem is an NP-hard variant of maximum flow and a generalization of single-source edge-disjoint paths. Single-source unsplittable flow is also important as a unifying framework for a variety of scheduling and load balancing problems [18]. It can model bin-packing [18], certain scheduling problems on parallel machines [22] and virtual-circuit routing. For the more general multiple-source disjoint paths problem a large amount of work exists for solving exactly interesting special cases (e.g. [6], [27], [26]) or approximating with limited success various objective functions (e.g. [25],[7], [19], [20]). Shedding light on single-source unsplittable flow may lead to a better understanding of this more general problem.

Three main optimization versions of unsplittable flow can be defined.

minimum congestion. Find a flow which minimizes *relative congestion,* i.e. the maximum ratio over all edges e of flow through e divided by the capacity c_e. Minimizing congestion was a primary testbed for the randomized rounding technique of Raghavan and Thompson [25] and has been studied extensively for its connections to multicommodity flow and cuts (e.g. [17, 21, 16]). In fact randomized rounding gives a logarithmic approximation for the multiple-source unsplittable flow problem.

maximum routable demand. Find a subset of the sinks of maximum total demand that can be routed while respecting capacity constraints.

minimum number of rounds. Route all the sinks in the minimum number of communication rounds so that during each round capacities are respected.

In a major development, Kleinberg [18] recently gave constant-factor approximation algorithms for all three optimization versions presented above on both directed and undirected graphs. In the same paper additional motivation for unsplittable flow is presented.

Our results. We give a new approach for single-source unsplittable flow that has three main advantages.

- The algorithms are simple and do not need the machinery developed in [18].
- Our approach treats in a unified manner directed and undirected graphs.
- We obtain significant improvements upon the approximation ratios for all three optimization versions.

All our algorithms follow the same generic framework. We first find a maximum (splittable) flow. We then use information from this maximum flow to allocate capacity to different subproblems, where each subproblem contains demands in a fixed range. A separate near-optimal solution to each subproblem is found and the solutions are then combined. When combining the solutions one might expect a blow-up of the approximation guarantee. Our conceptual contribution lies into showing that this is not the case given an appropriate capacity allocation. For example, for minimizing congestion we can show the rather counterintuitive fact that there exists a way of dividing the set of demands into n groups so that the total congestion of the combined solution is within a constant factor of the congestion of any group. We note that a grouping scheme is also used in [18], though in a more complicated way; ours is based solely on demand values and does not require any information from the graph. We now elaborate on our results.

Minimum congestion. We give a $(3.23 + o(1))$-approximation algorithm for both directed and undirected graphs. The previously known bounds were 16 for the directed and 8.25 for the undirected case. Our approach gives the first constant factor approximation for minimum-cost unsplittable flow on directed graphs; it also improves considerably upon the constants known for the minimum cost version on undirected graphs. In particular, existing results for undirected graphs [18] give a simultaneous $(7.473, 10.473)$ approximation for cost and congestion. We give a $(1.68, 3.23+o(1))$ simultaneous approximation on both directed and undirected graphs. Moreover we show how to obtain a $1+\delta$ approximation for cost, for any $\delta > 0$, at the expense of a larger constant factor for congestion.

Maximum routable demand. We show how to route at least .075 of the optimum. In [18] the constant is not given explicitly but it can be as low as .031 for undirected graphs and of the order of 10^{-9} for directed graphs.

Minimum number of rounds. We show how to route all the demands in at most 13 times the optimum number of rounds, an improvement upon the 32 upper bound given in [18].

We emphasize that the algorithms for all three different versions are simple and make use of the same generic framework. Although they are presented separately for ease of exposition, they could all be stated in terms of one algorithm, with different subroutines invoked for subproblems. The dominant computational steps are maximum flow and flow decomposition; these are tools that work well on both directed and undirected graphs. In [18] it is noted that "the disjoint paths problem is much less well understood for directed than it is for undirected graphs ... in keeping with the general principle we will find that the algorithms we obtain for directed graphs will be more complicated".

Generalizations. The algorithmic techniques we introduce are quite general and can be applied to related problems. For example we show how to use them to obtain a constant-factor approximation for relative congestion on a network where the minimum edge capacity can be arbitrarily low and does not depend on the demand values. This is the first result of this type that we are aware of. Consider next the *multiple-source unsplittable flow* problem also called *single-path multicommodity flow* (see [30] for recent results). In this problem there is a set of sources $s_1, \ldots, s_k$ and demands are associated with pairs (s_i, t_i). Again the i-th commodity must be routed along a single s_i-t_i path so that capacity constraints on the edges are satisfied. The multiple-source problem where all demands and capacities are equal to 1 is the well-known edge-disjoint paths problem. Assume there is an algorithm $\mathcal{A}$ for edge-disjoint paths achieving ρ times the fractional optimum for relative congestion. Then we show how to use $\mathcal{A}$ as a subroutine in the algorithmic framework we developed for the single-source case to obtain a 4ρ-approximation for the congestion metric for a multiple-source unsplittable flow problem with general demand and capacity values, i.e. demands in $(0, 1]$ and capacities in $[1, \infty)$. On the negative side we show that for the problem with two sources, no ρ-approximation with $\rho < 2$ can be achieved for relative congestion, unless P=NP, even when all the demands and capacities have value 1. This gives an idea of how hard it might be to formulate an approximate version of the fundamental Theorem 2.1 that we use in our single-source algorithms.

Applications to scheduling. We also examine applications of unsplittable flow to scheduling problems. A lower bound of 3/2 exists for the approximability of minimum makespan on parallel machines when a job j can be processed only on a subset $M(j)$ of the available machines [22]. Let $\mathcal{S}$ denote this scheduling problem. The 2-approximation algorithm of Lenstra, Shmoys and Tardos [22] or Shmoys and Tardos [28] for the more general problem of scheduling on unrelated machines is the best known for $\mathcal{S}$. On the other hand an approximation preserving reduction is known to exist from $\mathcal{S}$ to single-source unsplittable flow, so the 3/2 lower bound for unrelated machine scheduling applies to single-source unsplittable flow as well. The lower bound holds when processing times (or demands) take the values p or $2p$ for some $p > 0$. We give an additive p approximation for this restricted unsplittable flow problem which corresponds to a ratio of at most 3/2. This is the best possible approximation ratio for the problem unless P=NP. Improved approximation factors are also obtained for other restricted sets of demands.

