Experimental Evaluation of Approximation Algorithms for the Minimum Cost Multiple-source Unsplittable Flow Problem

YASUHITO ASANO  
The University of Tokyo, Japan

Abstract

For the minimum cost multiple-source unsplittable flow problem, we propose a $2$-cost and $(c + 2)$-congestion approximation algorithm, where $c$ is the number of distinct sources. We also propose some heuristics based on a linear programming relaxation with randomized rounding and a greedy approach, and implement the proposed approximation algorithms and examine the quality of approximation achieved through computational experiments.

Keywords

Unsplittable flow, Congestion, Minimum-cost flow, Approximation algorithm.

1 Introduction

We consider the multiple-source unsplittable flow problem which is a generalization of the edge disjoint paths problem. Let $G = (V, E)$ be a directed or undirected graph with a positive capacity function $cap : E \to \mathbb{R}_+$. Let $T$ be the set of $k$ commodities with terminal pairs $(s_i, t_i)$ and positive demands $d_i (s_i, t_i \in V, s_i \neq t_i$ for each $i = 1, 2, ..., k)$. We call $s_i$ and $t_i$ source and sink of commodity $i$, respectively. Let $S = (s_1, s_2, ..., s_k)$, $T = (t_1, t_2, ..., t_k)$ and $D = (d_1, d_2, ..., d_k)$. Thus we can assume an input network is $N = (G, cap, S, T, D)$. Then the multiple-source unsplittable flow problem is to find a set of paths $P_1, P_2, ..., P_k$ such that each $P_i$ $(i = 1, 2, ..., k)$ is a path from $s_i$ to $t_i$ and each edge $e \in E$ satisfies $\sum_{j=1}^k d_j p_j(e) \leq cap(e)$, where $p_j(e) = 1$ if $P_j$ contains $e$ and $p_j(e) = 0$ otherwise. If such a set of paths $P_1, P_2, ..., P_k$ exists then we can send each commodity $i$ at least $d_i$ units from $s_i$ to $t_i$ along path $P_i$ using each edge within its capacity. Thus, the demand requirement for each commodity $i$ to be sent at least $d_i$ units is met. Let $f_i$ be the number of units of commodity $i$ sent along $P_i$ from $s_i$ to $t_i$ $(f_i(e) = f_i p_i(e)$, i.e., $f_i(e) = f_i$ if $e$ is in $P_i$ and $f_i(e) = 0$ otherwise) and set $f(e) = \sum_{i=1}^k f_i(e)$. Then we can consider $f$ is a flow satisfying the demand requirement ($f_i \geq d_i$ for each commodity $i$) and the capacity constraint ($f(e) \leq cap(e)$).
A flow $f$ is called an *unsplittable flow* if $f$ is decomposed into a set of paths $P_1, P_2, \ldots, P_k$ with $f_i(e) = f_i \geq d_i$ ($e$ in $P_i$) and $f_i(e) = 0$ (otherwise) and $f(e) = \sum_{i=1}^{k} f_i(e)$. An unsplittable flow is called *feasible* (infeasible, resp.) if it satisfies (violates, resp.) the capacity constraint. The *value* of an unsplittable flow $f$ is defined to be $\sum_{i=1}^{k} f_i$. For simplicity, we use $n = |V|$, $m = |E|$, $d_{\text{max}} = \max_{1 \leq i \leq k} d_i$, and $c_{\text{apop}} = \min_{e \in E} \text{cap}(e)$ and $c$ denotes the number of distinct vertices in $S$. We allow $s_i = s_j$ for some $i \neq j$ and $c \leq k$.

This problem has several variations. Kleinberg [3], [4] introduced the following problems.

**Feasibility:** Has a network $N = (G, \text{cap}, S, T, D)$ a feasible unsplittable flow?

**Maximum unsplittable flow:** Assuming that a network $N = (G, \text{cap}, S, T, D)$ has a feasible unsplittable flow, find a feasible unsplittable flow of maximum value.

**Minimum congestion flow:** Assume that a network $N = (G, \text{cap}, S, T, D)$ may have no feasible unsplittable flow. Thus, the capacity constraint may be violated for some edge $e$ or the demand request of some commodity may not be met. Since all the demand requests will be met if we make the edge capacities sufficiently large, define the *congestion* of an unsplittable flow $f$ to be $\max_{e \in E} f(e)/\text{cap}(e)$. Then, find an unsplittable flow minimizing the congestion.

