Approximation Algorithms for Single-Source Unsplittable Flow*

Stavros G. Kolliopoulos[1] Clifford Stein[2]

Abstract

In the *single-source unsplittable flow* problem, we are given a network G, a source vertex s and k commodities with sinks t_i and real-valued demands ρ_i, $1 \leq i \leq k$. We seek to route the demand ρ_i of each commodity i along a single s-t_i flow path, so that the total flow routed across any edge e is bounded by the edge capacity u_e. The conceptual difficulty of this NP-hard problem arises from combining packing constraints due to the existence of capacities with path selection in a graph of arbitrary topology. In this paper we give a generic framework, which yields approximation algorithms that are simpler than those previously known and achieve significant improvements upon the approximation ratios. Our framework, with appropriate subroutines, applies to all optimization versions previously considered and, unlike previous work, treats in a unified manner directed and undirected graphs. We provide extensions of our algorithms which yield the best possible approximation guarantees for restricted sets of demand values and an associated scheduling problem.

Keywords: approximation algorithms, network algorithms, routing, network flow, unsplittable flow, disjoint paths, parallel machine scheduling. *AMS subject classifications:* 68Q25, 90B10, 90B12, 90B35.

Running head: Unsplittable Flow.

*A preliminary version of this paper appeared in the Proceedings of the 38th IEEE Symposium on Foundations of Computer Science, pp. 426–435, 1997.

[1]Department of Computing and Software, Faculty of Engineering, McMaster University, Hamilton, Ontario, L8S 4L7, Canada. Email: `stavros@mcmaster.ca`. Work performed while at the Department of Computer Science, Dartmouth College and partially supported by NSF Award CCR-9308701 and NSF Career Award CCR-9624828.

[2]Dartmouth College, Department of Computer Science, Hanover, NH 03755-3510. E-mail: `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 (UFP), we are given a network $G = (V, E, u)$, a source vertex s and a set of k commodities with sinks $t_1, \ldots, t_k$ and associated real-valued demands $\rho_1, \ldots, \rho_k$. We seek to route the demand ρ_i of each commodity i, along a single s-t_i flow path, so that the total flow routed across any edge e is bounded by the edge capacity u_e. See Figure 1 for an example instance. The minimum edge capacity is assumed to have value at least $\max_i \rho_i$. The requirement of routing each commodity on a single path bears resemblance to integral multicommodity flow and in particular to the multiple-source edge-disjoint path problem. For the latter problem, a large amount of work exists either for solving exactly interesting special cases (e.g. [9], [36], [35]) or for approximating, with limited success, various objective functions (e.g. [34],[11], [24], [25],[22]). Further, shedding light on single-source unsplittable flow may lead to a better understanding of multiple-source disjoint-path problems. From a different perspective, UFP can be approached as a variant of standard, single-commodity, maximum flow since in both problems there is one source for the flow. From this point of view, UFP generalizes single-source edge-disjoint paths. UFP is also important as a unifying framework for a variety of scheduling and load balancing problems [23]. It can be used to model bin-packing [23] and certain scheduling problems on parallel machines [31]. Another possible application for UFP is in virtual-circuit routing, where the source would represent a node wishing to multicast data to selected sinks. The conceptual difficulty in UFP arises from combining packing constraints due to the existence of capacities with path selection in a graph of arbitrary topology.

The feasibility question for UFP, i.e., can all commodities be routed unsplittably, is strongly NP-complete [22]. We focus thus on efficient algorithms to obtain approximate solutions. For a minimization (resp. maximization) problem, a *ρ-approximation algorithm,* $\rho > 1$, ($\rho < 1$), outputs in polynomial time a solution with objective value at most (at least) ρ times the optimum. Three main optimization versions of unsplittable flow can be defined.

minimum congestion. Find an unsplittable flow f satisfying all demands, which minimizes *congestion,* i.e. the quantity $\max\{\max_e\{f_e/u_e\}, 1\}$. That is, the minimum congestion is the smallest number $\alpha \geq 1$, such that if we multiplied all capacities by α, f would satisfy the capacity constraints. The congestion metric was a primary testbed for the randomized rounding technique of Raghavan and Thompson [34] and has been studied extensively for its connections to multicommodity flow and cuts (e.g. [30, 21, 29, 20]). [1]

maximum routable demand. Route unsplittably a subset of the commodities of maximum total demand, while respecting capacity constraints.

[1]In [29] congestion is defined as $\max_e\{f_e/u_e\}$. In this setting a congestion of $\lambda \in (0, 1)$, implies that it is possible to multiply all demands by $1/\lambda$ and still satisfy them by a flow which respects the capacity constraints. Algorithmically the two definitions are equivalent: with a polynomial number of invocations of an algorithm which minimizes congestion according to our definition, one can minimize congestion according to the definition in [29].

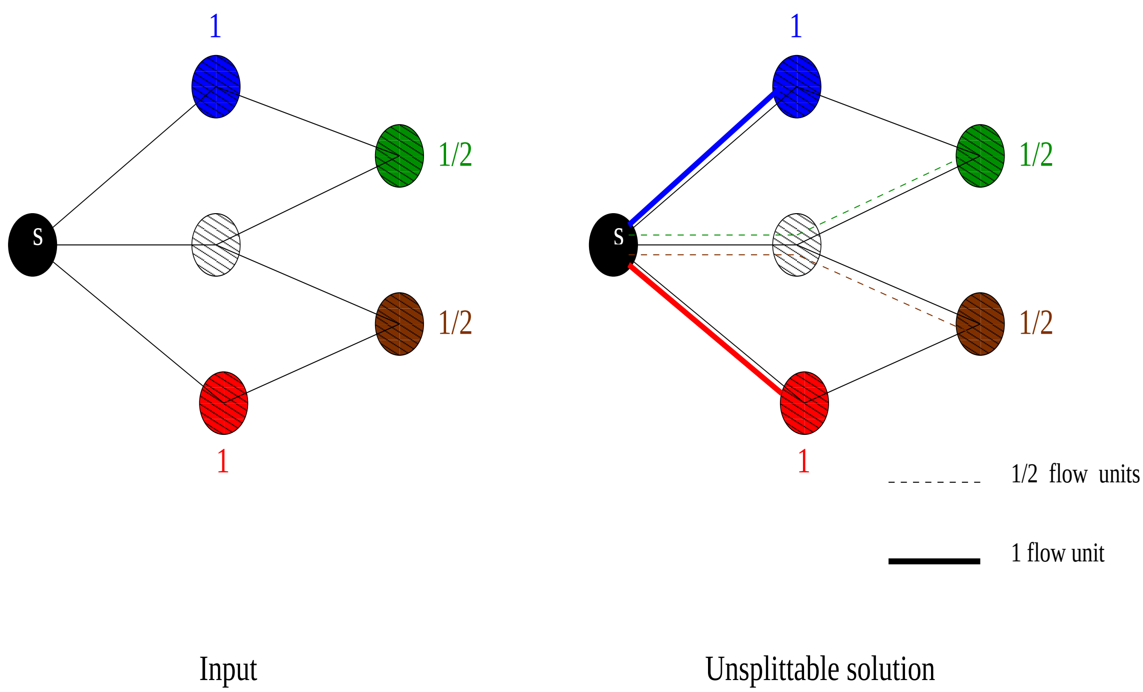


Figure 1: Example UFP instance. Numbers on the vertices denote demands; all edge capacities are equal to 1.

minimum number of rounds. Partition the commodities into a minimum number of sets, called *rounds,* so that commodities assigned to the same round can be routed unsplittably without violating the capacity constraints.

The randomized rounding technique applies to the more general multiple-source unsplittable flow problem and provides an $O(\log|E|/\log\log|E|)$ approximation for congestion; for the maximum demand and minimum number of rounds versions, it can give meaningful approximations only when the minimum capacity value is $\Omega(\log|E|)$ times the value of the maximum demand [34, 33]. Kleinberg [22, 23] was the first to consider UFP as a distinct problem and gave constant-factor approximation algorithms for all three optimization versions presented above, on both directed and undirected graphs.

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 [23].
- Our approach treats in a unified manner directed and undirected graphs.
- We obtain significant improvements upon the approximation ratios for all three optimization versions.

Our algorithms follow the same *grouping-and-scaling* technique. We first find a maximum flow. This relaxation is a "splittable" solution, i.e., it allows the demand for a commodity to be split along more than one path. As in the previous work [23] our work uses the maximum flow solution as a guide for the discovery of an unsplittable routing. However we use this solution in a much stronger way: we employ information from maximum flow to *allocate capacity* to a set of subproblems, where each subproblem G^i contains commodities with demands in a fixed range. For each subproblem G^i, a near-optimal routing g^i is separately computed by exploiting the fact that demands are close in range. These solutions are then combined by superimposing the g^i's on the original graph. Since we are interested in constant-factor approximations, when combining the solutions it is important to avoid an additive deterioration of the approximation guarantees obtained for each individual G^i. One of our conceptual contributions lies in showing that this is possible, given an appropriate initial capacity allocation. Combining solutions presents different challenges for the maximum demand and minimum number of rounds versions of our algorithms as opposed to minimizing congestion. In the congestion version, increasing the original edge capacities as the routings g^i are superimposed on G only affects the approximation ratio. For the other two versions, it would be infeasible to do so. We note that a grouping scheme on the commodities is also used in [23], though in a more complicated way; ours is based solely on demand values and does not require any topological information from the graph. We now elaborate on the approximation ratios we obtain.

Minimum congestion. We give a 3-approximation algorithm for both directed and undirected graphs. The bounds known before our work were 16 for the directed and 8.25 for the undirected case [23]. In the first publication of this work [27] we gave PARTITION, a 4-approximation algorithm and obtained also a $10/3 = 3.33\ldots$ bound[2] for a more elaborate version of PARTITION. Dinitz et al. recently obtained a new 2-approximation result (appearing in [5]). Following this development we further refined our 10/3-approximation algorithm and obtained H_PARTITION, a 3-approximation algorithm appearing also in [26]. In this paper we describe both PARTITION and H_PARTITION. Algorithm H_PARTITION also exploits the grouping and scaling technique by scaling demands in a fixed range to the same value. However instead of producing "in parallel" an unsplittable solution for all the ranges, H_PARTITION outputs the partial routings g^i by iteratively improving the quality of the solution to a maximum flow relaxation. The quality of the latter solution is characterized by the minimum amount of flow routed along a single path to any commodity which has not been routed yet unsplittably.

