

Incremental Flow

Jeff Hartline, Alexa Sharp

Department of Computer Science, Cornell University

Address: 4130 Upson Hall, Cornell University, Ithaca, NY 14853-7501, USA.

Phone: 607-255-7316, fax: 607-255-4428

Email: {jhartlin, asharp}@cs.cornell.edu

March 2005

Abstract

This paper defines an incremental version of the maximum flow problem. In this model, the capacities increase over time and the resulting solution is a sequence of flows that build on each other incrementally. Thus far, incremental problems considered in the literature have been built on NP-complete problems. To the best of our knowledge, our results are the first to find a polynomial time problem whose incremental version is NP-complete. We present approximation algorithms and hardness results for many versions of this problem, and comment on the relation to multicommodity flow.

keywords Incremental Flow, Incremental Problems, Incremental Networks, Maximum Flow, Multicommodity Flow, Concurrent Flow, Approximation Algorithms

1 Introduction

There has been recent interest in hierarchical versions of classic problems such as facility location [8], clustering [2], and bin packing [1]. These problems model situations in which there is a natural hierarchy of levels with different characteristics, such as local vs. wide-area networks or multi-level memory caches. Incremental variations of NP-complete problems contain their non-incremental versions as special cases and therefore remain NP-complete. It is interesting to ask whether incremental versions of polytime problems remain polytime, or whether the incremental structure alters the problem enough to increase its complexity.

In this paper we study incremental versions of a classic polytime problem, the maximum flow problem. It turns out that the incremental variant is sufficiently different from the original problem to warrant independent study, and bears an interesting relationship to multicommodity flow. We present various complexity results concerning incremental flow.

An incremental flow problem is defined on a directed network $G = (V, E)$ with source s , sink t , and a non-decreasing sequence of k capacity functions $c_i : E \rightarrow \mathbb{Q}, i \leq k$. The goal is to find k s - t flows f_i that optimize some objective function such that the flow f_i on an edge e is at most its capacity $c_i(e)$ but at least the amount sent by the previous flow, $f_{i-1}(e)$.

Let $|f_i|$ denote the amount of f_i flow sent from s to t . For the *maximum sum flow* problem, the objective is to maximize the sum of the flows: $\max \sum_i |f_i|$. For the *maximum ratio flow* problem, given demand d_i for each level i , the objective is to satisfy the maximum possible proportion of each level's demand: maximize $r = \min_i \frac{|f_i|}{d_i}$.

One may notice similarities between the incremental flow and multicommodity flow (MCF) problems. The input to MCF is also a directed network $G = (V, E)$ but with a single capacity function $c : E \rightarrow \mathbb{Q}$ and k source-sink pairs $(s_i, t_i), 1 \leq i \leq k$. The two predominant MCF objective functions (*maximum multicommodity flow* and *maximum concurrent flow*) correspond to the objectives stated above.

Some of the results of this paper are in agreement with results for MCF [9, 3, 7]. For example, an optimal solution to the incremental flow problem is not necessarily integral, in contrast to the standard flow problem. Fractional solutions can usually be found in polynomial time, whereas finding an integral solution is NP-complete even in the simplest cases. We consider both directed and undirected graphs, simple (unit-capacity edges) and general graphs, and integral and fractional flows. Our results are summarized in Figure 1.

Any flow problem in which constraints rise over time and rerouting flow carries an implicit cost would benefit from this type of incremental model. The canonical example of flight scheduling, where discontinuing a flight leg is undesirable, is one such application.

2 Incremental Flow

2.1 Incremental Flow Networks

An *incremental flow network* is a directed graph $G = (V, E)$ with a non-decreasing sequence of capacity functions $c_i : E \rightarrow \mathbb{Q}$, $1 \leq i \leq k$. In particular, $c_{i+1}(e) \geq c_i(e)$ for all edges e . We call each i a *level*. An *incremental s-t flow* is a sequence of flows $f_i : E \rightarrow \mathbb{R}$ such that f_i is an s-t flow with respect to capacity function c_i (flow constraint) and $f_{i+1}(e) \geq f_i(e)$ for all levels i (incremental constraint). We denote the flow at level i by f_i , its value by $|f_i|$, and a maximum flow at level i by f_i^* . An (a_1, a_2, \dots, a_k) flow is an incremental flow having value a_i at level i , and a (c_1, c_2, \dots, c_k) edge is an edge having capacity c_i at level i .