**Minimum cost flow:** Assume that a network $N = (G, \text{cap}, S, T, D)$ has a feasible unsplittable flow and each edge $e \in E$ has a nonnegative cost $\text{cost}(e)$ (in this case the network will be denoted by $N = (G, \text{cap}, \text{cost}, S, T, D)$). The cost of an unsplittable flow $f$ is defined to be $\text{cost}(f) = \sum_{e \in E} \text{cost}(e)f(e)$. Then find an optimal unsplittable flow, i.e., feasible unsplittable flow of minimum cost.

The unsplittable flow problem in which all demands and capacities are equal to 1 exactly corresponds to the edge disjoint paths problem, which has been extensively studied since it is a fundamental model for the connection requests in networks, and, therefore, the unsplittable flow problem can be a fundamental model for the connection requests in high-bandwidth networks. The unsplittable flow problem is also closely related to the multicommodity flow problem in which each commodity can be sent along one or more paths to satisfy its demand.

If a given network $N = (G, \text{cap}, S, T, D)$ in the multiple-source unsplittable flow problem satisfies $c = 1$ (i.e., $s_1 = s_2 = \cdots = s_k$), the problem is called the *single-source unsplittable flow problem*. Kleinberg [3], [4] proved that the feasibility problem is NP-complete and thus all other problems described above are NP-hard even for the single-source case. Thus, approximation algorithms have been proposed. Kleinberg [3], [4] considered the single-source unsplittable flow problem and proposed a constant-factor approximation algorithms for several problems by assuming the feasibility and $d_{\text{max}} \leq c_{\text{apop}}$. For the minimum congestion flow problem, he proposed a 16-congestion approximation algorithm on directed graphs and an 8.25-congestion on undirected graphs. He also proposed a 10.473-congestion, 7.473-cost approximation algorithm for the minimum cost flow problem on undirected graphs. Kolliopoulos and Stein [6] improved these results and obtained a 3.23-congestion, 1.68-cost approximation algorithm based on
Kleinberg’s basic theorem. They also proposed a 3-congestion, 2-cost algorithm [8], [5]. These algorithms can be applied to the minimum congestion flow problem. Dinizt-Garg-Goemans proposed a 2-congestion algorithm, the best possible result for the minimum congestion flow problem [1].

On the other hand, the multiple-source unsplittable flow problem seems to be hard to approximate. Actually, we could find no result for the minimum cost unsplittable flow problem in the multiple-source case. For other problems, the following results have been known. For the maximum unsplittable flow problem, Srinivasan proposed approximation algorithms based on a linear programming relaxation (for short, LP relaxation) [10]. Kolliopoulos-Stein proposed several greedy algorithms for the edge disjoint paths problem [7]. For the edge disjoint paths, the maximum integral multicommodity flow and the maximum unsplittable flow problems, Guruswani, et al. [2] obtained approximability results and approximation algorithms on the basis of an LP relaxation with randomized rounding method and the greedy method.

In this paper, by assuming the feasibility and $d_{\text{max}} \leq cap_{\text{min}}$, we consider the minimum cost multiple-source unsplittable flow problem. To construct approximation algorithms, we use three approaches. First, by extending the Kolliopoulos-Stein 3-congestion, 2-cost approximation algorithm for the minimum cost single-source unsplittable flow problem in [8], [5], we propose a $(c + 2)$-congestion, 2-cost approximation algorithm for the minimum cost multiple-source unsplittable flow problem ($c$ is the number of distinct vertices in $S$ and $c \leq k$) in Section 3. Then, we propose heuristics in Section 4 based on an LP relaxation with randomized rounding (for short, LPRR) developed by Raghavan and Thompson, and propose other heuristics based on a greedy method in Section 5. Finally we implement the proposed approximation algorithms including heuristics and evaluate their performance by computational experiments in Section 6.

2 Basic notions and Kleinberg’s basic theorem

As mentioned above, a polynomial time approximation algorithm for the minimum cost multiple-source unsplittable flow problem is called an $\alpha$-congestion, $\beta$-cost approximation algorithm if it always produces an unsplittable flow $f$ on $N = (G, cap, cost, S, T, D)$ with cost at most $\beta$ times the cost of an optimal unsplittable flow and $f(e) = \sum_{j=1}^{k} f_i(e) \leq \alpha \text{cap}(e)$ holds for each edge $e$ of $G$ (thus $\max_{e \in E} f(e)/\text{cap}(e) \leq \alpha$). The minimum cost multiple-source unsplittable flow problem can be formulated as an integer programming problem. If a variable $p_i(e)$ indicating that commodity $i$ is sent through edge $e$ is allowed to take on any value between $[0, 1]$, then we have an LP formulation which corresponds to a relaxation of the minimum cost multiple-source unsplittable flow problem. If the relaxation has a feasible solution, we call it a fractional feasible solution or a fractional feasible flow to the problem. Since the minimum cost multiple-source unsplittable flow
problem is NP-hard and it is difficult to actually obtain an optimal solution, we use a fractional optimal solution to the LP relaxation to estimate an approximate solution. Let \( \text{cost}(\text{opt}) \), \( \text{cost}(\text{frac}) \) and \( \text{cost}(\text{app}) \) denote the costs of an optimal solution, the fractional solution and an approximate solution to the problem, respectively. Then \( \frac{\text{cost}(\text{app})}{\text{cost}(\text{opt})} \leq \frac{\text{cost}(\text{app})}{\text{cost}(\text{frac})} \) holds and thus we can use the value \( \frac{\text{cost}(\text{app})}{\text{cost}(\text{frac})} \) as an upper bound of the approximation ratio.