2 Preliminaries

A *single-source unsplittable flow problem* (UFP) $(G, s, \mathcal{T})$ is defined on a capacitated network $G = (V, E, c)$ where a subset $\mathcal{T} = \{t_1, \ldots, t_k\}$ of V is the set of *sinks*; every edge e has a capacity c_e; each sink t_i has a demand ρ_i. A *source* vertex $s \in V$ is specified. For any $I \subseteq \mathcal{T}$, a *partial routing* is a set of s-t_i paths P_i, used to route ρ_i amount of flow from s to t_i for each $t_i \in I$. A partial routing that contains paths to all the sinks in $\mathcal{T}$ is called simply a *routing.* Given a routing g, the flow g_e through edge e is equal to $\sum_{P_i \in g, P_i \ni e} \rho_i$. A (partial) routing g for which $g_e \leq c_e$ for every edge e is a *(partial) unsplittable flow.* A *fractional* routing is one where the flow to each sink is split on potentially many paths; this is just standard max flow. We make two standard assumptions about an UFP:

C1. there is a feasible fractional flow solution.

C2. all demands are at most 1 and the minimum edge capacity is 1.

The first assumption is introduced only for simplicity. For each of the three optimization metrics, an optimal fractional solution can be found efficiently. Since our analysis uses the fractional optima as lower bounds, the approximation results hold without C1. For the second assumption we note that the actual numerical value of 1 is not crucial; demands and capacities can be scaled without affecting solutions. The important requirement is that the minimum capacity is greater than the maximum demand. We will explicitly state it when we deal with a UFP in which the minimum capacity can be arbitrarily small. Finally, the general *k-source unsplittable flow problem* (k-UFP) is similarly defined but now $k > 1$ sources exist and demands are defined for pairs (s_i, t_i) of sources and sinks. Unless otherwise stated, we use n, m to denote $|V|, |E|$ respectively.

The following theorem is an easy consequence of the well-known augmenting path algorithm ([5], [4]) and will be of use. It was also used as part of the approximation techniques in [18].

Theorem 2.1 *Let G be an arbitrary capacitated network with a source vertex s and a set $\mathcal{T}$ of sinks with demands. Let all demands in $\mathcal{T}$ be equal to a common value σ and all edge capacities be multiples of σ. If there is a fractional flow of value f, then there is an unsplittable flow of value at least f. This unsplittable flow can be found in time $O(km)$ where k is the number of sinks in $\mathcal{T}$.*

In the same way a routing corresponds to an edge flow, an edge flow can be represented as a path flow. Our algorithms will make use of the well-known *flow decomposition theorem* [2]. Given problem $(G, s, \mathcal{T})$ let a flow solution f be represented as an edge flow, i.e. an assignment of flow values to edges. Then one can find in $O(nm)$ time a representation of f as a path and cycle flow, where the paths join s to sinks in $\mathcal{T}$.

3 The approximation algorithm for minimum congestion

In this section we give a $(4 + o(1))$-approximation algorithm for relative congestion. Subsequently we show how to improve the approximation guarantee to $3.23+o(1)$. All our results can be seen to hold for minimizing *absolute congestion* as well, defined as the maximum over all edges e of the amount of flow crossing e. The same algorithm works on both directed and undirected graphs with the same performance guarantee. We use an initial max-flow solution to allocate capacity to different subproblems, where each subproblem contains demands in a fixed range. Capacities in the subproblems are arbitrary and may violate Condition C2. The subproblems are separately solved and then combined. Combining the partial solutions will have a *subadditive* effect on the total congestion and thus near-optimality is guaranteed.

We present first two simple lemmata that deal with cases where demands are bounded. They will be of use in the general algorithm.

Lemma 3.1 *For an* UFP *with demands in the interval $(0, \alpha]$ algorithm α-*ROUTING *finds in time $O(n+m)$ an unsplittable routing where the flow through each edge is at most $n\alpha$.*

Proof. A DFS from the source s gives a path from s to each of the at most n sinks. For each sink t_i route, on the corresponding path, flow equal to the demand $\rho_i \leq \alpha$. ∎

Given a real number interval with endpoints a and b, $a < b$, $a > 0$, the *ratio* $r(a, b)$ of the interval is b/a. The following fact is also used in [18]; we include a proof for the sake of completeness.

Lemma 3.2 *Given an* UFP $\Pi = (G, s, \mathcal{T})$*, with demands in the interval $(a, b]$ and arbitrary capacities, algorithm* INTERVAL_ROUTING *finds in polynomial time an unsplittable routing g where the flow g_e through an edge e is at most $r(a, b)c_e + b$.*

Proof sketch. Let f be a feasible fractional solution to Π. Round all demands up to b and call Π' the new unsplittable problem. To get a feasible fractional solution for Π' it suffices to multiply the flow f_e and the capacity c_e on each edge e by a factor of at most $r(a,b)$. Now further round all capacities up to the closest multiple of b by adding at most b. The new edge capacities are now at most $r(a,b)c_e + b$. By Theorem 2.1 an unsplittable routing for Π' can now be found in polynomial time. Algorithm INTERVAL_ROUTING finds first this unsplittable routing g' for Π'. To obtain an unsplittable routing g for Π, from every path in g' the flow in excess of the demands in $\mathcal{T}$ is removed. ∎

We present now Algorithm PARTITION. Subsequently we show how to choose the parameters $\xi, \alpha_1, \ldots, \alpha_\xi$ of the partitioning scheme. The two previous lemmata will be used to provide subroutines. Recall that in an UFP demands lie in the interval $(0,1]$ and capacities are at least 1.

ALGORITHM PARTITION$(G = (V,E,c), s, \mathcal{T}, \xi, \alpha_1, \ldots, \alpha_\xi)$

Step 1. Find a feasible fractional solution f.

Step 2. Define a partition of the $(0,1]$ interval into ξ consecutive subintervals $(0,\alpha_1], (\alpha_1,\alpha_2], \ldots, (\alpha_{\xi-1},\alpha_\xi]$, $\alpha_\xi = 1$. Construct ξ copies of G where the set of sinks in G_i, $1 \le i \le \xi$, is the subset $\mathcal{T}_i$ of $\mathcal{T}$ with demands in the interval $(\alpha_{i-1}, \alpha_i]$. Using flow decomposition determine for every edge e the amount c_e^i of c_e used by f to route flow to sinks in $\mathcal{T}_i$. Set the capacity of edge e in G_i to c_e^i.

Step 3. Invoke INTERVAL_ROUTING on each G_i, $2 \le i \le \xi$, to obtain an unsplittable routing g^i. Invoke α-ROUTING on G_1 to obtain an unsplittable routing g^1.

Step 4. Output routing g, the union of the path sets g^i, $1 \le i \le \xi$.

We need to examine the effect of the combination to the total congestion.