As an example, Figure 2(a) is a 2-level incremental flow network. We achieve a $(0, 2)$ flow in this network by sending no level 1 flow and two units of level 2 flow in parallel along paths $s \cdot u \cdot t$ and $s \cdot v \cdot t$. Alternatively, we achieve a $(1, 1)$ flow by sending one unit of level 1 flow along the only level 1 path $s \cdot u \cdot v \cdot t$ and no additional level 2 flow. No $(1, 2)$ flow exists because using uv in level 1 precludes us from sending any additional level 2 flow.

We also examine the incremental flow problem where the underlying network is undirected. In such cases we consider each edge $e = \{u, v\}$ to be two directed edges $\hat{e} = (u, v)$ and $\check{e} = (v, u)$. There are two possible interpretations of undirected networks. In *bidirectional* networks, one may send flow on both \hat{e} and \check{e} so long as the sum of the flows is less than $c(e)$. In *unidirectional* networks, either \hat{e} or \check{e} may carry flow but not both. These two constraints are equivalent in the integral unit-capacity case.

2.2 The Incremental Flow Problem

In contrast to the maxflow problem, where the single objective is to maximize flow into the sink, there are many possible objective functions in the incremental variation. Of the two mentioned in the introduction, this paper concentrates on the *max ratio problem*.

Given demands d_i , the max ratio problem finds a (a_1, a_2, \dots, a_k) flow maximizing $\min_i \{\frac{a_i}{d_i}\}$. We focus on the case $d_i = |f_i^*|$, but all the results generalize. Now the goal is to satisfy the maximum possible proportion of each level's maximum flow: determine the max ratio r^* such that $a_i \geq r^*|f_i^*|$ for all i . This ensures that adding an incremental aspect to the flow problem does not make any level arbitrarily worse than it would have been without the incremental restrictions. This is a standard metric for incremental problems [8].

It is well known that for the maxflow problem with integer edge capacities, the optimal fractional flow is no better than the optimal integral flow. Interestingly enough, this does not carry over to the incremental version of the problem. Similar phenomena have been observed in the multicommodity flow problem.

Figure 2(a) shows it is not always possible to obtain a ratio of 1 at each level. In fact, the best we can do with integral flow is $r^* = \frac{1}{2}$. However, if we allow fractional flows we can obtain $r^* = \frac{2}{3}$ with a $(\frac{2}{3}, \frac{4}{3})$ flow. We generalize this example in Figure 2(b) to show that the best attainable guarantee with integral flows is $r^* = O(\frac{1}{n})$, where n is the number of vertices in the network, n even. In this case $|f_1^*| = 1$ and $|f_2^*| = \frac{n}{2}$, but sending 1 unit of level 1 flow limits our level 2 flow to $|f_2| = 1 = \frac{2}{n}|f_2^*|$. With integral flow, this is the best ratio we can achieve. If we are willing to allow fractional flow, we can obtain $r^* = \frac{1}{2}(\frac{n}{n-1})$ on this network with a $(\frac{1}{2}(\frac{n}{n-1}), \frac{n}{4}(\frac{n}{n-1}))$ flow. These examples can be generalized further to show that for some k -level networks, $r^* = O(\frac{1}{k})$ even for fractional flow. We investigate the integral and fractional cases separately in the following two sections.

	directed	bidirectional [†] undirected	unidirectional [†] undirected
Integral	NPC, no $\omega(n^{-1})$ approx. (S2*) $O(n^{-1})$ approx. alg. (G)	NPC, no approx. $> \frac{1}{2}$ (S2*) no $\omega(n^{-1/3})$ approx. (S3*)	
Fractional	LP solvable (G), PTAS (G) $O(k^{-1})$ approx alg (G)	LP solvable (G) $O(k^{-1})$ approx alg (G)	NPC (G3) open (S2, G2, S3)

Figure 1: †: defined in Section 2.1, S: simple, G: general, 2: 2 levels, 3: 3 levels, *: unless $P = NP$.



Figure 2: Two example incremental flow networks. Throughout this paper we represent $(1,1)$ edges with solid lines and $(0,1)$ edges with dashed lines unless stated otherwise.

3 Integral Flows

We have observed that, in contrast to the maxflow problem, optimal incremental solutions are not always integral. Moreover, if we insist on integrality, then the incremental flow problem is NP-complete. Although this can be shown by reduction from MCF, we present a reduction from 3-SAT for the sake of obtaining stronger inapproximability results. We denote the decision version of the max ratio problem by r -ratio.