Minimizing congestion and cost. A natural optimization question arises when routing a unit of flow through edge e incurs cost c_e. The objective is to find an unsplittable flow that minimizes the total cost. Since UFP without costs is already NP-hard, the best we can hope for is a bicriteria approximation of cost and congestion. If z is the optimal congestion, define the optimal cost to be the minimum cost of an unsplittable flow that is feasible with respect to capacities zu. Our approach gives the first constant-factor approximation for minimum-cost, minimum-congestion 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 [23] give a simultaneous $(7.473, 10.473)$ approximation for cost and congestion. We provide a $(2, 3)$ 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, i.e., 7.5% of the optimum. In [23] the constant is not given explicitly but it can be as low as .031 for undirected graphs and on 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 [23]. A guarantee of routing in x rounds implies that at least $1/x$ of the total demand is routed in some round. Kleinberg's 32-approximation algorithm for rounds can be seen to imply a $1/32 = 0.03125$ approximation for routable demand when the Cut Condition is met, i.e. when the maximum flow relaxation satisfies all demands. Accordingly we obtain a $1/13 = .0769$ approximation for routable demand under the Cut Condition. We elaborate on this observation in Section 8.

We emphasize that our 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.

[2] erroneously reported in [27] as 3.23.

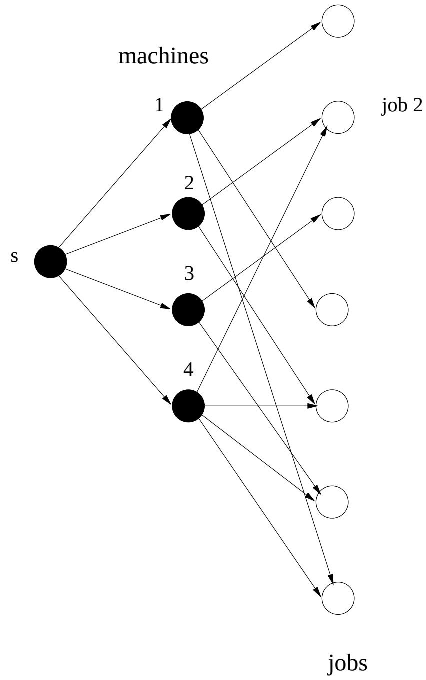


Figure 2: Reduction of the scheduling problem to UFP . Job 2 can be processed only on machines 2 and 4. The capacity of the edges connecting s to the machine vertices is equal to a makespan estimate.

The dominant computational steps are maximum flow and flow decomposition; these are tools that work well on both directed and undirected graphs. In [23] 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. We show how to use them to obtain a constant-factor approximation for minimum congestion on a network, where the minimum edge capacity can be arbitrarily low with respect to the maximum demand. This is the first result of this type that we are aware of. In subsequent work [28], we show how to apply the algorithmic techniques developed here for UFP to obtain improved approximations for classes of packing integer programs and multiple-source unsplittable flow. On the negative side, we show in this paper, that for the unsplittable flow problem with two sources on a directed network, no ρ-approximation with $\rho < 2$ can be achieved for 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 [31]. Let $\mathcal{S}$ denote this scheduling problem. The 2-approximation algorithm of Lenstra, Shmoys and Tardos [31] or Shmoys and Tardos [37] 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 (see Fig. 2) [23], so the 3/2 lower bound for $\mathcal{S}$ [31] 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 provide 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 the more general case, in which all demand values are powers of 1/2.

The outline of this paper is as follows. In Section 2 we present definitions and a basic theorem from flow theory. In Section 3 we provide a $(4+o(1))$-approximation algorithm for congestion, which introduces our basic techniques. In Section 4 we provide the improved 3-approximation. In Section 5 we give the approximation algorithm for minimum congestion unsplittable flow with arbitrarily small capacities. In Section 6 we present the simultaneous cost-congestion approximation. In Sections 7 and 8 the approximation algorithms for maximum total demand and minimum number of rounds are presented. In Section 9 we present the hardness result for 2-source unsplittable flow. Finally, in Section 10 we examine the connection between UFP and scheduling problems. A preliminary version of the material in this paper appeared in [27]. After the first publication of this work, Dinitz, Garg and Goemans obtained improved approximation ratios by using a different approach [5].

2 Preliminaries

All logarithms in this paper are base 2 unless otherwise noted. A *single-source unsplittable flow problem* (UFP) $(G, s, \mathcal{T})$ is defined on a capacitated network $G = (V, E, u)$ where a subset $\mathcal{T} = \{t_1, \ldots, t_k\}$ of V is the set of *sinks*; every edge e has a real-valued capacity u_e; each sink t_i has a real-valued demand ρ_i. The pair (t_i, ρ_i) defines the *commodity* i. To avoid cumbersome notation, we use occasionally the set $\mathcal{T}$ to denote the set of commodities. We assume in our definition that each vertex in V hosts at most one sink; the input network can always be transformed to fulfill this requirement. A *source* vertex $s \in V$ is specified. For any $I \subseteq \mathcal{T}$, a *partial routing* is a set of s-t_{i_j} paths P_j, $1 \leq j \leq |I|$ used to route ρ_{i_j} amount of flow from s to t_{i_j} for each $t_{i_j} \in I$. A partial routing that contains a path to each sink in $\mathcal{T}$ is simply called 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 $g_e \leq u_e$ for every edge e, is a *(partial) unsplittable flow.* A *fractional routing* is a set of flow paths, satisfying capacity constraints, in which the flow to each sink is split along potentially many paths. A fractional routing can be computed by standard maximum flow. Thus we shall refer to a fractional routing satisfying all demands of the underlying UFP as a *feasible fractional flow solution.* For clarity we will also refer to a standard maximum flow as *fractional maximum flow.* We assume that a UFP input meets the following two requirements:

Cut Condition. There is a feasible fractional flow solution.

Balance Condition. All demands are at most 1 and the minimum edge capacity is 1.

The first requirement is introduced only for simplicity. For each of the optimization metrics s-tudied, an optimal fractional solution, potentially violating the unsplittability requirement, can be efficiently found. Since our analysis uses the fractional optima as lower bounds, our approximation results hold without the Cut Condition. For the Balance Condition, 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 or equal to the maximum demand. We will explicitly state it when we deal with a UFP in which the minimum capacity can be arbitrarily small. Finally, *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 ([7], [6]) and will be of use. It was also used as part of the approximation techniques in [23].

Theorem 2.1 *Let $(G = (V, E, u), s, \mathcal{T})$ be a* UFP *on an arbitrary network G with all demands ρ_i, $1 \leq i \leq k$, equal to the same value σ, and edge capacities u_e equal to integral multiples of σ for all $e \in E$. 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 $\mathcal{T}$.*

The above theorem is a consequence of the integrality property for maximum flow with all demands and capacities scaled by σ.

We defined a routing as a collection of flow paths; it can equivalently be represented as an *edge flow,* i.e., a collection of functions assigning flow values to every edge. Conversely, an edge flow whether fractional or unsplittable can be represented as a path flow. In particular our algorithms will make use of the well-known *flow decomposition theorem* [2]. Given problem $(G, s, \mathcal{T})$ let a fractional flow solution f be represented as 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 $\mathcal{T}$. In the remainder of the paper, we will assume that the cycles are deleted from the output of a flow decomposition. The flow decomposition theorem trivially applies also to transforming an unsplittable flow from edge representation to a path representation. We will point out cases in which outputting the unsplittable flow as an edge flow results in faster algorithms.

3 The approximation algorithm for minimum congestion

In this section we give a $(4 + o(1))$-approximation algorithm for minimizing congestion. In Section 4 we show how to improve the approximation guarantee to 3. All our results can be seen to hold for minimizing *absolute congestion* as well, defined as $\max\{\max_e\{f_e\}, 1\}$. The same algorithm works on both directed and undirected graphs with the same performance guarantee. The algorithm works by partitioning the original UFP into subproblems, each containing only demands in a fixed range. We use an initial maximum-flow solution to allocate capacity to different subproblems. Capacities in the subproblems are arbitrary and may violate the Balance Condition. The subproblems are separately solved by introducing bounded congestion; the partial solutions are then combined by superimposing on the original network G the flow paths obtained in the respective routings. The combination step will have a subadditive effect on the total congestion and thus near-optimality of the global solution is achieved.

We present first two simple lemmata for cases in which demands are bounded. They will be used to provide subroutines to the final algorithm.

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

Proof. Algorithm α-ROUTING works as follows. A Depth First Search from the source s gives a path from s to each of the at most n sinks. For each sink t_i, route, along the corresponding path, flow equal to the demand $\rho_i \leq \alpha$. ■

Given a real number interval with endpoints a and b, $0 < a < b$, the *ratio* $r(a, b)$ of the interval is b/a. The following fact is also used in [23]; we include a proof for the sake of completeness.

Lemma 3.2 *[23] Given an* UFP $\Pi = (G, s, \mathcal{T})$, *with demands in the interval* $(a, b]$ *and arbitrary capacities, there is an algorithm,* INTERVAL_ROUTING, *which finds in polynomial time an unsplittable routing* g, *such that for all edges* e, $g_e \leq r(a, b)u_e + b$.

Remark: The statement of the lemma holds also when demands lie in the interval $[a, b]$.