Lemma 3.3 *Algorithm* PARTITION *outputs an unsplittable routing g with relative congestion at most $n\alpha_1 + \max_{2 \le i \le \xi}\{r(\alpha_{i-1},\alpha_i)\} + \sum_{i=2}^{\xi}\alpha_i$. The running time of* PARTITION *is $O(T_1(n,m) + nm + m\xi)$ where $T_1(n,m)$ is the time to solve a fractional max-flow problem.*

Proof. In order to determine in Step 2 the capacity c_e^i of edge e in the i-th copy G_i, $1 \le i \le \xi$, of G we use the flow decomposition theorem [14]. Any flow along cycles given by the decomposition theorem is discarded since it does not contribute to the routing of demands. We focus on the congestion of a particular edge e. By Lemma 3.2 algorithm INTERVAL_ROUTING finds an unsplittable routing in G_i, $2 \le i \le \xi$, at Step 3, by pushing through edge e at most $r(\alpha_{i-1},\alpha_i)c_e^i + \alpha_i$ units of flow. By Lemma 3.1 algorithm α-ROUTING pushes at most $n\alpha_1$ units of flow through edge e to find a routing on G_1. At Step 4 the superposition of the routings g^i gives an approximate unsplittable solution g where g_e is at most

$$n\alpha_1 + \sum_{i=2}^{\xi} c_e^i r(\alpha_{i-1},\alpha_i) + \sum_{i=2}^{\xi}\alpha_i \quad \le$$

$$n\alpha_1 + \max_{2 \le i \le \xi}\{r(\alpha_{i-1},\alpha_i)\}\sum_{i=2}^{\xi} c_e^i + \sum_{i=2}^{\xi}\alpha_i \quad \le$$

$$n\alpha_1 + \max_{2 \le i \le \xi}\{r(\alpha_{i-1},\alpha_i)\}c_e + \sum_{i=2}^{\xi}\alpha_i c_e.$$

The last step follows because $c_e \ge 1$. Finding the fractional flow in step 1 requires $T_1(n,m)$ time. Flow decomposition can be done in $O(nm)$ time [2], thus the running time of step 2 is $O(nm + m\xi)$. Implementing INTERVAL_ROUTING with the augmenting path algorithm gives total running time $O(nm)$ for all the $\xi - 1$ invocations at Step 3. The reason is that by Theorem 2.1 the algorithm takes $O(m)$ time per sink and the total number of sinks in the ξ subproblems is at most n. Hence the total running time of PARTITION is $O(T_1(n,m) + nm + m\xi)$. ∎

In order to substantiate the approximation factor we need to come up with a partitioning scheme to be used at Step 2 of algorithm PARTITION. The following theorem is the main result of this section.

Theorem 3.1 *Given an* UFP Π, *Algorithm* PARTITION *finds a $(4+\varepsilon)$-approximation for relative congestion, for any $\varepsilon > 0$. The running time of the algorithm is $O(T_1(n,m) + nm + m\log(1/\varepsilon))$ where $T_1(n,m)$ is the time to solve a fractional max-flow problem.*

Proof. We show how to achieve $\varepsilon \le 1/n^c$, for some c. At step 2 of PARTITION partition the interval $(0,1]$ of demand values into ξ geometrically increasing subintervals

$$(0, 1/2^{\xi-1}], \ldots, (1/2^{i+1}, 1/2^i], \ldots, (1/2^2, 1/2], (1/2, 1]$$

such that $1/2^{\xi-1} \le 1/n^c, c > 1$. Thus ξ must be $\lceil c\log n + 1\rceil$. By Lemma 3.3, the relative congestion of the routing g returned at step 4 of the algorithm is at most $\frac{1}{n^{c-1}} + 2 + \sum_{i=0}^{\xi-2}\frac{1}{2^i} < 4 + \frac{1}{n^{c-1}}$. Obtaining a better $o(1)$ factor is straightforward by increasing the number of intervals. ∎

A fractional maximum flow can be found by the push-relabel method of Goldberg and Tarjan [8] whose currently fastest implementation has running time $T_1(n,m) = O(nm\log_{\frac{m}{n\log n}} n)$ [15]. In that case even when our algorithm is used to obtain a $4 + \frac{1}{2^n}$ approximation, the running time is dominated by a single max-flow computation. Alternatively, the new max-flow algorithm of Goldberg and Rao [10] may be used with $T_1(n,m) = O(\min(n^{2/3}, m^{1/2})m\log(\frac{n^2}{m})\log U)$, when edge capacities can be expressed as integers in the range $[1,\ldots,U]$. Finally for problem instances with a particular distribution of demands in $(0,1]$, a custom implementation of Lemma 3.3 can yield improved approximations. An example is when demands are bounded away from 0 or 1. We omit the details.

An improved approximation algorithm with the same running time as PARTITION can be obtained by combining the techniques used above with the idea from the proof of Theorem 10.1.

Theorem 3.2 *Given an* UFP II, *there is a polynomial-time algorithm that obtains a* $(3.23+o(1))$*-approximation for relative congestion.*

4 Minimizing congestion with arbitrary capacities

We examine here UFP in its full generality, when demands lie in $(0,1]$ and capacities in $(0,\infty)$. We assume in this version that the minimum demand is $\Omega(1/2^{n^k})$ for some constant k. In the subsequent sections we return to the original assumptions on the capacity values as stated in Section 2.

The only modification required in algorithm PARTITION is at Step 1. We compute a feasible fractional solution such that the flow of commodity i through edge e is set to 0 if $\rho_i > c_e$. These constraints can be easily enforced, for example, in a linear programming formulation. Notice that the resulting problem formulation is a valid relaxation of the original problem and the unsplittable optimum is not affected by the additional constraints. The analysis of the performance guarantee is largely the same. Let L_PARTITION be the modified algorithm as outlined above. We maintain the notation of PARTITION for the decomposition parameters.

Lemma 4.1 *Algorithm* L_PARTITION *runs in polynomial time and outputs an unsplittable routing* g *with total flow through edge* e *at most* $\max_{2\le i\le\nu}\{r(\alpha_{i-1},\alpha_i)\} + \sum_{i=2}^{\nu}\alpha_i$, *where* $c_e \in (\alpha_{\nu-1},\alpha_\nu]$, $\nu \le \xi$.