3.1 Hardness

Theorem 3.1 *The r -ratio problem is NP-hard for directed graphs with integral flow.*

The proof of Theorem 3.1 follows from a reduction from 3-SAT. We construct an instance of the directed flow problem in which $|f_1^*| = 1$ and $|f_2^*| = 2$. For this instance, a $(1,2)$ flow is a 1-ratio flow. We show that achieving such a flow is possible if and only if the 3-SAT instance has a satisfying assignment.

We are given an instance of 3-SAT with variables V and clauses C . Each clause is a set of three literals. The set of literals is the union of all positive literals v and all negative literals \bar{v} for $v \in V$. We denote each literal occurrence as a clause-literal pair $(c \in C, \ell \in c)$.

Literal Gadgets For every (c, ℓ) we create a literal gadget: an *in* vertex, an *out* vertex, and a $(1,1)$ directed link between these two vertices.

Clause Gadgets We create a clause gadget for every clause $c \in C$. Each clause gadget contains an *in* vertex, an *out* vertex, and the three literal gadgets of the form $(c, \ell \in c)$. These elements are linked together with $(1,1)$ edges as shown in Figure 3(a).

Variable Gadgets We create a variable gadget for every variable $v \in V$. Each variable gadget consists of an *in* vertex, an *out* vertex, and all literal gadgets of the form (c, v) or (c, \bar{v}) for any c . These elements are linked together as shown in Figure 3(b): positive literals on one side, negative literals on the other, and each side joined by $(0,1)$ edges.

Linkage We link the source, sink, and all clause gadgets together in series with $(1,1)$ edges. The same is done for the variable gadgets with $(0,1)$ edges. We call these edges *connector* edges. Finally, we use a $(0,1)$ *shortcut* edge to link the input node of the first variable gadget with the output node of the last clause gadget. Linkage is shown in Figure 3(c).

Lemma 3.2 *There is a satisfying assignment iff there exists a 1-ratio flow.*

[\Rightarrow] Given a satisfying assignment, we identify one true literal (c, ℓ) for each clause c . We route a unit of level 1 flow through all clause gadgets, passing through gadget c using literal (c, ℓ) . We route an extra unit of level two flow through all variable gadgets, using only false literals by passing through the positive (negative) side of gadget v if v is false (true).

[\Leftarrow] We first make the following observations concerning $(1,2)$ flows in our construction:

1. The level 1 component of any $(1,2)$ flow is a path including every $(1,1)$ connector and one literal from each clause gadget. This is because the three literals of any clause form a level 1 cut, as does any $(1,1)$ connector.
2. In any $(1,2)$ flow, level 2 flow at $(c, v).in$ (or $(c, \bar{v}).in$) must proceed to $(c, v).out$ ($(c, \bar{v}).out$) and back to the positive (negative) side of variable gadget v . The only other path would be to $c.out$, whose sole out edge is a $(1,1)$ connector already carrying level 1 flow (see observation 1).