Proof. Let f be a feasible fractional solution to Π. Round all demands up to b and call Π' the resulting UFP instance. To obtain a feasible fractional solution for Π', it suffices to multiply the flow f_e and the capacity u_e on each edge e by a factor of at most $r(a, b)$. Further round all capacities up to the closest multiple of b by adding at most b. The new edge capacities are now at most $r(a, b)u_e + b$. 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 are now ready to present Algorithm PARTITION in Fig. 3. Algorithm PARTITION takes as input a UFP $(G, s, \mathcal{T})$, together with a set of parameters $\xi, \alpha_1, \ldots, \alpha_\xi$ that will determine the exact

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

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, $1 \leq i \leq \xi$, 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 amount u_e^i of u_e used by f to route flow to sinks in $\mathcal{T}_i$. Set the capacity of edge e in G_i to u_e^i.

Step 3. Invoke INTERVAL_ROUTING on each G_i, $2 \leq i \leq \xi$, 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, $1 \leq i \leq \xi$.

Figure 3: Algorithm PARTITION.

partitioning scheme. Subsequently we show how to choose these parameters so as to optimize the approximation ratio. The algorithm outputs an unsplittable routing g. 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.

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

Lemma 3.3 *Algorithm* PARTITION *outputs an unsplittable routing g with congestion at most*

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

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 maximum flow problem.*

Proof. In order to determine in Step 2 the capacity u_e^i of edge e in the i-th copy G_i, $1 \leq i \leq \xi$, of G we use the flow decomposition theorem [8]. 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, $2 \leq i \leq \xi$, at Step 3, by pushing through edge e at most $r(\alpha_{i-1}, \alpha_i)u_e^i + \alpha_i$ 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

$$\begin{aligned} n\alpha_1 + \sum_{i=2}^{\xi} u_e^i r(\alpha_{i-1}, \alpha_i) + \sum_{i=2}^{\xi} \alpha_i &\leq \\ n\alpha_1 + \max_{2 \leq i \leq \xi}\{r(\alpha_{i-1}, \alpha_i)\} \sum_{i=2}^{\xi} u_e^i + \sum_{i=2}^{\xi} \alpha_i &\leq \\ n\alpha_1 u_e + \max_{2 \leq i \leq \xi}\{r(\alpha_{i-1}, \alpha_i)\} u_e + \sum_{i=2}^{\xi} \alpha_i u_e. & \end{aligned}$$

The last step follows because $\sum_{i=2}^{\xi} u_e^i \leq u_e$ from the feasibility of f, and $u_e \geq 1$.

As for the running time, 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 $O(km)$ for all the $\xi - 1$ invocations at Step 3. This time includes the time required by flow decomposition to output each partial routing in a path representation. The reason is that by Theorem 2.1 the algorithm takes $O(m)$ time per sink. The total number of sinks in the ξ subproblems is $k \leq n$. 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* $O(T_1(n,m) + nm + m\log(n/\varepsilon))$ *where* $T_1(n,m)$ *is the time to solve a fractional maximum flow problem.*

Proof. 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 $1/2^{\xi-1} \leq \varepsilon/n$. Thus it suffices for ξ to be $\Theta(\log(n/\varepsilon))$. By Lemma 3.3, the congestion of the routing g returned at step 4 of the algorithm is at most $n\frac{\varepsilon}{n} + 2 + \sum_{i=0}^{\xi-2} \frac{1}{2^i} < 4 + \varepsilon$. ■

In order to achieve $\varepsilon = o(1)$, it suffices to set $1/2^{\xi-1} \leq 1/n^c$, $c > 1$. 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 [12] whose currently fastest implementation has running time $T_1(n,m) = O(nm \log_{\frac{m}{n \log n}} n)$ [19]. 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 maximum flow computation. Alternatively, the new maximum flow algorithm of Goldberg and Rao [14] with $T_1(n,m) = O(\min(n^{2/3}, m^{1/2}) m \log(\frac{n^2}{m}) \log U)$ may be used, if edge capacities can be expressed as integers in a range $[1, \ldots, U]$. We point out that if an edge representation of the output routing suffices, the Goldberg-Rao algorithm can be used to surpass the $O(nm)$ bottleneck. We sketch the idea.

The flow decomposition computation in Step 2 of PARTITION can be replaced by $\xi = O(\log(n/\varepsilon))$ maximum-flow computations; in each only the demands in the corresponding subinterval are included. The first maximum flow computation has available on every edge the capacity used by the original fractional solution f. The flow found at the i-th computation is subtracted from the capacity allotted to the $(i+1)$-th computation. This iterative procedure is correct for the same reason that standard flow decomposition is correct. In the latter case flow is subtracted along a single path (or cycle) at every iteration. In each iteration of our scheme, we use Goldberg-Rao to find all the flow paths leading to sinks with demand values in a given subinterval. Also the INTERVAL_ROUTING subroutine can be implemented by the Goldberg-Rao algorithm. The total running time of the new implementation would be $O(\log(n/\varepsilon)\min(n^{2/3}, m^{1/2})m\log(\frac{n^2}{m})\log(UD))$, assuming in the original network we have integral capacities in the range $[1,\ldots,U]$ and the minimum demand is $1/D$ for $D \in Z_{>0}$.

Although we can improve upon the 4-approximation, algorithm PARTITION introduces some of the basic ideas behind all of our algorithms. A demonstration of the algorithm execution on an example instance follows in Figures 4, 5, 6, 7. In these figures, numbers on vertices denote demands while numbers on edges denote capacities. The capacities on the original input are equal to 1. Flow paths are depicted as lines parallel to the original edges. Solid and dashed flow paths carry 1 and 1/2 units of flow respectively.

4 A 3-approximation algorithm for congestion

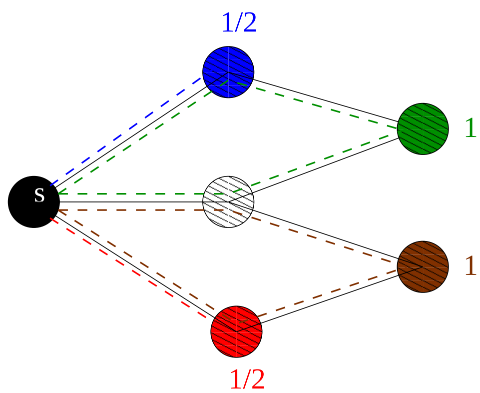
An improved approximation algorithm can be obtained by combining the partitioning idea above with a more careful treatment of the subproblems. At the first publication of our results [27] we obtained a $(10/3 + o(1))$-approximation. Subsequently we improved our scheme to obtain a 3 ratio; prior to this improvement, new results on UFP were independently obtained by Dinitz, Garg and Goemans [5]. See also [26].

We give first a specialized version of Lemma 3.2 to be used in the new algorithm.

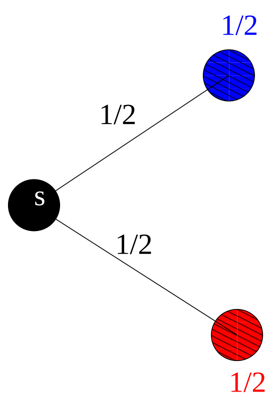
Lemma 4.1 *Let $\Pi = (G, s, \mathcal{T})$, be a* UFP *in which all demands have value $1/2^x$ for $x \in N$, and all capacities are multiples of $1/2^{x+1}$, and let f be a fractional flow solution such that the flow f_e through any edge is a multiple of $1/2^{x+1}$. We can find in polynomial time an unsplittable routing g where the flow g_e through an edge e is at most $f_e + 1/2^{x+1}$.*

Algorithm PARTITION finds a routing for each subproblem by scaling up subproblem capacities to ensure they conform to the requirements of Lemma 3.2. The new algorithm operates in phases, during which the number of distinct paths used to route flow for a particular commodity is progressively decreased. Call *granularity* the minimum amount of flow routed along a single path to any commodity, which has not been routed yet unsplittably. Algorithm PARTITION starts with a fractional solution of arbitrary granularity and essentially in one step per commodity, rounds this fractional solution to an unsplittable one. In the new algorithm, we delay the rounding process

Fractional Solution:



Subproblem G^1 for demands of $1/2$:



Subproblem G^2 for demands of 1:

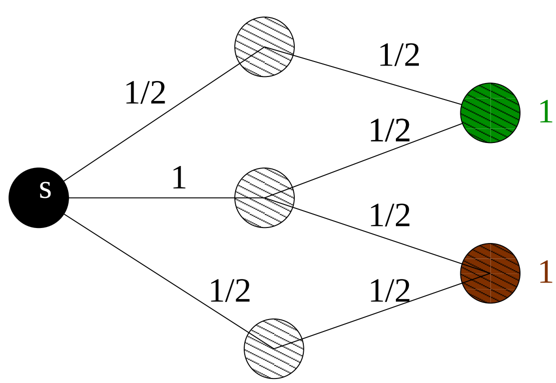


Figure 4: Algorithm PARTITION Demonstration I. Numbers on the edges and the vertices denote capacities and demands respectively. Original input capacities are 1.

Scale up capacities for G^1 to $2u_e^1 + (1/2)$. Unsplittable solution:

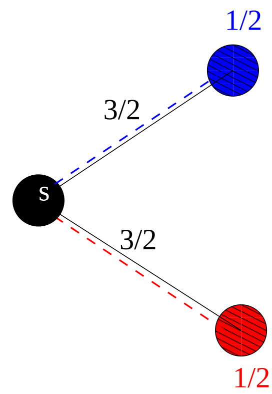


Figure 5: Algorithm PARTITION Demonstration II.

Scale up capacities for G^2 to $2u_e^2 + 1$. Unsplittable solution:

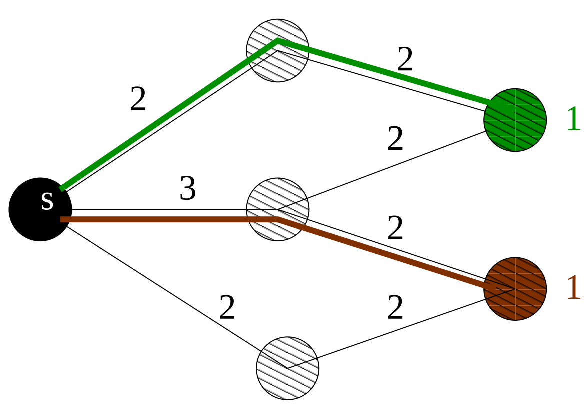


Figure 6: Algorithm PARTITION Demonstration III.