In this section, we describe Kleinberg’s basic theorem and its corollary [3], which played a central role in approximation algorithms for the single-source unsplittable flow problem [6], [8]. Our \((c + 2)\)-congestion, 2-cost algorithm will also utilize it.

**Theorem 1 (Kleinberg’s basic theorem) [3]** If all demands \( d_i \) are equal to a common value \( d \), all capacities are multiples of \( d \), and there is a fractional feasible flow in \( N = (G, \text{cap}, S, T, D) \) with \( s_1 = s_2 = \cdots = s_k \), then there is a maximum fractional flow in \( N \) that is a feasible unsplittable flow and it can be found in polynomial time.

**Corollary 1** [3] If all demands \( d_i \) are equal to a common value \( d \), all capacities are multiples of \( d \), and there is a fractional feasible flow \( f \) with cost \( \text{cost}(f) \) in \( N = (G, \text{cap}, \text{cost}, S, T, D) \) with \( s_1 = s_2 = \cdots = s_k \), then we can find a feasible unsplittable flow \( g \) in \( N \) with total cost \( \text{cost}(g) \leq \text{cost}(f) \) in polynomial time.

### 3 \((c + 2)\)-congestion, 2-cost algorithm

In this section we present a \((c + 2)\)-congestion, 2-cost approximation algorithm for the minimum cost multiple-source unsplittable flow problem by extending the 3-congestion, 2-cost algorithm for the single-source unsplittable flow problem in [8] and [5], where \( c \) denotes the number of the distinct vertices in the sources. We assume \( d_{\text{max}} \leq c \) and the fractional feasibility of an input network \( N = (G, \text{cap}, \text{cost}, S, T, D) \) as Kleinberg did. First, we normalize demands and capacities of the input network and assume, without loss of generality, all demands are in \((0, 1]\) and all capacities are in \([1, \infty)\). Our \((c + 2)\)-congestion, 2-cost approximation algorithm is based on the following lemma and corollary which can be obtained by a direct application of Kleinberg’s basic theorem.

**Lemma 1** [8] Let \( N = (G, \text{cap}, \text{cost}, S, T, D) \) be a network for the minimum congestion single-source unsplittable flow problem satisfying (a) all demands are equal to \( 1/2^x \) (\( x \) is a positive integer), (b) capacities are multiples of \( 1/2^{x+1} \), (c) it has a fractional feasible solution. Then there is a fractional feasible solution \( f \) with granularity \( 1/2^{x+1} \) (i.e. \( f(e) \) is a multiple of \( 1/2^{x+1} \) for each edge \( e \)). Furthermore, we can find, in polynomial time, an unsplittable flow \( g \) such that \( g(e) \) is at most \( f(e) + 1/2^{x+1} \) for each edge \( e \), by adding at most \( 1/2^{x+1} \) to each capacity.
Corollary 2 [5] In the above lemma, we can find an unsplittable flow $g'$ which satisfies $g'(e) \leq f(e) + 1/2^{x+1}$ for each $e$ and $\text{cost}(g') \leq \text{cost}(f)$, by adding at most $1/2^{x+1}$ to each capacity.

By using Corollary 2 we can show that the 3-congestion algorithm in [8] for the minimum congestion single-source unsplittable flow problem becomes a 3-congestion, 2-cost algorithm in [5] for the minimum cost single-source unsplittable flow problem. Now we are ready to describe our algorithm.