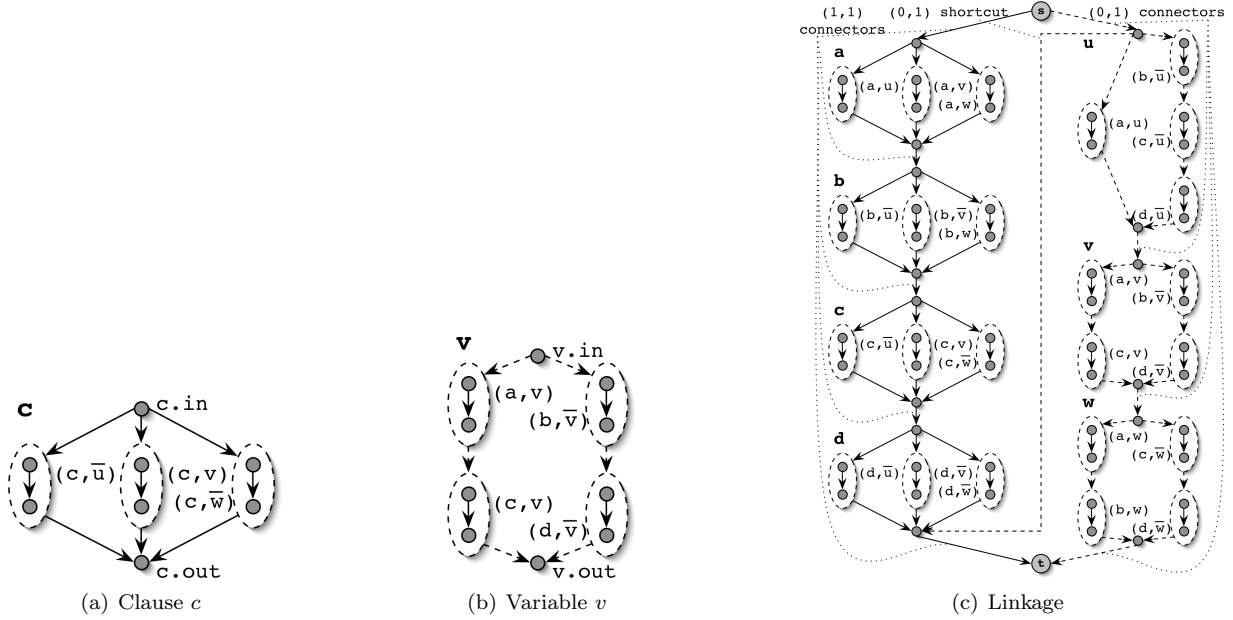


Figure 3: The directed flow construction for example $(u \vee v \vee w) \wedge (\bar{u} \vee \bar{v} \vee w) \wedge (\bar{u} \vee v \vee \bar{w}) \wedge (\bar{u} \vee \bar{v} \vee \bar{w})$. The clauses are denoted a, b, c, d . Literal gadgets appear inside labeled ovals. Note that each of the twelve literal gadgets appears once in a clause gadget and once in a variable gadget, and thus the seemingly separate source-sink paths in Figure 3(c) actually share many vertices.

3. In any $(1, 2)$ flow, level 2 flow at $v.in$ must proceed through either all (c, v) gadgets or all (c, \bar{v}) gadgets to $v.out$. This is because $v.in$ has two out edges: one to a sequence of all (c, v) gadgets and the other to a sequence of all (c, \bar{v}) gadgets. Once flow proceeds to one of these sequences, induction on observation 2 implies it must pass through every literal gadget in the targeted sequence and end at $v.out$.
4. The level 2 component of any $(1, 2)$ flow is a path passing through the positive or negative side of each variable gadget. The source vertex has only two out edges: a $(1, 1)$ connector to a clause gadget and a $(0, 1)$ connector to a sequence of all variable gadgets. By observation 1, the $(1, 1)$ connector is already used, forcing the flow to the first variable gadget. Repeated application of observation 3 implies that the flow proceeds through the positive or negative side of every variable gadget to the sink.

By observation 4, every variable v carries level 2 flow through one of its sides. We set v false if this flow passes through v 's positive side and true otherwise. Under this assignment, all false literals carry level 2 flow. By observation 1, one literal from each clause must carry level 1 flow (and not level 2 flow) and thus cannot be false. This assignment satisfies 3-SAT. \square

Theorem 3.1 follows from Lemma 3.2 and the polynomial nature of our reduction.

Theorem 3.3 *The r -ratio problem is NP-hard for undirected graphs with integral flow.*

Proof. Theorem 3.3 follows from a slight modification of the reduction used in the directed case. If we simply take our directed construction and remove directionality then observations 1-4 of Lemma 3.2 \Leftarrow no longer hold. We recover equivalent observations by replacing each clause gadget with the undirected clause gadget shown in Figure 4(a) and doubling the number of $(1, 1)$ connectors. A $(2, 3)$ flow is a 1-ratio flow in the resulting construction.

We state without proof the following variation of observation 1, which can be used to establish observations 2-4 for $(2, 3)$ flows in the undirected construction. The level 1 component of any $(2, 3)$ flow specifies two paths that together use every $(1, 1)$ connector and at least one literal from each clause. Furthermore, this flow disconnects the unused literals from each other as illustrated in Figure 4(b). The rest of the proof follows as before. \square