Proof sketch. Similar to that of Lemma 3.3. The additional observation is that because of the new constraints in the fractional relaxation, during Step 2 of the algorithm edge e is not included in any graph G_i with $i \ge \nu$. ▮

Theorem 4.1 *Given an* UFP II *with arbitrary capacities, Algorithm* L_PARTITION *finds a* 5.8285*-approximation for relative congestion. The running time of the algorithm is polynomial.*

Proof sketch. Let r be the ratio of the geometrically decreasing intervals. (In the proof of Theorem 3.1 $r = 2$.) Then by Lemma 4.1 the maximum flow through edge e is at most $rc_e + \frac{rc_e}{1-1/r}$ which yields a relative congestion of $\frac{2r^2-r}{r-1}$. Selecting $r = 1.707$ yields 5.8285. ▮

5 Minimum-cost unsplittable flow

In this section we examine the problem of finding a minimum-cost unsplittable flow for $(G, s, \mathcal{T})$ when a cost $\gamma_e \ge 0$ is associated with every edge e. The cost $\gamma(P)$ of a path P is $\sum_{e\in P}\gamma_e$. Our objective is to find an unsplittable flow solution that minimizes the total cost $\sum_{t_i\in\mathcal{T}}\rho_i\gamma(P_i)$. We show how to modify algorithm PARTITION by invoking min-cost flow instead of max-flow routines to simultaneously achieve a constant factor approximation for both cost and congestion. Since PARTITION works on both directed and undirected graphs the same holds for the new algorithm. We need the analogue of Theorem 2.1.

Theorem 5.1 *Let* G *be an arbitrary network with costs and capacities on the edges, a source vertex* s *and a set* $\mathcal{T}$ *of sinks with demands. If all demands in* $\mathcal{T}$ *are equal to a common value* σ *and all edge capacities are multiples of* σ, *then there is a maximum fractional flow of minimum cost that is an unsplittable flow. Moreover, this unsplittable flow can be found in polynomial time.*

This theorem is an easy consequence of the well known successive shortest path algorithm for min-cost flow (developed independently by [13], [12], [3]).

The lemmata for the analysis follow. They are extensions of the lemmata from the previous section; we omit the proofs in this extended abstract.

Lemma 5.1 *Given a min-cost* UFP *with demands in the interval* $(0,\alpha]$ *and a fractional solution* f *with cost* $\gamma(f)$, *algorithm* α-C_ROUTING *finds an unsplittable routing where the flow through an edge is at most* $n\alpha$ *and the cost is at most* $\gamma(f)$. *The running time of the algorithm is* $O(n\log n + m)$.

Lemma 5.2 *Let a min-cost* UFP II *have demands in the interval* $(a,b]$, *arbitrary capacities and a fractional solution* f *with cost* $\gamma(f)$. *Then algorithm* C_INTERVAL_ROUTING *finds in polynomial time an unsplittable routing where the flow through an edge* e *is at most* $r(a,b)c_e + b$ *and with cost at most* $r(a,b)\gamma(f)$.

Our algorithm C_PARTITION has the same steps as PARTITION. The differences of C_PARTITION from PARTITION lie only in Steps 1 and 3. At Step 1 of C_PARTITION a fractional min-cost solution f is found. At Step 3 we invoke C_INTERVAL_ROUTING instead of INTERVAL_ROUTING. C_INTERVAL_ROUTING is implemented with the successive shortest paths algorithm for min-cost flow. Finally algorithm α-C_ROUTING is used to compute a routing for graph G_1 of the decomposition. We keep the same notation for the partitioning scheme.

Lemma 5.3 *Algorithm* C_PARTITION *outputs an unsplittable routing* g *with relative congestion at most* $n\alpha_1 + \max_{2\le i\le\xi}\{r(\alpha_{i-1},\alpha_i)\} + \sum_{i=2}^{\xi}\alpha_i$ *and cost at most* $\max_{2\le i\le\xi}\{r(\alpha_{i-1},\alpha_i)\}\gamma(f)$, *where* $\gamma(f)$ *is the minimum cost of a fractional solution. The running time of* C_PARTITION *is* $O(T_2(n,m)+nm+m\xi)$ *where* $T_2(n,m)$ *is the time to solve a fractional min-cost flow problem.*

Using the same partitioning scheme as in Theorem 3.1 we obtain

Theorem 5.2 *Given a min-cost* UFP II *we can obtain a simultaneous* $(2, 4+\varepsilon)$ *approximation for cost and relative congestion for any* $\varepsilon > 0$. *The running time of the algorithm is* $O(T_2(n,m)+nm+m\log(1/\varepsilon))$ *where* $T_2(n,m)$ *is the time to solve a fractional min-cost flow problem.*

We observe that the approximation factor for the cost in the above algorithm is the ratio of the subintervals used in the partitioning scheme. By increasing the constant for congestion we can improve upon the cost approximation.

Theorem 5.3 *Given a min-cost* UFP Π *we can obtain a simultaneous* $(1+\delta, 2+\frac{\delta^2+1}{\delta}+\varepsilon)$ *approximation for cost and relative congestion for any* $\delta, \varepsilon > 0$. *The running time of the algorithm is* $O(T_2(n,m)+m\log(1/\varepsilon))$ *where* $T_2(n,m)$ *is the time to solve a fractional min-cost flow problem.*

Currently the best time bound for min-cost flow is $T_2(n,m) = O(\min\{nm\log(n^2/m)\log(nC), nm(\log\log U)\log(nC), (m\log n)(m+n\log n)\})$ [9, 1, 23]. Finally, employing the congestion algorithm of Theorem 3.2 we can obtain a $(1.68, 3.23)$ simultaneous approximation for cost and congestion. In that case the trade-off of Theorem 5.3 becomes $(1+\delta, 1+\delta+\frac{(1+\delta)^2}{(1+\delta)^2-1}+\varepsilon)$. We omit further details.

6 Maximizing the routable demand

In this section we turn to the problem of finding a routable subset of sinks with maximum total demand. In other words we are interested in maximizing the *value* of the unsplittable flow such that the following two constraints are met: the capacity constraints are to be observed and only flow routed to demands that are fully satisfied counts for the total value. Note that if the latter requirement was dropped, a c-approximation, $c > 1$, for relative congestion would imply a $1/c$-approximation for the maximum amount of flow that can be routed unsplittably.