Superimpose the two solutions to obtain final solution:

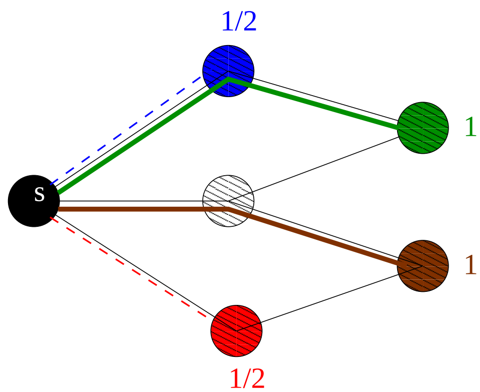


Figure 7: Algorithm PARTITION Demonstration IV. Final congestion is 3/2.

in the sense that we keep computing a fractional solution of successively increasing granularity. The granularity will always be a power of 1/2 and this unit will be doubled after each iteration. Once the granularity surpasses $1/2^x$, for some x, we guarantee that all commodities with demands $1/2^x$ or less have already been routed unsplittably. Every iteration improves the granularity at the expense of increasing the congestion. The method has the advantage that by Lemma 4.1, a fractional solution of granularity $1/2^x$ requires only a $1/2^x$ additive offset to current capacities to find a new flow of granularity $1/2^{x-1}$. Therefore, if the algorithm starts with a fractional solution of granularity $1/2^\lambda$, for some potentially large λ, the total increase to an edge capacity from then on would be at most $\sum_{j=1}^{j=\lambda} 1/2^j < 1$. Expressing the granularity in powers of 1/2 requires an initial rounding of the demand values; this rounding will force us to scale capacities by at most a factor of 2. We first provide an algorithm for the case in which demand values are powers of 1/2. The algorithm for the general case will then follow easily.

Theorem 4.1 *Let $\Pi = (G = (V, E, u), s, \mathcal{T})$ be a* UFP, *in which all demand values are of the form $2^{-\nu}$ with $\nu \in N$, the maximum demand value is denoted by $\rho_{\max}$, and the minimum demand value is denoted by $\rho_{\min}$. There is an algorithm,* 2H_PARTITION, *which obtains in polynomial time an unsplittable routing such that the flow through any edge e is at most $zu_e + \rho_{\max} - \rho_{\min}$ where z is the optimum congestion.*

Remark: We first obtained a relative $(1 + \rho_{\max} - \rho_{\min})$-approximation for the case with demand values in $\{1/2, 1\}$ in [27]. Dinitz, Garg and Goemans [5] first extended that theorem to arbitrary powers of 1/2 to obtain a $u_e + \rho_{\max} - \rho_{\min}$ absolute guarantee on the flow through edge e. Their derivation uses a different approach.

Proof. We describe the algorithm 2H_PARTITION. The crux of the algorithm is a relaxed decision procedure that addresses the following question. Is there an unsplittable flow after scaling all the input capacities by x? The procedure will either return no or will output a solution where the flow through edge e is at most $xu_e + \rho_{\max} - \rho_{\min}$. Embedding the relaxed decision procedure in a binary search completes the algorithm. See [15] for background on relaxed decision procedures. The following claim will be used to show the correctness of the procedure.

Claim 4.1 *Given an* UFP Π *with capacity function* r, *all demands of the form* $2^{-\nu}$, $\nu \in N$ *and minimum demand* $2^{-\lambda}$, *let* Π' *be the modified version of* Π *in which every edge capacity* r_e *has been rounded down to the closest multiple* r'_e *of* $1/2^{\lambda}$. *If the optimum congestion is* 1 *for* Π, *the optimum congestion is* 1 *for* Π' *as well.*

Proof of claim. Let G be the network with capacity function r and G' be the network with rounded capacities r'. The amount $r_e - r'_e$ of capacity will be unused by any optimal unsplittable flow in G. The reason is that the sum $\sum_{i \in S} 1/2^i$, for any finite multiset $S \subset N$ is equal to a multiple of $1/2^{i_0}$, $i_0 = \min_{i \in S}\{i\}$. ▮

We describe now the relaxed decision procedure. Let u' denote the scaled capacity function xu. Let $\rho_{\min} = 1/2^{\lambda}$. The relaxed decision procedure is broken into at most $\lambda - \log(\rho_{\max}^{-1}) + 1$ iterations; each iteration aims to double the granularity.

During the first iteration, split every commodity in $\mathcal{T}$, called an *original commodity*, with demand $1/2^y$, $0 \le y \le \lambda$, into $2^{\lambda - y}$ *virtual commodities* each with demand $1/2^{\lambda}$. Call the resulting set of commodities $\mathcal{T}_1$. Round down all capacities to the closest multiple of $1/2^{\lambda}$. By Theorem 2.1 we can find an unsplittable flow solution $f^0 \equiv g^1$, for $\mathcal{T}_1$, in which the flow through any edge is a multiple of $1/2^{\lambda}$. If this solution does not satisfy all demands the decision procedures returns no; by Claim 4.1 no unsplittable solution is possible without increasing the capacities u'. Else the decision procedure continues to iterate as described below. We denote by u_e^i, $i \ge 1$, the flow through edge e at the end of the i-th iteration. Observe that f^0 is a fractional solution for the set of original commodities as well, a solution with granularity $1/2^{\lambda}$. The flow u_e^1 through an edge is trivially $f_e^0 \le u'_e \le u_e$.

At iteration $i > 1$, the following steps are taken:

Step i1. Extract from g^{i-1}, using flow decomposition, the set of flow paths f^{i-1} used to route the set S_{i-1} of virtual commodities in $\mathcal{T}_{i-1}$ which correspond to original commodities in $\mathcal{T}$ with demand $1/2^{\lambda - i + 1}$ or higher.

Step i2. Pair up the virtual commodities in S_{i-1} corresponding to the same original commodity. Call the resulting set of virtual commodities $\mathcal{T}_i$.

Step i3. Find an unsplittable routing g^i for $\mathcal{T}_i$.

At Step i2, pairing is possible since the demand of an original commodity is inductively an even multiple of the demands of the virtual commodities. $\mathcal{T}_i$ will contain virtual commodities with demand $1/2^{\lambda - i + 1}$. For Step i3 observe that the set of flow paths f^{i-1} is a fractional solution for the commodity set $\mathcal{T}_i$. Inductively each flow path in f^{i-1} carries flow which is a cmultiple of $1/2^{\lambda - i + 2}$.

By Lemma 4.1, we can find an unsplittable routing g^i for $\mathcal{T}_i$ such that the flow g^i_e through an edge e is at most

$$f_e^{i-1} + 1/2^{\lambda-i+2}.$$

After the i-th iteration the granularity has been doubled to $1/2^{\lambda-i+1}$. A crucial observation is that from this quantity, g^i_e, only an amount of at most $1/2^{\lambda-i+2}$ corresponds to new flow, added to e during iteration i. It is easy to see that during the execution of the procedure the following two invariants are maintained.

INV1: After iteration i, all original commodities with demands $1/2^{\lambda-i+1}$ or less have been routed unsplittably.

INV2: After iteration $i > 1$, the total flow u^i_e though edge e, which is due to all i first iterations, satisfies the relation

$$u^i_e \leq u_e^{i-1} + 1/2^{\lambda-i+2}.$$

Given that $u^1_e = u'_e$ the total flow though e used to route unsplittably all commodities in $\mathcal{T}$ is at most

$$u'_e + \sum_{i=2}^{\lambda-\log(\rho_{\max}^{-1})+1} 1/2^{\lambda-i+2} \quad = xu_e + \rho_{\max} - 1/2^{\lambda} \quad = xu_e + \rho_{\max} - \rho_{\min}.$$

Therefore the relaxed decision procedure returns the claimed guarantee. For the running time, we observe that in each of the $l = \lambda - \log(\rho_{\max}^{-1}) + 1$ iterations, the number of virtual commodities appears to be $O(k2^l)$, where k is the number of the original commodities. We opted to present a "pseudopolynomial" version of the decision procedure for clarity. There are two approaches to ensure polynomiality of our method. One is to consider only demands of value $1/2^{\lambda} > 1/n^d$ for some $d > 1$. It is known [27] that we can route the very small demands while affecting the approximation ratio by an $o(1)$ factor. The other approach relies on changing the units at which demands and capacities are expressed. During iteration i map all virtual demands of value $1/2^{\lambda-i+1}$ to 1. The different virtual commodities hosted by the same sink t are not represented explicitly. Their number will be equal to the total demand of t under the new units. Lemma 4.1 requires the capacity of every edge e to be a multiple $d_e(1/2^{\lambda-i+1})$ of $1/2^{\lambda-i+1}$; map this capacity to d_e. An integral solution to the problem after remapping demands and capacities in the manner described is an unsplittable one for the virtual commodities of demand $1/2^{\lambda-i+1}$. ■

Recall that for a given UFP we assume that the Cut Condition holds. In the proof of Theorem 4.1 one can avoid the binary search under the following additional condition: all the capacities are multiples of $2^{-\lambda}$. E.g., integral capacities fulfill this requirement. Together with the Cut Condition, the extra condition implies that under the original capacity function u, a routing g^1 which satisfies all the demands in $\mathcal{T}_1$ can always be found. This gives the following corollary.

Corollary 4.1 *Let $\Pi = (G = (V, E, u), s, \mathcal{T})$ be a* Ufp, *in which all demand values are of the form $2^{-\nu}$ with $\nu \in N$, the maximum demand value is denoted by $\rho_{\max}$, and the minimum demand value is denoted by $\rho_{\min}$. If all capacity values are multiples of $\rho_{\min}$, algorithm* 2H_Partition *will obtain in polynomial time an unsplittable routing such that the flow through any edge e is at most $u_e + \rho_{\max} - \rho_{\min}$.*

Theorem 4.2 *Let $\Pi = (G = (V, E, u), s, \mathcal{T})$ be a* Ufp *with maximum and minimum demand values $\rho_{\max}$, and $\rho_{\min}$ respectively. There is an algorithm,* H_Partition, *which obtains in polynomial time a*

$$\min\{3 - \rho_{\min}, 2 + 2\rho_{\max} - \rho_{\min}\}$$

approximation for congestion.