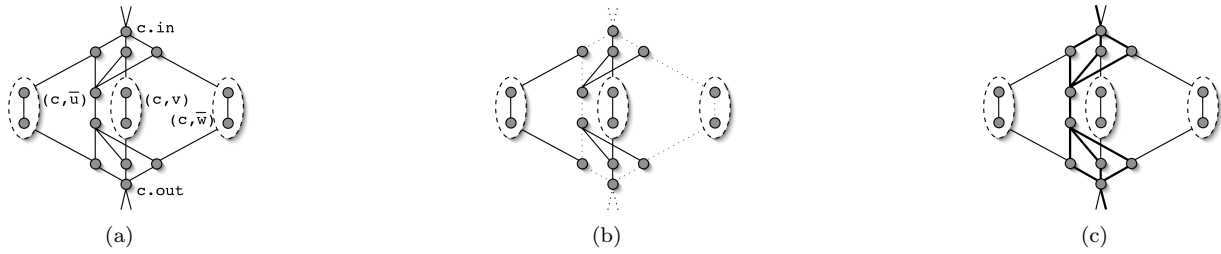


Figure 4: 4(a): An example undirected clause gadget for $c = \{\bar{u}, v, \bar{w}\}$. 4(b): One way to route two units of flow through this gadget. Any routing will disconnect the unused literal gadgets as demonstrated here by the dotted flow paths. 4(c): The 3-level clause gadget used in Theorem 3.6. Bold lines represent level 0 links.

3.2 Inapproximability

The following theorems show we have a tight $O(\frac{1}{n})$ -approximation algorithm for directed integral max ratio and a strong inapproximability result for the undirected case.

Theorem 3.4 *Directed integral max ratio is NP-hard to $g(n)$ -approximate for $g \in \omega(\frac{1}{n})$.*

Proof. Given an instance of 3-SAT, we compose an instance of the directed flow problem out of N copies of the network constructed for Theorem 3.1. These copies are joined as shown in Figure 5(a) to create a network of size $n = \Theta(SN)$, where S denotes the size of the original network. By the arguments of Lemma 3.2, there is a $(1, N + 1)$ flow iff there is a satisfying assignment. Furthermore, if there is no satisfying assignment then there is no (a_1, a_2) flow for $a_1 > 0$ and $a_2 > 1$. For any $g(n) \in \omega(\frac{1}{n})$, we can pick N sufficiently large to ensure that a $g(n)$ -approximation distinguishes between these cases. \square

Theorem 3.5 *There is a greedy $O(\frac{1}{n})$ -approximation for directed integral max ratio.*

Proof. Consider the algorithm that starts at level 1 and sequentially sends the maximum flow possible on each level, given the incremental constraint imposed by the previous flow. For any level i , let P be a single unit flow path of the greedily constructed f_i . Each edge of P could have been used by at most one unit flow path of each f_j^* , $j \geq i$, and since $\text{length}(P) < n$, the path P could have blocked at most $n - 1$ units of f_j^* 's flow. Therefore

$$|f_j| = |f_j^*| - \# \text{ paths } f_{j-1} \text{ blocks} \geq |f_j^*| - (n - 1)|f_{j-1}| \geq |f_j^*| - (n - 1)|f_j| \geq \frac{1}{n}|f_j^*|. \quad \square$$

Theorem 3.6 *Two-level undirected integral max ratio is NP-hard to α -approximate for $\alpha > \frac{1}{2}$. For three or more levels it is NP-hard to $g(n)$ -approximate for $g \in \omega(n^{-1/3})$.*

Proof. The two-level result follows from a modification of the reduction used in the directed case (Theorem 3.4) where we remove directionality as described in Theorem 3.3. We obtain the stronger 3-level approximation result by adding a level to our clause gadget as shown in Figure 4(c) and enhancing the construction as follows:

Given an instance of 3-SAT, we compose an instance of the undirected flow problem out of N^3 copies of the network constructed in Theorem 3.3 using multi-level clause gadgets. We join these copies as in Figure 5(b) to create a network of size $n = \Theta(SN^3)$, where S denotes the size of the original network. There is a $(1, N + 1, N^2 + N + 1)$ flow iff there is a satisfying assignment. Furthermore, if there is no satisfying assignment then there is no (a_0, a_1, a_2) flow for $a_0 > 0$, $a_1 > 1$, and $a_2 > N + 1$. For any $g(n) \in \omega(n^{-1/3})$, we can pick N sufficiently large to ensure that a $g(n)$ -approximation distinguishes between these cases. \square

4 Fractional Flows

Theorem 4.1 *Directed fractional and bidirectional fractional max ratio are polytime.*



Figure 5: Constructions for non-approximation results for integral max ratio. Both of these constructions paste together the clause and variable components of many copies of the constructions described in Section 3.1. We label clause and variable components of copy x as C_x and V_x , respectively.