Let $(G, s, \mathcal{T})$ be the UFP we want to solve. The approach we follow is quite similar to the one used for minimizing congestion. Consider splitting the interval $(0,1]$ of demands into two subintervals $(0,\alpha]$, and $(\alpha, 1]$. We will give separate constant factor approximation algorithms for each of the two resulting subproblems. Since at least one of the two subintervals contains at least half of the total demand in $\mathcal{T}$, we obtain a constant factor approximation for the entire problem. This was also the high-level approach used in [18]. In fact, in order to route the demands in the subinterval $(\alpha, 1]$ we will use a simple and intuitive algorithm developed by Kleinberg [18]. For the demands in $(0,\alpha]$, we show how a partitioning scheme similar to the one used for minimizing congestion can give a simple algorithm with a constant performance guarantee. Again G can be either directed or undirected. Let $\alpha^f(G, s, \mathcal{T})$ denote the maximum total demand routed by a fractional solution f. We will use $\alpha^f(G, s, \mathcal{T})$, with f a maximum fractional solution, as an upper bound to measure the quality of our approximation.

The following lemma, shown in [18], provides us with a routine for our algorithm. The philosophy of the proof is similar to the one of Lemma 3.2.

Lemma 6.1 *[18] Given an* UFP $\Pi = (G, s, \mathcal{T})$, *with demands in the interval* $(a, b]$, *there is an algorithm* K_ROUTING *that finds a partial unsplittable flow* g *of value at least* $\frac{1}{2}[r(a,b)]^{-1}\alpha^f(G,s,\mathcal{T})$. *The algorithm runs in polynomial time.*

Adapting the proof of Lemma 6.1 we can show a slightly different one that will be also of use in our algorithm. We include the proof for the sake of completeness.

Lemma 6.2 *Given an* UFP $\Pi = (G, s, \mathcal{T})$, *with demands in the interval* $(a, b]$, *and capacities multiples of* b, *there is an algorithm* M_ROUTING *that finds, in polynomial time, a partial unsplittable flow* g *of value at least* $[r(a,b)]^{-1}\alpha^f(G,s,\mathcal{T})$.

Proof. We outline how algorithm M_ROUTING works. Let f be a fractional solution to Π. Round down all demands to a. Multiply all capacities in G by $[r(a,b)]^{-1}$. Let $\Pi' = (G', s, \mathcal{T}')$, be the resulting problem. We show that Π' has a fractional solution f' of value $\alpha^{f'}(G', s, \mathcal{T}') \geq [r(a,b)]^{-1}\alpha^f(G,s,\mathcal{T})$: multiply the flow pushed by f on every edge by $[r(a,b)]^{-1}$ and let f' be the resulting fractional flow. Clearly f' does not route more than a units of flow to any sink in $\mathcal{T}'$ and respects the capacities in G' so it is feasible for Π'.

We now show how to obtain a partial unsplittable solution for Π of value $\alpha^{f'}(G', s, \mathcal{T}')$. In G all capacities are multiples of b so in G' they are multiples of a. But all demands in G' are a so by Theorem 2.1 there is an unsplittable flow g' of value at least $\alpha^{f'}(G', s, \mathcal{T}')$. Multiplying the flow on every path in g' by a factor of at most $r(a,b)$ gives a partial unsplittable flow g for Π, respecting the capacities in G and with value at least $\alpha^{f'}(G', s, \mathcal{T}')$. ∎

Our aim is to use a partitioning scheme as in the congestion case. However, in the latter problem, only a fraction of the capacity of an edge is assigned to each partition. On the other hand in order to make use of Lemma 6.2 on each subproblem, we require capacities to be multiples of the maximum demand. To circumvent this difficulty we scale down the fractional solution to each subproblem. On each edge a constant fraction of the edge capacity is then left unused and can provide the required extra capacity for the subproblems. The scaling of the fractional solution will incur a scaling by the same factor to the approximation ratio. We provide now the general algorithm. Subsequently we show how to choose the parameters $x, \xi, \alpha_1, \ldots, \alpha_\xi$ of the partitioning.

ALGORITHM

M_PARTITION($G = (V, E, c), s, \mathcal{T}, \xi, \alpha_1, \ldots, \alpha_\xi, x$)

Step 1. Find a feasible fractional solution f.

Step 2. Let x be a constant, $0 < x < 1$. Define a partition of the $(0,1]$ interval into ξ consecutive subintervals $(0,\alpha_1], (\alpha_1,\alpha_2], \ldots, (\alpha_{\xi-1},\alpha_\xi]$, $\alpha_\xi = 1$, such that $\sum_{i=1}^{\xi-1}\alpha_i \leq x$. Construct $\xi - 1$ copies of G where the set of sinks in G_i, $2 \leq i \leq \xi$, is the subset $\mathcal{T}_i$ of $\mathcal{T}$ with demands in the interval $(\alpha_{i-1}, \alpha_i]$. Using flow decomposition determine for every edge e the fraction c_e^i of c_e used by f to route flow to sinks in $\mathcal{T}_i$.

Step 3. Set the capacity of edge e in G_i, $2 \leq i \leq \xi - 1$, equal to the smallest multiple of α_i greater than or equal to $(1-x)c_e^i$. Set the capacity of edge e in G_ξ to c_e.

Step 4. Invoke M_ROUTING on each G_i, $2 \leq i \leq \xi - 1$, to obtain a partial unsplittable flow g^i. Invoke K_ROUTING on G_ξ to obtain a partial unsplittable flow g^ξ.

Step 5. Set partial routing g to be the union of the path sets g^i, $2 \leq i \leq \xi - 1$. Of the two partial routings g and g^ξ output the one of greater value.

By Lemma 6.1, partial routing g^ξ is a partial unsplittable flow that respects the capacities. We need to establish this for partial routing g as well and also get an estimate of its value. Note that algorithm M_PARTITION does not route any demands in $(0, \alpha_1]$. Let $a^f(G, s, \mathcal{T}_i)$ denote the flow routed by the fractional solution to demands in the interval $(\alpha_{i-1}, \alpha_i]$.

Lemma 6.3 *Partial routing g found by algorithm* M_PARTITION *is a partial unsplittable flow of value at least* $\min_{2 \leq i \leq \xi-1}\{[r(\alpha_{i-1}, \alpha_i)]^{-1}\}(1-x)\alpha^f(G, s, \bigcup_{2 \leq i \leq \xi-1} \mathcal{T}_i)$*. Moreover g respects the capacity constraints in G. The running time of* M_PARTITION *is $O(T_1(n,m) + nm + m\xi)$ where $T_1(n,m)$ is the time to solve a fractional max-flow problem.*

Proof sketch. We examine first the value of g. Let f_i^x be the fractional optimal flow on G_i. By Lemma 6.2 the value of the partial routing g_i, $2 \leq i \leq \xi - 1$, found at Step 5, is at least $[r(\alpha_{i-1}, \alpha_i)]^{-1} f_i^x$. By the capacity assignment in G_i, $\alpha^{f_i^x}(G, s, \mathcal{T}_i) \geq (1-x)\alpha^f(G, s, \mathcal{T}_i)$. Hence the value of the partial routing g^i is at least $(1-x)[r(\alpha_{i-1}, \alpha_i)]^{-1}\alpha^f(G, s, \mathcal{T}_i)$. Since $\alpha^f(G, s, \bigcup_{2 \leq i \leq \xi-1} \mathcal{T}_i) = \sum_{i=2}^{i=\xi-1} \alpha^f(G, s, \mathcal{T}_i)$ the claim on the value follows.