Proof. We describe the algorithm H_Partition. Round up every demand to its closest power of $1/2$. Call the resulting Ufp Π' with sink set $\mathcal{T}'$. Multiply all capacities in Π' by at most 2 and let u' denote the resulting capacity function in Π'. Then there is a fractional feasible solution f' to Π'. The purpose of the rounding is to always be able to express the granularity as a power of $1/2$. The remainder of the proof consists of finding an unsplittable solution to Π'; this solution can be efficiently converted to an unsplittable solution to Π without further capacity increase. Such a solution to Π' can be found by the algorithm 2H_Partition from Theorem 4.1. Observe that the optimum congestion for Π' is at most z, the optimum congestion for Π. 2H_Partition will route through edge e flow that is at most

$$zu'_e + \rho'_{\max} - \rho'_{\min} \leq 2zu_e + \rho'_{\max} - \rho'_{\min},$$

where $\rho'_{\max}$ and $\rho'_{\min}$ denote the maximum and minimum demand values in Π'. But $\rho'_{\max} < 2\rho_{\max}$, $\rho'_{\max} \leq 1$, and $\rho'_{\min} \geq \rho_{\min}$ from the construction of Π'. Therefore $\rho'_{\max} - \rho'_{\min} \leq 1 - \rho_{\min}$ and $\rho'_{\max} - \rho'_{\min} \leq 2\rho_{\max} - \rho_{\min}$. Dividing the upper bound on the flow by $zu_e \geq 1$ yields the claimed guarantees on congestion. ■

5 Minimizing congestion with arbitrary capacities

We examine here Ufp in its full generality, when demands lie in $(0, 1]$ and capacities in $(0, \infty)$. In particular, let $1/D$ be the minimum demand value for a real $D \geq 1$. We assume without loss of generality that $u_e \geq 1/D$, for all e, since the optimal solution will not use any edge violating this condition. Our approximation algorithm will have time complexity polynomial in $\lfloor \log D \rfloor$. In subsequent sections we return to the original Balance Condition on the capacity values as stated in Section 2.

The essential 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 $\rho_i > u_e$. These constraints can be easily enforced, for example, in a linear programming formulation. The resulting problem formulation is a valid relaxation of the original problem and the unsplittable optimum is not affected by the additional constraints. A second modification is that no subproblem is constructed for a range of the form $(0, \alpha_1]$. The first subinterval of the decomposition is $[a_1, a_2]$ with $a_1 = 1/D$. 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 5.1 *Algorithm* L_PARTITION *runs in polynomial time and outputs an unsplittable routing* g *with total flow through edge* e *at most* $\max_{2\leq i\leq \nu}\{r(\alpha_{i-1}, \alpha_i)\}u_e + \sum_{i=2}^{\nu} \alpha_i$*, where* $u_e \in (\alpha_{\nu-1}, \alpha_\nu]$*,* $\nu \leq \xi$*.*

Proof. The proof is 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 $i \geq \nu$. ■

Theorem 5.1 *Given* UFP Π *with arbitrary capacities and minimum demand value* $\rho_{\min} = 1/D$*,* $D \geq 1$*, Algorithm* L_PARTITION *finds a* $(5.8285 - 1.4144\rho_{\min})$*-approximation for congestion. The algorithm runs in polynomial time.*

Proof. We instantiate the partitioning scheme as follows. The parameter $r > 1$, to be chosen later, is the ratio of the geometrically decreasing intervals. (In the proof of Theorem 3.1, $r = 2$.) We partition the interval $[1/D, 1]$ of demands into $O(\log_r D)$ geometrically increasing subintervals:

$$[1/D, 1/r^{\lfloor \log_r D\rfloor}], \ldots, (1/r^{i+1}, 1/r^i], \ldots, (1/r, 1].$$

By Lemma 5.1 the total flow through an edge e such that $1/r^{i+1} < u_e \leq 1/r^i$ is at most

$$ru_e + \sum_{j=i}^{\lfloor \log_r D\rfloor} 1/r^j = ru_e + \frac{r}{r^i(r-1)} - \frac{1}{(r-1)r^{\lfloor \log_r D\rfloor}} \leq ru_e + \frac{r^2 u_e}{r-1} - \frac{u_e}{(r-1)D}.$$

Thus we obtain a congestion of $\frac{2r^2-r}{r-1} - \frac{1}{(r-1)D}$. Selecting $r = 1.707$, yields $5.8285 - 1.4144\rho_{\min}$. This analysis hinges on the fact that $u_e \leq 1$. The total flow through an edge e with $u_e > 1$ will be

$$ru_e + \sum_{j=0}^{\lfloor \log_r D\rfloor} 1/r^j < ru_e + \frac{r}{r-1}u_e,$$

which is at most $4.12143u_e$ when $r = 1.707$. Since $\rho_{\min} \leq 1$, $4.12143 < 5.8285 - 1.4144\rho_{\min}$. ■

6 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 $c_e \geq 0$ is associated with every edge e. The cost $c(P)$ of a path P is $\sum_{e\in P} c_e$. The objective of the optimization problem we consider is to find among all unsplittable flow solutions of minimum congestion the one which minimizes the total cost $\sum_{t_i \in \mathcal{T}} \rho_i c(P_i)$. We show how to modify algorithm PARTITION to invoke a minimum-cost flow subroutine instead of maximum-flow routines. The new algorithm will 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 6.1 *Let $(G = (V, E, u), s, \mathcal{T})$ be a* UFP *with costs on an arbitrary network G with all demands ρ_i, $1 \leq i \leq k$, equal to the same value σ and edge capacities u_e equal to integral multiples of σ for all $e \in E$. 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 minimum-cost flow (developed independently by [17], [16], [3]); it is a corollary of the integrality property of minimum-cost flow, with integral units scaled by σ.

Two lemmata for the analysis follow. They generalize Lemmata 3.1 and 3.2 to accommodate costs.

Lemma 6.1 *Given a minimum-cost* UFP *with demands in the interval $(0, \alpha]$ and a fractional solution f with cost $c(f)$, there is an algorithm,* α-C_ROUTING, *which finds an unsplittable routing of cost at most $c(f)$, such that the flow through an edge is at most $n\alpha$. The running time of the algorithm is $O(n \log n + m)$.*

Proof. Algorithm α-C_ROUTING works as follows. Find shortest paths from s to all the sinks in G using Dijkstra's algorithm [4, 10]. For each sink t_i route on the shortest path from s flow equal to the demand ρ_i. ∎

Lemma 6.2 *Let a minimum-cost* UFP Π *have demands in the interval $(a, b]$, arbitrary capacities, and a fractional solution f of cost $c(f)$. There is an algorithm,* C_INTERVAL_ROUTING, *which finds in polynomial time an unsplittable routing of cost at most $r(a, b)c(f)$ such that the flow through an edge e is at most $r(a, b)u_e + b$.*

Proof. The proof is similar to that of Lemma 3.2, but we use Theorem 6.1 instead of Theorem 2.1. Round up all the demands to b and call Π' the resulting problem. By multiplying the flow f_e and the capacity u_e on every edge e by at most $r(a, b)$ we obtain a feasible fractional solution f' for Π'; the cost of f' is at most $r(a, b)c(f)$. Now add at most b to every edge capacity so that it becomes a multiple of b. Find an unsplittable routing by using Theorem 6.1. The cost of the unsplittable routing is at most equal to the cost of the fractional solution f'. ∎

Algorithm C_PARTITION takes the same steps as PARTITION, with two differences in Steps 1 and 3. At Step 1 of C_PARTITION a fractional minimum-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 minimum-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. The proof is very similar to that of Lemma 3.3 and we omit it.

Lemma 6.3 *Algorithm* C_PARTITION *outputs an unsplittable routing g with 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 $\max_{2\le i\le\xi}\{r(\alpha_{i-1},\alpha_i)\}c(f)$, *where $c(f)$ 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 minimum-cost flow problem.*

Using the same partitioning scheme as in Theorem 3.1 we obtain:

Theorem 6.2 *Given a minimum-cost* UFP Π, *we can obtain a simultaneous $(2, 4+\varepsilon)$ approximation for cost and congestion, for any $\varepsilon > 0$. The running time of the algorithm is $O(T_2(n,m) + nm + m\log(n/\varepsilon))$ where $T_2(n,m)$ is the time to solve a fractional minimum-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 6.3 *Given a minimum-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 $O(T_2(n,m) + m\log(n/\varepsilon))$ where $T_2(n,m)$ is the time to solve a fractional minimum-cost flow problem.*

Proof. The new algorithm is the same as C_PARTITION except for the partitioning scheme. The interval $(0,1]$ of demands is partitioned into ξ geometrically increasing subintervals

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

such that $1/(\delta+1)^{\xi-1} \le \varepsilon/n$. According to this scheme, by Lemma 6.3 the approximation ratio for the cost will be $1+\delta$ and the approximation ratio for the congestion will be at most

$$\begin{aligned} 1+\delta+\sum_{i=0}^{\xi-2}\frac{1}{(\delta+1)^i}+\varepsilon &\le \\ 1+\delta+\frac{1+\delta}{\delta}+\varepsilon &= \\ 2+\frac{\delta^2+1}{\delta}+\varepsilon. \end{aligned}$$

■

Currently the best time bound $T_2(n,m)$ for fractional minimum-cost flow is

$$O(\min\{nm\log(n^2/m)\log(nC), nm(\log\log U)\log(nC), (m\log n)(m+n\log n)\})$$

due to [13, 1, 32]. Finally, by modifying the congestion algorithm of Theorem 4.2, it is easy to see how to obtain a $(2,3)$ simultaneous approximation for cost and congestion. The essential modification to H_PARTITION lies in the use of the minimum-cost analogue to Lemma 4.1.