Proof. We can formulate the directed fractional max ratio problem as a linear program (LP) in $O(mk)$ variables and constraints. The objective is to maximize r such that $f_i(e)$ and r are non-negative, $f_{i-1}(e) \leq f_i(e) \leq c_i(e)$, and

$$\sum_{e \text{ out of } v} f_i(e) - \sum_{e \text{ into } v} f_i(e) = 0 \quad \sum_{e \text{ out of } s} f_i(e) - \sum_{e \text{ into } s} f_i(e) \geq r \cdot d_i.$$

For the bidirectional case, we have the additional constraint that $f_i(\hat{e}) + f_i(\check{e}) \leq c_i(e)$. \square

Unfortunately, it is not known how to formulate an LP for the unidirectional case, due to the non-linear constraint $f_j(\hat{e}) > 0 \rightarrow f_i(\check{e}) = 0$ for all $i \geq j$. In fact, the following result states it is not possible to do so unless $P=NP$. We omit the proof for lack of space. See [6].

Theorem 4.2 *The r -ratio problem is NP-hard in the unidirectional fractional case.*

4.1 Approximation Algorithms

We can use LP techniques to find exact solutions for the directed and bidirectional max ratio problem, but these techniques are impractical. This leads us to search for faster algorithms to either solve the problem exactly, or even solve the problem approximately.

A polytime algorithm for the k -level fractional directed max ratio problem also solves the k -commodity concurrent flow problem. As there are no currently known combinatorial algorithms for the k -commodity concurrent flow problem for $k \geq 3$, we focused our efforts on the 2-level problem, unsuccessfully. Although we did not find an exact solution, we did find a straightforward $\frac{1}{k}$ -approximation for the directed and bidirectional cases. We start with a level 1 flow of f_1^*/k , and for each subsequent level i we define $f_i = f_{i-1} + f_i^*/k$.

In the directed case we can do better. There is a polytime approximation scheme (PTAS) for max ratio based on the techniques of [5, 4, 10]. For each level ℓ , the algorithm assigns a length $y_\ell(e)$ to each edge e . For an s - t path P , let $w_\ell(P)$ denote the cost of P , where the cost of edge e is $w_\ell(e) = \sum_{j \geq \ell} y_j(e)$. Let $\delta = (m/(1-\epsilon))^{-1/\epsilon}$.

Initially the algorithm sets $y_\ell(e) = \delta/kc_\ell(e)$ and $f_\ell \equiv 0$ for all levels ℓ . It then proceeds in phases; each phase is composed of k iterations. In the j^{th} iteration, the objective is to route $d_j - d_{j-1}$ units of level j flow from s to t . This is done in steps. In each step, a least-cost path P in the level j graph is computed using the cost function w_j . Let b be the minimum of the bottleneck capacity of P and the remaining demand. We send b units of flow along P and for each edge $e \in P$ and level $\ell \geq j$ we set $f_\ell(e) = f_\ell(e) + b$ and update the length function $y_\ell(e) = y_\ell(e)(1 + \epsilon \frac{b}{c_\ell(e)})$. The entire procedure stops when $\sum_{\ell=1}^k \sum_e c_\ell(e)y_\ell(e) \geq 1$. We scale the resulting incremental flow down by $\log_{1+\epsilon} \frac{1}{\delta}$.

Ideas similar to those of [5, 10] can be used to show that this procedure yields an efficient $(1-\epsilon)^3$ -approximate solution to the k -level incremental max ratio flow problem.

References

- [1] B. Codenotti, G. D. Marco, M. Leoncini, M. Montangero, and M. Santini. Approximation algorithms for a hierarchically structured bin packing problem. *Inf. Process. Lett.*, 89(5):215–221, 2004.
- [2] S. Dasgupta. Performance guarantees for hierarchical clustering. In *Proceedings of the 15th Annual Conference on Computational Learning Theory*, pages 351–363. Springer-Verlag, 2002.
- [3] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5(4):691–703, 1976.
- [4] L. K. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal on Discrete Math.*, 13(4):505–520, 2000.
- [5] N. Garg and J. Koenemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, page 300. IEEE Computer Society, 1998.
- [6] J. Hartline and A. Sharp. Hierarchical flow. Technical Report 2004-1965, Computer Science Department, Cornell University, September 2004.
- [7] T. Leighton, C. Stein, F. Makedon, É. Tardos, S. Plotkin, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 101–111. ACM Press, 1991.
- [8] C. G. Plaxton. Approximation algorithms for hierarchical location problems. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 40–49. ACM Press, 2003.
- [9] T.C.Hu. Multi-commodity network flows. *Operations Research*, 11(3):344–360, 1963.
- [10] K. D. Wayne and L. Fleischer. Faster approximation algorithms for generalized flow. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 981–982. Society for Industrial and Applied Mathematics, 1999.