For the capacity constraints, we note that the aggregate capacity used in all partial routings g^i, $2 \leq i \leq \xi - 1$, on any edge e is by Step 4 at most $(1-x)\sum_{i=2}^{i=\xi-1} c_e^i + \sum_{i=2}^{i=\xi-1} \alpha_i$. By Step 2, $\sum_{i=2}^{\xi-1} \alpha_i \leq x$ and $\sum_{i=2}^{i=\xi-1} c_e^i \leq c_e$. Thus the aggregate capacity used by g is at most $(1-x)c_e + x \leq c_e$. The running time follows in the same manner as in Lemma 3.3. ∎

It remains to choose the parameters x, ξ and the α_i of the partitioning, and to account for the "missing" flow $a^f(G, s, \mathcal{T}_1)$.

Theorem 6.1 *Let x_1, x_2, x_3 be constants in $(0,1)$ such that $x_3 = x_1(1-x_2)$. Given an* UFP $\Pi = (G, s, \mathcal{T})$*, algorithm* M_PARTITION *finds a $(\beta - \varepsilon)$-approximation for maximum routable demand, for any $0 < \varepsilon < \beta$ and $\beta = \min\{(1-x_1)x_2/2, x_3/4\}$. The running time of the algorithm is $O(T_1(n,m) + nm + m\log(1/\varepsilon))$ where $T_1(n,m)$ is the time to solve a fractional max-flow problem.*

Proof sketch. We assume that there is a demand of value 1, otherwise we can rescale the demand values. We partition the interval $(0,1]$ of demand values into ξ geometrically increasing subintervals $(0, x_3(x_2)^{\xi-2}], \ldots, (x_3(x_2)^{i+1}, x_3(x_2)^i], \ldots, (x_3x_2, x_3]$, $(x_3, 1]$, such that $x_3(x_2)^{\xi-2} \leq 1/n^c, c > 1$. We then apply Lemma 6.3 and Lemma 6.1 and choose the one of greatest value. ∎

By choosing x_1, x_2, x_3 to be $4/10, 1/4$ and $3/10$ respectively we obtain $(1-x_1)x_2 = x_3/2 = 0.15$. Substituting the values in the formula for β we obtain $\beta = .075$.

7 Minimizing the number of rounds

In this section we sketch our approach to the problem of routing in rounds. A partitioning of the set of sinks into a minimum number of communication rounds is sought so that the set of terminals assigned to each round is routable under the capacity constraints.

We introduce notation $\zeta^f(G, s, \mathcal{T})$ for the minimum relative congestion of a *fractional* solution to an UFP $(G, s, \mathcal{T})$. Note that $\lceil \zeta^f(G, s, \mathcal{T}) \rceil$ is a lower bound on the minimum number of rounds. We employ a subroutine that we call KR_ROUTING (see Lemma 5.3 in [18]) and a variant R_ROUTING to deal with subproblems that have bounded demands. The subroutines are similar in spirit to the subroutines K_ROUTING and M_ROUTING used for the maximum demand metric however their basic ingredient is not Theorem 2.1. Max-flow integrality is not useful in the rounds setting and instead Lemma 5.2 from [18] is used. We omit the details of the subroutines.

Our algorithm R_PARTITION and its analysis are very similar to M_PARTITION. Steps 1 through 3 are exactly the same for both algorithms. In Step 4 routines R_ROUTING and KR_ROUTING are invoked on G_i, $2 \leq i \leq \xi - 1$, and G_ξ respectively. Algorithm R_ROUTING outputs for every G_i, $2 \leq i \leq \xi - 1$, a set of paths $\mathcal{P}_{ij}$ to be used on round j. Let j^* be the maximum number of rounds output from R_ROUTING for any G_i, $2 \leq i \leq \xi - 1$. During round j, $1 \leq j \leq j^*$ we route on all paths in $\bigcup_i \mathcal{P}_{ij}$. Subsequently we route demands in the first and the last subinterval of the partitioning in two separate sets of rounds. The following lemma accounts for the rounds needed to route all demands except for the ones in the first subinterval.

Lemma 7.1 *Algorithm* R_PARTITION *runs in polynomial time and routes the demands in $(\alpha_1, 1]$ in at most* $\max_{2 \leq i \leq \xi-1}\{\lceil \frac{1}{1-x} r(\alpha_{i-1}, \alpha_i)\zeta^f(G, s, \mathcal{T}_i) \rceil\} + \lceil 2r(a_{\xi-1}, 1)\zeta^f(G, s, \mathcal{T}_\xi) \rceil$ *rounds. The capacity constraints are respected in each round.*

By choosing a partitioning scheme as in Theorem 6.1, the total demand in $(0, \alpha_1]$ is small enough to be routed in one round.

Theorem 7.1 *Let x_1, x_2, x_3 be constants in $(0,1)$ so that $x_3 = x_1(1-x_2)$. Given an* UFP $\Pi = (G, s, \mathcal{T})$*, we can obtain in polynomial time a β-approximation for minimum number of rounds where $\beta = \lceil 1/(1-x_1)x_2 \rceil + \lceil 2/x_3 \rceil + 1$.*

By choosing x_1, x_2, x_3 to be $1/2, 1/2$ and $1/4$ respectively we obtain $\beta = 13$.

8 Multiple-source unsplittable flow

In this section we examine a multiple-source k-UFP. The optimization metric of interest is relative congestion. Let k-UFP(σ) denote a multiple-source unsplittable flow problem where all demands and capacities have the same value σ; this is the heavily studied edge-disjoint paths problem. We prove that within a constant factor, approximating k-UFP with arbitrary demands and capacities is no harder than approximating k-UFP(σ). The result assumes that the performance guarantee ρ for k-UFP(σ) is measured against the fractional optimum. This has been the standard practice in the literature for approximating multicommodity flow and more general packing problems (e.g. [25, 17, 24, 29]).

Theorem 8.1 *Let $\mathcal{A}$ be a polynomial-time algorithm that outputs a solution for k-UFP(σ) that is within ρ of the optimal relative congestion. Then there is a polynomial-time $(4\rho + o(1))$-approximation algorithm for relative congestion on an arbitrary k-UFP $(G, \mathcal{S}, \mathcal{T})$.*

Proof sketch. The proof consists of modifying algorithm PARTITION from Section 3 to use $\mathcal{A}$ as a subroutine. The multi-source analogue of Lemma 3.2 follows.