Lemma 6.4 *Let $\Pi = (G, s, \mathcal{T})$, be a minimum-cost* UFP, *in which all demands have value $1/2^x$ for $x \in N$, and all capacities are multiples of $1/2^{x+1}$, and let f be a fractional flow solution of cost $c(f)$ such that the flow f_e through any edge is a multiple of $1/2^{x+1}$. We can find in polynomial time an unsplittable routing g of cost at most $c(f)$ such that the flow g_e through an edge e is at most $f_e + 1/2^{x+1}$.*

Theorem 6.4 *Let Π be a minimum-cost* UFP, *with maximum and minimum demand values $\rho_{\max}$, and $\rho_{\min}$ respectively. We can obtain in polynomial time a simultaneous*

$$(2, \min\{3-\rho_{\min}, 2+2\rho_{\max}-\rho_{\min}\})$$

approximation for cost and congestion.

7 Maximizing the routable demand

In this section we turn to the maximization problem of finding a routable subset of sinks with maximum total demand. Let the *value* of a (partial) routing be the sum of the demands of the commodities being routed. We seek a routing of maximum value such that the capacity constraints are satisfied. We obtain a .075 approximation which is slightly improved in the next section to .0769 as an indirect application of the algorithm for minimizing the number of rounds; the latter algorithm uses subroutines from matroid theory. In contrast to most of the results in this paper the Cut Condition is a necessary prerequisite for the .0769-approximation. Therefore the .075-approximation is our most generally applicable algorithm for maximum demand. The exposition in the present section introduces in a direct fashion the general technique while the algorithm invokes only maximum flow subroutines. Note that if the value definition was relaxed to take into account paths carrying flow to commodities without fully satisfying the respective demands, a c-approximation, $c > 1$, for congestion would imply a $1/c$-approximation for this relaxed maximization problem.

Let $(G, s, \mathcal{T})$ be the UFP we want to solve. Consider splitting the interval $(0,1]$ of demands into two intervals $(0,\alpha]$, and $(\alpha,1]$, for an appropriate *breakpoint* α. We will give separate constant-factor approximation algorithms for each of the two resulting problems. At least one of the two intervals contains at least half of the total demand in $\mathcal{T}$, so we obtain a constant-factor approximation for the entire problem. This was also the high-level approach used in [23]. In fact, in order to route the demands in the subinterval $(\alpha,1]$ we will use an intuitive algorithm developed by Kleinberg

[23]. 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. For our partitioning scheme to succeed in the maximization setting, it is essential to have the maximum demand bounded away from 1. Hence the different treatment for $(0, \alpha]$, and $(\alpha, 1]$.

Let $\alpha^f(G, s, \mathcal{T})$ denote the total demand routed by a fractional solution f. For any given UFP we will use $\alpha^f(G, s, \mathcal{T})$, with f a fractional solution of maximum value, as an upper bound to measure the quality of our approximation.

The following lemma, shown in [23], will be used to attack the demands in $(\alpha, 1]$. The spirit of the proof is similar to the one of Lemma 3.2.

Lemma 7.1 *[23] Given an* UFP $\Pi = (G, s, \mathcal{T})$, *with demands in the interval* $(a, b]$, *there is an algorithm,* K_ROUTING, *which 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 7.1 we can show a slightly different lemma, which will be also of use in our algorithm. We include the proof for the sake of completeness.

Lemma 7.2 *Given an* UFP $\Pi = (G, s, \mathcal{T})$, *with demands in the interval* $(a, b]$, *and capacities that are multiples of* b, *there is an algorithm,* M_ROUTING, *which 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 algorithm M_ROUTING. Let f be a fractional solution to Π. Round down all demands to a. Multiply all capacities in G by $[r(a,b)]^{-1} < 1$. 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 at least $\alpha^{f'}(G', s, \mathcal{T}')$. 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 of value at least $\alpha^{f'}(G', s, \mathcal{T}')$. ■

We now outline our approach for the second half of the problem, namely the demands in $(0, \alpha]$. Our aim is to use a partitioning scheme as in the congestion case: partition the interval $(0, \alpha]$ of demand values into subintervals and generate an appropriate subproblem for each subinterval. We would like to find a near-optimal solution to each subproblem, by exploiting Lemma 7.2. Ideally, we could then combine these near-optimal solutions to the subproblems to obtain a close-to-maximum unsplittable flow for the original problem. However, in the congestion version, only a fraction of the capacity of an edge is assigned initially to each subproblem, a fraction determined by flow decomposition. On the other hand, in order to make use of Lemma 7.2 on each subproblem, we require capacities to be multiples of the maximum demand. In the congestion algorithm, adding

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 amount c_e^i of u_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 u_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.

Figure 8: Algorithm M_PARTITION.

the necessary amount to each capacity only increased the approximation ratio. In the current setting we are not allowed to increase the original input capacities, as this would violate feasibility. To circumvent this difficulty we scale down the fractional solution to the original problem. 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 by a $1 - x$ factor, for appropriately chosen x, will incur a scaling by the same factor to the approximation ratio, as the value of the final routing is given in terms of the value $\alpha^f(G, s, \mathcal{T})$ of the maximum flow with which we begin. We give the general algorithm in Figure 8. It takes as input a UFP $(G, s, \mathcal{T})$ together with a set of parameters $x, \xi, \alpha_1, \ldots, \alpha_\xi$ that will determine the exact partitioning scheme. The breakpoint α will be $\alpha_{\xi-1}$ with α_ξ set to 1. Subsequently we show how to choose these parameters so as to optimize the approximation ratio. The algorithm outputs a partial routing.

By Lemma 7.1, partial routing g^ξ computed in Step 4 is a partial unsplittable flow, which respects the capacities. We need to establish this fact for partial routing g computed in Step 5 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 $\alpha^f(G, s, \mathcal{T}_i)$ denote the flow routed by the fractional solution to demands in the interval

$(\alpha_{i-1}, \alpha_i]$.

Lemma 7.3 *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 maximum flow problem.*

Proof. We examine first the value of g. Let f_i^x be the fractional optimal flow on G_i. By Lemma 7.2 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, 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 $\sum_{i=2}^{i=\xi-1} c_e^i \leq u_e$. Thus, the aggregate capacity used by g is at most $(1-x)u_e + x \leq u_e$. 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 $\alpha^f(G, s, \mathcal{T}_1)$. Without loss of generality we assume that there is one demand in $\mathcal{T}$ of value 1. Otherwise we can rescale the interval of demands. Thus $\alpha^f(G, s, \mathcal{T})$ is at least 1.

Theorem 7.1 *Let x_1, x_2, x_3 be constants in $(0,1)$ such that $x_3 = x_1(1-x_2)$. Given an* UFP *$\Pi = (G, s, \mathcal{T})$, and choosing parameters based on x_1, x_2, x_3, algorithm* M_PARTITION *finds a $\beta(1-\varepsilon)$-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 $O(T_1(n,m) + nm + m\log(n/\varepsilon))$ where $T_1(n,m)$ is the time to solve a fractional maximum flow problem.*

Proof. At step 3 of M_PARTITION, 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 $x_3(x_2)^{\xi-2} \leq \varepsilon/n$. Thus it suffices for ξ to be $\Theta(\log n/\varepsilon)$. The flow $\alpha^f(G, s, \mathcal{T}_1)$ is at most ε so

$$\alpha^f(G, s, \bigcup_{2 \leq i \leq \xi} \mathcal{T}_i) \geq \alpha^f(G, s, \mathcal{T}) - \varepsilon \geq (1-\varepsilon)\alpha^f(G, s, \mathcal{T}).$$

Achieving therefore a β-approximation to $\alpha^f(G, s, \bigcup_{2 \leq i \leq \xi} \mathcal{T}_i)$ will yield the claimed $\beta(1-\varepsilon)$ ratio.

Set parameter x in the algorithm to x_1. Note that by the choice of x_3 the sum $\sum_{i=0}^{i=\xi-2} x_3(x_2)^i$ is at most x_1 as required by the algorithm. By Lemma 7.3, the partial unsplittable flow g found by

M_Partition is a $(1-x_1)x_2$-approximation to $\alpha^f(G,s,\bigcup_{2\leq i\leq \xi-1}\mathcal{T}_i)$. By Lemma 7.1, the partial unsplittable flow g^ξ is a $x_3/2$-approximation to $\alpha^f(G,s,\mathcal{T}_\xi)$. Choosing of the two the one of greatest value at Step 5 yields a β-approximation to $\alpha^f(G,s,\bigcup_{2\leq i\leq \xi}\mathcal{T}_i)$. ■

By choosing x_1, x_2, x_3 to be $4/10, 1/4$ and $3/10$ respectively we obtain $(1-x_1)x_2 = x_3/2 = 0.15$. Accordingly, $\beta = .075$.

8 Minimizing the number of rounds

In this section we examine 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. The requirement to satisfy the capacity constraints leads to a high-level treatment similar to the one for the maximum demand metric. We split the interval of demand values into $(0,\alpha]$ and $(\alpha,1]$, for an appropriate breakpoint α, and give a constant-factor approximation for each of the two resulting problems. Demands in $(0,\alpha]$ will be processed under a partitioning scheme, that will generate subproblems. In the subproblems, we relax the capacities to allow congestions of up to $1/(1-x)$ in order to free up capacity that we can reallocate conveniently among subproblems. This scaling, will be reflected in an increase on the approximation ratio by $1/(1-x)$. Intuitively, since we are using paths from the fractional solution in the unsplittable routings, when the fractional solution carries $(1-x)$ times less flow for each commodity, we need more rounds.

Recall that the input Ufp comes with the assumption that a fractional solution with relative congestion 1 exists, i.e., the Cut Condition holds. In this section we will generate subproblems on which this assumption does not hold. Therefore we introduce notation $\zeta^f(G,s,\mathcal{T})$ for the congestion of a *fractional* solution f to an Ufp $(G,s,\mathcal{T})$. Note that if f is a fractional solution of minimum congestion for a problem $(G,s,\mathcal{T})$, the quantity $\lceil\zeta^f(G,s,\mathcal{T})\rceil$ is a lower bound on the minimum number of rounds, which we denote by $\chi(G,s,\mathcal{T})$. Our analysis bounds the number of rounds against $\zeta^f(G,s,\mathcal{T})$, which we have assumed to be 1 (because of the Cut Condition). But since $\chi(G,s,\mathcal{T}) \geq \lceil\zeta^f(G,s,\mathcal{T})\rceil$, the approximation ratio holds even if we drop this assumption. For the sake of clarity we include $\zeta^f(G,s,\mathcal{T})$ in the approximation guarantees because we find that the rounds metric is the less intuitive of the three metrics as far as validity of the approximations without the Cut Condition is concerned.

We employ a subroutine that we call KR_Routing (see Lemma 5.3 in [23]) and a variant R_Routing to deal with subproblems having demands in a bounded range. 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. Maximum flow integrality is not useful in the rounds setting and instead a result from [23] is used, given as Theorem 8.1 below. The proof of the latter theorem uses results from matroid theory.

Theorem 8.1 *[23] Given an* Ufp $(G,s,\mathcal{T})$ *with all demands equal to* σ *and all capacities mul-*

tiples of σ, $\lceil \zeta^f(G,s,\mathcal{T}) \rceil = \chi(G,s,\mathcal{T})$. Moreover, a routing in $\chi(G,s,\mathcal{T})$ rounds can be found in polynomial time.

Lemma 8.1 *[23] Let $\Pi = (G,s,\mathcal{T})$, be a* UFP *with demands in the interval $(a,b]$, and f a corresponding fractional solution. There is a polynomial-time algorithm,* KR_ROUTING, *which routes Π in at most $\lceil 2r(a,b)\zeta^f(G,s,\mathcal{T}) \rceil$ rounds.*

Again we can adapt the proof of Lemma 8.1 to show a slightly different result.

Lemma 8.2 *Let $\Pi = (G,s,\mathcal{T})$ be a* UFP *with demands in the interval $(a,b]$, and capacities multiples of b, and f a corresponding fractional solution not necessarily respecting capacities. There is a polynomial-time algorithm,* R_ROUTING, *which routes Π in at most $\lceil r(a,b)\zeta^f(G,s,\mathcal{T}) \rceil$ rounds.*

Proof. We outline algorithm R_ROUTING. Round all the demands up to b. Call Π' the resulting problem and f' a corresponding fractional solution not necessarily respecting capacities. In Π' all demands are equal to b and all capacities are multiples of b; therefore by Theorem 8.1 we can route in $\chi(\Pi') = \lceil \zeta^{f'}(\Pi') \rceil$ rounds and use the paths from this routing to route Π. Clearly, the paths used in each round respect the capacity constraints in $(G,s,\mathcal{T})$. This completes the description of the algorithm.

It remains to demonstrate a suitable f', which will help us to upper bound $\zeta^{f'}(\Pi')$ in terms of $\zeta^f(G,s,\mathcal{T})$. Decompose the flow f for Π, and consider the paths routing demand to a particular sink t_i. Multiply the flow on each path by the same constant (no greater than b/a) so that b units of flow are routed to each t_i. Do this for all sinks. Let f' be the resulting fractional solution. f' satisfies all the demands for Π' and has congestion $\zeta^{f'}(\Pi')$ at most $r(a,b)\zeta^f(G,s,\mathcal{T})$. ■

Our algorithm R_PARTITION and its analysis are very similar to M_PARTITION up to the subroutine level. When dealing with $(\alpha_1, \alpha_{\xi-1}]$, we scale down the fractional capacity on each edge by $(1-x)$, thereby inducing congestion of up to $1/(1-x)$ in the subproblems, in order that we can reallocate capacity among the subproblems to satisfy the hypotheses of Lemma 8.2. 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, $2 \le i \le \xi - 1$, and G_ξ respectively. Algorithm R_ROUTING outputs for each G_i, $2 \le i \le \xi - 1$, 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, $2 \le i \le \xi - 1$. During round j, $1 \le j \le j^*$, we route along all paths in $\bigcup_i \mathcal{P}_{ij}$. Subsequently we route the demands in $(0, \alpha_1]$ and $(\alpha_{\xi-1}, 1]$ in two separate sets of rounds. Recall that $\zeta^f(G,s,\mathcal{T}_i)$ is the minimum congestion for routing demands fractionally in $\mathcal{T}_i$ on the original unscaled capacities and thus is equal to 1. The following lemma accounts for the rounds needed to route all demands except for the ones in the $(0, \alpha_1]$.

Lemma 8.3 *Algorithm* R_PARTITION *runs in polynomial time and routes the demands in $(\alpha_1, 1]$ in at most*

$$\max_{2 \le i \le \xi - 1} \{ \lceil \frac{1}{1-x} r(\alpha_{i-1}, \alpha_i) \zeta^f(G,s,\mathcal{T}_i) \rceil \} + \lceil 2r(a_{\xi-1}, 1)\zeta^f(G,s,\mathcal{T}_\xi) \rceil$$

rounds.

Proof. For the number of rounds to route demands in $(\alpha_i, \alpha_{i+1}]$, $1 \leq i \leq \xi - 2$, we note that Lemma 8.2 applies in the corresponding subproblems. A fractional solution satisfying all demands in subproblem G_i would have to potentially introduce congestion given that at Step 3, we assign capacity to edge e in G_i which is as low as $(1-x)c_e^i$. There is a fractional solution f_i to G_i with congestion at most $\frac{1}{1-x}\zeta^f(G, s, \mathcal{T}_i)$, obtained by setting $f_e^i = c_e^i$. Therefore, the number of rounds for a subproblem is at most $\lceil \frac{1}{1-x} r(\alpha_{i-1}, \alpha_i)\zeta^f(G, s, \mathcal{T}_i)\rceil$. By exactly the same argument as in Lemma 7.3, the aggregate capacity on any edge e used on all these subproblems does not exceed u_e. Thus during the same round j, $1 \leq j \leq j^*$, we can route all paths in $\bigcup_{2 \leq i \leq \xi - 1} \mathcal{P}_{ij}$.

To route the demands in $(\alpha_{\xi-1}, 1]$ we need by Lemma 8.1 at most an additional $\lceil 2r(a_{\xi-1}, 1)\zeta^f(G, s, \mathcal{T}_\xi)\rceil$ number of rounds. ■

By choosing $\alpha_1 \leq 1/n$, all the demand in $(0, \alpha_1]$ can be routed in one round, by Lemma 3.1.

Theorem 8.2 *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 $\beta = \lceil 1/(1-x_1)x_2\rceil + \lceil 2/x_3\rceil + 1$.*

Proof. The partitioning scheme is the same as in the proof of Theorem 7.1 with $\varepsilon = 1$. The demands in $(0, x_3(x_2)^{\xi-2}]$ are small enough to be routed in one round using the algorithm α-Routing from Lemma 3.1. Substituting x_1, x_2, x_3 in the number of rounds given by Lemma 8.3 and adding one extra round for the demands in $(0, x_3(x_2)^{\xi-2}]$ completes the proof. ■

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

Routing all commodities in x rounds implies that during one round at least a $1/x$ fraction of the total demand is routed. If for the problem $\Pi = (G, s, \mathcal{T})$ we begin with $\zeta^f(G, s, \mathcal{T}) = 1$, i.e. the Cut Condition is met, Theorem 8.2 implies that we can route all commodities in at most 13 rounds. During one of those at least $1/13 = .0769$ of the total demand is routed.

Corollary 8.1 *Given an* Ufp $\Pi = (G, s, \mathcal{T})$*, which satisfies the Cut Condition we can obtain in polynomial time a* .0769 *approximation for maximizing routable demand.*

Observe that Theorem 7.1 guarantees routing at least .075 of $\alpha^f(G, s, \mathcal{T})$ which is an upper bound on the maximum routable demand but less than the total demand if the Cut Condition is not met. We do not see how to apply Theorem 8.2 to obtain a guarantee for maximum demand when the Cut Condition is violated.

9 A hardness result for unsplittable flow

In this section we consider the hardness of approximation for minimum congestion unsplittable flow on a directed network with 2 sources (2-Ufp). We give a gap-preserving reduction from the NP-complete problem 3-D Matching [18]. Our reduction establishes that it is NP-hard to achieve an approximation ratio better than 2 for 2-Ufp. In the 3-D Matching problem, we are given a set $M \subseteq A \times B \times C$, where A, B, and C are disjoint sets each of cardinality n. The objective is to find a *perfect matching,* i.e. a subset $M' \subseteq M$ such that $|M'| = n$ and no two elements of M' agree in any coordinate. In our reduction we use ideas from Theorem 5 in [31].

Lemma 9.1 *Given an instance I of* 3-D MATCHING, *we can obtain in polynomial time an instance I' of* 2-UFP *on a directed network 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 $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$, and $C = \{c_1, \ldots, c_n\}$ so that the m triples in I are of the form (a_i, b_j, c_k). 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 and c_k in I, we have respectively a vertex b_j and $\bar{c_k}$ in I'. We refer collectively to the $2n$ vertices b_j, $\bar{c_k}$, $1 \leq j, k \leq n$, as *b*- and *c-type vertices* respectively. There are m commodities corresponding to terminal pairs $(s_c, \bar{c_k})$, $1 \leq k \leq n$, and $(s_d, \bar{d_{i\tau}})$, $1 \leq i \leq n, 1 \leq \tau \leq t_i - 1$, and they 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* (see Fig. 9) 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 there is 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. For convenience let us identify a gadget with the triple it represents. Finally, an edge is directed from s_d to each x_μ, $1 \leq \mu \leq m$, and from s_c to each b_j, $1 \leq j \leq n$.

Since all demands and capacities are 1, every feasible unsplittable flow is simply a set of disjoint source-sink paths each carrying unit flow. Note that each flow path must pass through some gadget μ and in particular through the edge (x_μ, y_μ) of the gadget.