Lemma 8.1 *Given a k-UFP $\Pi = (G, \mathcal{S}, \mathcal{T})$, with demands in the interval $(a, b]$ and arbitrary capacities, algorithm* MULTI_ROUTING *finds in polynomial time an unsplittable routing g where the flow g_e through an edge e is at most $\rho(r(a,b)c_e + b)$.*

Proof. Let f be a feasible fractional solution to Π. Round all demands up to b and call Π' the new unsplittable problem. To get a feasible fractional solution for Π' it suffices to multiply the flow f_e and the capacity c_e on each edge e by a factor of at most $r(a, b)$. Now further round all capacities up to the closest multiple of b by adding at most b. The new edge capacities are now at most $r(a,b)c_e + b$. Π' is now a k-UFP(b) so $\mathcal{A}$ can be invoked on Π' to find an unsplittable routing g' in polynomial time. Algorithm MULTI_ROUTING finds first this unsplittable routing g' for Π'. To obtain an unsplittable routing g for Π, from every path in g' the flow in excess of the demands in $\mathcal{T}$ is removed. ▮

Modifying algorithm PARTITION i) to find a fractional concurrent flow at Step 1 and ii) to invoke MULTI_ROUTING instead of INTERVAL_ROUTING completes the proof of the theorem. ▮

9 A hardness result for unsplittable flow

In this section we consider the hardness of approximation for minimizing the relative congestion of an unsplittable flow on a network with 2 sources (2-UFP). We give a gap-preserving reduction from 3-D MATCHING to 2-UFP and show that it is NP-hard to achieve an approximation ratio better than 2. The reduction follows. We use ideas from a reduction in Theorem 5 in [22]. Lemma 9.1 and the ensuing theorem hold for both directed and undirected graphs. The proof for the undirected case is more complicated and we omit it.

Lemma 9.1 *Given an instance I of* 3-D MATCHING, *we can obtain in polynomial time an instance I' of* 2-UFP *such that:*

$$I \in \text{3-D MATCHING} \Rightarrow OPT(I') = 1,$$

$$I \notin \text{3-D MATCHING} \Rightarrow OPT(I') \geq 2.$$

Proof. Let the m triples in I be of the form (a_i, b_j, c_k) and n the size of a perfect 3-D matching. We show how to construct a directed network G for the corresponding instance I' of unsplittable flow. G contains two source vertices s_d and s_c. For each a_i occurring in t_i triples in I, G contains $t_i - 1$ vertices called the *dummies.* We denote as $\bar{d_{i\tau}}$, $1 \leq \tau \leq t_i - 1$, the dummy vertices corresponding to a_i. For each element b_j in I and c_k in I, we have respectively a vertex b_j and $\bar{c_k}$ in I'. Terminal pairs $(s_c, \bar{c_k})$, $1 \leq k \leq n$, and $(s_d, \bar{d_{i\tau}})$, $1 \leq i \leq n, 1 \leq \tau \leq t_i - 1$, have demand 1 each. All the edges in G have capacity 1.

For each of the m triples in I there will be a *triple gadget* in G. Let the μ-th triple be (a_i, b_j, c_k). The gadget for μ contains two dedicated vertices: x_μ and y_μ. The edges of the gadget are (x_μ, y_μ), (b_j, x_μ), $(y_\mu, \bar{c_k})$ and an edge from y_μ to each $\bar{d_{i\tau}}$, $1 \leq \tau \leq t_i - 1$. This completes the description of a gadget. Finally, an edge is directed from s_d to each job x_μ, $1 \leq \mu \leq m$, and from s_c to each b_j, $1 \leq j \leq n$.

We say that vertex v *covers* vertex w if the unique flow path satisfying the demand of w goes through v. Similarly we say that a path covers w or if it is clear from context to which gadget a path belongs that a gadget μ covers w. Note that every unit of flow towards a $\bar{c_k}$ or a dummy vertex must go through a gadget; on the other hand no more than one unit of flow can go through a gadget if capacities are to be respected. Also by the network construction a path from s_c to a $\bar{c_k}$ must go through a b-vertex.

If I contains a 3-D matching, a straightforward unsplittable flow solution satisfies all the demands in G with maximum congestion 1. If I contains no matching then we show that no unsplittable flow exists respecting all capacities. Assume this not the case. The unsplittable flow must select a set Γ of n gadgets not to be used for covering dummy jobs but to cover $\bar{c_k}$, $1 \leq k \leq n$. By the definition of the dummy jobs the n gadgets in Γ correspond to triples covering the n distinct a_i, $1 \leq i \leq n$. Since no more than one unit of flow can go through a b-vertex the covering relation in Γ defines a 2-D perfect matching between the b's and the c's. Taking into account the a_i corresponding to each gadget in Γ we obtain a 3-D matching, a contradiction. Thus when no 3-D matching exists, at least two units of flow must be routed via some gadget so the congestion is at least 2. ▮

Theorem 9.1 *No ρ-approximation algorithm, $\rho < 2$, exists for* 2-UFP *unless $P = NP$. The result holds even when all capacities and demands are equal to* 1.

10 Restricted sets of demands and applications to scheduling

In this section we examine connections between unsplittable flow and scheduling problems. Consider the scheduling problem $\mathcal{S}$ defined as follows. A set J of jobs is to be scheduled on a set M of parallel machines. A job j can be scheduled to run with processing time p_j on a set of machines $M(j)$ and has processing time ∞ on all machines in $M - M(j)$. In other words it is technologically infeasible for job j to run on machines in $M - M(j)$. The objective is to find a schedule that minimizes the *makespan,* i.e. the maximum completion time of a job. $\mathcal{S}$ is a special case of minimizing makespan on unrelated machines. However, the best approximation algorithm known for $\mathcal{S}$ is the 2-approximation for unrelated machine scheduling [22, 28]. It is known that unless P=NP, no approximation better than 3/2 exists for $\mathcal{S}$ even with processing times from the set $\{1, 2, \infty\}$ [22]. An approximation-preserving reduction exists from $\mathcal{S}$ to minimum congestion unsplittable flow on a three-level directed graph G [18].

We consider a UFP on an arbitrary network with demands p and $2p$, for some $p > 0$. We show how to find an unsplittable routing with flow exceeding the capacity of an edge by at most p.