If I contains a 3-D perfect matching M', it is straightforward to construct an unsplittable flow g in G with congestion 1. Flow g uses the n gadgets corresponding to triples in M' to satisfy the demands of vertices $\bar{c_k}$, $1 \leq k \leq n$. The remaining gadgets are used to route 1 unit of flow each (via vertex y_μ for the μ-th gadget) to a dummy vertex. Since M' is a matching, only one of the t_i gadgets associated with a_i, for any i, is used to route flow to a c-type vertex. As a result, for any i, there are $t_i - 1$ gadgets available to route the dummy vertices associated with a_i.

We show now that if I' has an unsplittable flow g', which satisfies the capacity constraints, then I contains a 3-D perfect matching. Let Γ be the set of n gadgets used by g' to route flow to c-type vertices. We claim that Γ forms a 3-D perfect matching in I. For each $1 \leq i \leq n$, every dummy vertex $\bar{d_{i\tau}}$ $(1 \leq \tau \leq t_i - 1)$ requires one flow path, so only one gadget in Γ corresponds to α_i. Each node b_i has in-degree 1, so only one gadget in Γ contains b_i. And the demand at each node c_i is 1, so only one gadget in Γ contains c_i. ■

The ensuing theorem is an immediate consequence of Lemma 9.1.

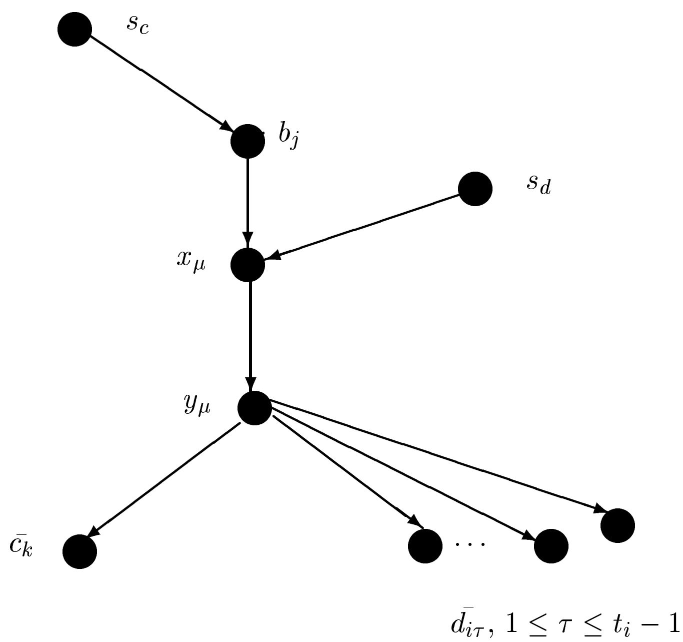


Figure 9: Gadget corresponding to the μ-th triple (a_i, b_j, c_k) together with the edges connecting it to the two sources.

Theorem 9.1 *No ρ-approximation algorithm, $\rho < 2$, exists for* 2-UFP *on directed graphs 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, which minimizes the *makespan,* i.e. the maximum completion time of a job. $\mathcal{S}$ is a special case of minimizing makespan on unrelated machines. In the unrelated machine setting, job j has a machine-dependent processing time p_{ij} on $i \in M$. The best approximation algorithm known for $\mathcal{S}$ is the 2-approximation for unrelated machine scheduling [31, 37]. From a straightforward modification to Theorem 5 in [31], the following hardness result is obtained.

Theorem 10.1 *[31] Unless P=NP, no approximation better than* $3/2$ *exists for $\mathcal{S}$ with processing times from the set* $\{1, 2, \infty\}$.

Kleinberg [23] gave an approximation-preserving reduction from $\mathcal{S}$ to minimum congestion unsplittable flow on a three-level directed graph G. A source vertex has edges directed to vertices representing the machines. The vertex set of G contains also one vertex for each job. Machine vertex i has an edge directed to job vertex j if and only if $i \in M(j)$. The edges out of the source have capacity T and the edges into the job vertices have infinite capacity. Finally every job vertex has demand equal to p_j. An unsplittable routing in G where at most ρT amount of flow is pushed through any edge corresponds to a schedule with makespan ρT for $\mathcal{S}$. See Figure 2 for an example network.

Consider a UFP on an arbitrary network with demands p and $2p$, for some $p > 0$. This includes the family of networks obtained for the scheduling problem with processing times $\{1, 2, \infty\}$. As a corollary to Theorem 4.1 and Corollary 4.1 we obtain a tight approximation ratio of 3/2. The lower bound comes from Theorem 10.1.

Corollary 10.1 *Given an* UFP $\Pi = (G, s, \mathcal{T})$, *with demands from the set* $\{p, 2p\}$, $0 < p \leq 1/2$, *there is an algorithm to find in polynomial time an unsplittable routing where the flow through an edge* e *is at most* $zu_e + p$, *where* z *is the optimal congestion. If all capacities are multiples of* p *the flow is at most* $u_e + p$. *Thus the approximation ratio for congestion is at most* $3/2$. *This is best possible, unless P=NP.*

Corollary 10.2 *For problem* $\mathcal{S}$ *with processing times from* $\{p, 2p, \infty\}$ *there is a polynomial-time algorithm, which outputs a schedule within an additive* p *of the optimum makespan. The approximation ratio is at most* $3/2$ *and this is best possible, unless P=NP.*

It is instructive to consider the action taken by algorithm 2H_PARTITION on the UFP resulting from scheduling problem $\mathcal{S}$. The algorithm splits every job of processing time $2p$, into two virtual jobs each of processing time p. Then a schedule g_1 with optimum makespan is computed for the resulting instance. In the scheduling context, when all the jobs have the same processing time, an optimum schedule can be found by solving an assignment problem. Schedule g_1 corresponds to a half-integral superoptimal solution for $\mathcal{S}$. In the second phase of 2H_PARTITION this half-integral solution is rounded to an integral one.

The results of Corollary 10.1 can be generalized to the following.

Corollary 10.3 *Given an* UFP $\Pi = (G, s, \mathcal{T})$, *with demands from the set* $\{p, Cp\}$, $C > 1, 1/C \geq p > 0$ *there is an algorithm to find in polynomial time an unsplittable routing where the flow through an edge* e *is at most* $zu_e + (C-1)p$, *where* z *is the optimal congestion. If all capacities are multiples of* p *the flow is at most* $u_e + (C-1)p$. *Thus the approximation ratio for congestion is at most* $2 - \frac{1}{C}$.

We now proceed to examine the case, in which the demands lie in the interval $[p, Cp]$, $p > 0$. Lemma 3.2 would give in this setting a $(C+1)$-approximation for congestion, since the minimum capacity is now Cp. We show how to achieve a C-approximation, the same as the ratio of the interval.

Theorem 10.2 *Given an* UFP $\Pi = (G, s, \mathcal{T})$, *with demands from the interval* $[p, Cp]$, $C > 1, p > 0$, *there is an algorithm to find in polynomial time an unsplittable routing where the flow through an edge* e *is at most* Cu_e.

Proof. As in the proof of Theorem 4.1 we assume without loss of generality that Π has an unsplittable routing with congestion 1. Otherwise one can use the algorithm we propose as a C-relaxed decision procedure [15] in conjunction with a binary search for the optimum congestion to obtain the claimed approximation. The algorithm is as follows. Round all demands down to p. Call the resulting problem 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 for 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. To obtain from g' an unsplittable routing for Π it suffices to increase the flow along paths in g' that lead to sinks in $\mathcal{T}$ with demands more than p. The increase will be at most by a multiplicative factor of C on each path, hence the approximation theorem. ■

Interestingly, the approach in Theorem 10.2 does not seem to yield an improvement on the result of Lemma 3.3. For problem Π' in the proof above, the capacities are already multiples of p, an assumption we cannot make in the setting of Lemma 3.2.

Acknowledgement. Thanks to Aravind Srinivasan for useful discussions and to Michel Goemans for sending us a copy of [5]. We also wish to thank the two anonymous referees for their comments that substantially improved the presentation and led to a strengthening of Theorem 5.1.

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 minimal-cost flow network patterns. Technical Report ORO-15, Operational Research Office, Johns Hopkins University, Baltimore, MD, 1961.

[4] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:260–271, 1959.

[5] Y. Dinitz, N. Garg, and M. Goemans. On the single-source unsplittable flow problem. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pages 290–299, November 1998.

[6] 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.

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

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

[9] 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.

[10] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34:596–615, 1987.

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

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

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

[14] A.V. Goldberg and S. Rao. Beyond the flow decomposition barrier. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 2–11, 1997.

[15] 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.

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

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

[18] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[19] V. King, S. Rao, and R. Tarjan. A faster deterministic maximum flow algorithm. *Journal of Algorithms*, 17:447–474, 1994.

[20] 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.

[21] P. Klein, S. Rao, A. Agrawal, and R. Ravi. Approximation through multicommodity flow. *Combinatorica*, 15:187–202, 1995.

[22] J. M. Kleinberg. *Approximation algorithms for disjoint paths problems.* PhD thesis, MIT, Cambridge, MA, May 1996.

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

[24] 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.

[25] 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.

[26] S. G. Kolliopoulos. *Exact and Approximation Algorithms for Network Flow and Disjoint-Path Problems.* PhD thesis, Dartmouth College, Hanover, NH, August 1998.

[27] S. G. Kolliopoulos and C. Stein. Improved approximation algorithms for unsplittable flow problems. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 426–435, 1997.

[28] S. G. Kolliopoulos and C. Stein. Approximating disjoint-path problems using greedy algorithms and packing integer programs. In R. E. Bixby, E. A. Boyd, and R. Z. Rios-Mercado, editors, *Proceedings of the 6th Conference on Integer Programming and Combinatorial Optimization*, volume 1412 of *Lecture Notes in Computer Science*, pages 153–168. Springer-Verlag, 1998.

[29] T. Leighton, F. Makedon, S. Plotkin, C. Stein, É. Tardos, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems. *Journal of Computer and System Sciences*, 50:228–243, 1995.

[30] 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.

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

[32] J. B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41:338–350, 1993.

[33] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37:130–143, 1988.

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

[35] 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.

[36] 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.

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