Theorem 10.1 *Given an* UFP $\Pi = (G, s, \mathcal{T})$, *with demands from the set* $\{p, 2p\}$, $p > 0$, *there is an algorithm to find in polynomial time an unsplittable routing where the flow through an edge* e *is at most* $c_e + p$. *Thus the approximation ratio for the relative congestion is at most* $3/2$. *This is best possible, unless P=NP.*

Proof. Assume that Π has an unsplittable routing with relative congestion 1. Let $\mathcal{T}_1$, $\mathcal{T}_2$ denote the set of sinks in G with demands p and $2p$ respectively. Form $\bar{\mathcal{T}}_2$ by putting two copies, each with demand p, of every sink in $\mathcal{T}_2$. Call the $(G, s, \mathcal{T}_1 \cup \bar{\mathcal{T}}_2)$, Π'. Since an unsplittable flow solution exists for Π, one exists for Π' as well. Rounding down the capacities of edges to the closest multiple of p does not affect the existence of this solution. But in Π' all demands are p and all capacities are multiples of p therefore by Theorem 2.1 we can find an unsplittable flow solution g' in polynomial time.

Create a graph G_2 where edge e from G is present only if it is used by g' to route $x_e > 0$ units of flow to sinks in $\bar{\mathcal{T}}_2$. Note that x_e is a multiple of p. Assign capacities x_e to the edges in G_2. G_2 can be constructed using flow decomposition on g'. For the unsplittable flow problem $(G_2, s, \mathcal{T}_2)$, we have a fractional solution, namely x. All the demands are equal to $2p$ and all capacities are multiples of p. By adding p to each capacity we can obtain an unsplittable routing for sinks in $\mathcal{T}_2$. Combining with the unsplittable routing given by g' for sinks in $\mathcal{T}_1$ we obtain an unsplittable routing for Π where the flow through an edge is at most $c_e + p$. Recall that c_e is the capacity of edge e in G and is at least $2p$. Thus the approximation ratio achieved for relative congestion in G is at most 3/2. The lower bound follows from [22].

In case the relative congestion of the unsplittable routing is not 1 one can use the algorithm we propose as a 3/2-relaxed decision procedure in conjunction with a binary search for the optimum congestion to obtain the claimed approximation. See [11] for details on relaxed decision procedures. ∎

Corollary 10.1 *For problem* $\mathcal{S}$ *with processing times from* $\{p, 2p, \infty\}$ *there is a polynomial time algorithm that comes within an additive* p *of the optimum makespan. The approximation ratio is at most* $3/2$ *and this is best possible, unless P=NP.*

We can generalize the results of Theorem 10.1 to show that if the demands are in the set $\{p, Cp\}$ then we can approximate relative congestion to within a $2 - \frac{1}{C}$ factor. We can also show that if the demands lie in the interval $[p, Cp]$ then we can achieve a C-approximation for relative congestion. This improves on the $(C + 1)$-approximation given by Lemma 3.2.

Acknowledgement. Thanks to Aravind Srinivasan for useful discussions.

References

[1] R. K. Ahuja, A. V. Goldberg, J. B. Orlin, and R. E. Tarjan. Finding Minimum-Cost Flows by Double Scaling. *Mathematical Programming,* 53:243–266, 1992.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: Theory, Algorithms and Applications.* Prentice Hall, 1993.

[3] R. G. Busaker and P. J. Gowen. A procedure for determining minima-cost flow network patterns. Technical Report ORO-15, Operational Research Office, Johns Hopkins University, Baltimore, MD, 1961.

[4] P. Elias, A. Feinstein, and C. E. Shannon. Note on maximum flow through a network. *IRE Transactions on Information Theory IT-2,* pages 117–199, 1956.

[5] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canad. J. Math.,* 8:399–404, 1956.

[6] A. Frank. Packing paths, cuts and circuits - a survey. In B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, editors, *Paths, Flows and VLSI-Layout,* pages 49–100. Springer-Verlag, Berlin, 1990.

[7] N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica,* 18:3–20, 1997.

[8] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM,* 35:921–940, 1988.

[9] A. V. Goldberg and R. E. Tarjan. Solving minimum-cost flow problems by successive approximation. *Mathematics of Operations Research*, 15(3):430–466, 1990.

[10] A.V. Golderg and S. Rao. Beyond flow decomposition barrier, 1997. These proceedings.

[11] D.S. Hochbaum and D.B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34:144–162, 1987.

[12] M. Iri. A new method of solving transportation-network problems. *Journal of the Operations Research Society of Japan*, 3:27–87, 1960.

[13] W. S. Jewell. Optimal flow through networks. Technical Report 8, Operations Research Center, MIT, Cambridge, MA, 1958.

[14] L. R. Ford Jr. and D. R. Fulkerson. *Flows in networks*. Princeton University Press, 1962.

[15] V. King, S. Rao, and R. Tarjan. A faster deterministic maximum flow algorithm. In *Proceedings of the 3rd ACM-SIAM Symposium on Discrete Algorithms*, pages 157–164, January 1992.

[16] P. Klein, A. Agrawal, R. Ravi, and S. Rao. Approximation through multicommodity flow. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 726–737, 1990.

[17] P. Klein, S. A. Plotkin, C. Stein, and É. Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM Journal on Computing*, 23(3):466–487, June 1994.

[18] J. M. Kleinberg. Single-source unsplittable flow. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 68–77, October 1996.

[19] J. M. Kleinberg and É. Tardos. Approximations for the disjoint paths problem in high-diameter planar networks. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 26–35, 1995.

[20] J. M. Kleinberg and É. Tardos. Disjoint paths in densely-embedded graphs. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 52–61, 1995.

[21] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 422–431, 1988.

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

[23] J.B. Orlin. A faster strongly polynomial minimum cost flow algorithm. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 377–387, 1988.

[24] S. Plotkin, D. B. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20:257–301, 1995.

[25] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.

[26] N. Robertson and P. D. Seymour. Outline of a disjoint paths algorithm. In B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, editors, *Paths, Flows and VLSI-Layout*. Springer-Verlag, Berlin, 1990.

[27] A. Schrijver. Homotopic routing methods. In B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, editors, *Paths, Flows and VLSI-Layout*. Springer-Verlag, Berlin, 1990.

[28] D. B. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming A*, 62:461–474, 1993.

[29] A. Srinivasan. An extension of the Lovász Local Lemma and its applications to integer programming. In *Proceedings of the 7th ACM-SIAM Symposium on Discrete Algorithms*, pages 6–15, 1996.

[30] A. Srinivasan. Improved approximations for edge-disjoint paths, unsplittable flow and related routing problems, 1997. These proceedings.