Our $(c + 2)$-congestion, 2-cost algorithm

1. Obtain a fractional feasible solution $f$ of $N = (G, \text{cap}, \text{cost}, S, T, D)$.

2. Decompose $f$ into $f^1, \ldots, f^c$ corresponding to the distinct sources, and for each $f^j$, define the $j$-th network $N^j = (G^j, \text{cap}^j, \text{cost}^j, S^j, T^j, D^j)$ as follows ($1 \leq j \leq c$): $G^j$ consists of $V$ and $E^j = \{e \in E, f^j(e) > 0\}$. For each $e \in E^j$, $\text{cap}^j(e) = f^j(e)$ and $\text{cost}^j(e) = \text{cost}(e)$. $S^j = (s^j_1, \ldots, s^j_{k_j})$ with $s^j_1 = \cdots = s^j_{k_j}$, $T^j = (t^j_1, \ldots, t^j_{k_j})$, $D^j = (d^j_1, \ldots, d^j_{k_j})$, $T^j = \{(s^j_{t^j_1}, \ldots, s^j_{t^j_{k_j}})\}$ ($k_1 + \cdots + k_c = k$).

3. For $j := 1$ to $c$ do the following:

   (a) For each demand $d^j_i$, define the new demands $d'^j_i$ to be the smallest power of $1/2$ not less than $d^j_i$, and set $\alpha^j_i = d'^j_i/d^j_i$ ($1 \leq \alpha^j_i < 2$).

   (b) Let $d'^j_{\text{min}}$ be the smallest new demand and set $\lambda_j = -\log_2 d'^j_{\text{min}}$ (thus, $d'^j_{\text{min}} = \min_{1 \leq i \leq k_j} d'^j_i$ and $d'^j_{\text{min}} = 1/2^{\lambda_j}$).

   (c) Update each capacity $\text{cap}^j(e)$ to the smallest multiple of $1/2^{\lambda_j}$ not less than $\sum_{1 \leq i \leq k_j} f^j_i(e)\alpha^j_i$.

   (d) Consider the virtual commodities as in [8], where every virtual commodity has $1/2^{\lambda_j}$ demand. That is, split each commodity $i$ in $T^j$ with $d^j_i = 1/2^{x}$ into $2^{\lambda_j-x}$ virtual commodities.

   (e) By using Corollary 1, obtain an unsplittable flow $f'^j$ with granularity $1/2^{\lambda_j}$ for the virtual commodities. Now every commodity $i$ with $d'^j_i = 1/2^{\lambda_j}$ has no longer to be considered since the commodity has exactly one path in $f'^j$ as desired.

   (f) Repeat the following until an unsplittable flow $g^j$ for $T^j$ with the new demands is obtained, that is, until the granularity becomes 1.

      i. Let $1/2^{t_j}$ be the current granularity and consider new virtual commodities where every virtual commodity has $1/2^{t_j-1}$ demand.

      ii. Apply Corollary 2 as in [8] to obtain an unsplittable flow for the current virtual commodities with granularity $1/2^{t_j-1}$. Now every commodity $i$ with $d'^j_i = 1/2^{t_j-1}$ has no longer to be considered.

4. By summing up $g^j$ for all $1 \leq j \leq c$, obtain an unsplittable flow $g$. 
5. By decreasing the demands of $g$ to the original demands, obtain $h$, an approximate solution of the given instance $N = (G, \text{cap}, \text{cost}, S, T, D)$.

**Theorem 2** Let an instance $N = (G, \text{cap}, \text{cost}, S, T, D)$ of the minimum cost unsplittable flow problem with $c$ distinct sources have a fractional feasible solution and satisfy $d_{\text{max}} \leq \text{cap}_{\text{min}}$. Then our algorithm above finds a $(c + 2)$-congestion and 2-cost approximate solution $h$ in polynomial time.

**Proof.** We can observe that $h$ is an approximate solution of $N$, since before step 3(e) the virtual commodities and the capacities satisfy the condition in Corollary 1 and before step 3(f) they satisfy the condition in Corollary 2 similarly, and in step 3(f) we can use the same argument in the proof of justification of Kolliopoulos-Stein’s 3-congestion, 2-cost algorithm.

For the congestion, we can observe that for any edge $e$, $f^{i'}(e) \leq 2f^{i}(e) + 1/2^\lambda_j$ ($1 \leq j \leq c$) since we can update the capacities at step 3(c) by multiplying $\text{cap}^j(e)$ by at most $\max_{1 \leq j \leq k} \alpha_i < 2$ and adding to it at most $1/2^\lambda_j$. From the definition of $\lambda_j$ and $d^{i'}_j$, the smallest original demand satisfies the following: $1/2^{\lambda_j+1} < \min_{i \in T_j} d_i \leq 1/2^{\lambda_j}$. At step 3(f), as in Kolliopoulos-Stein [8], we add at most $\sum_{v=1}^{\lambda_j} 1/2^v$ to capacities $\text{cap}^j(e)$ for each $j$. Therefore, we have $g^{i}(e) \leq 2f^{i}(e) + 1/2^{\lambda_j} + \sum_{v=1}^{\lambda_j} 1/2^v = 2f^{i}(e) + 1$. Since we obtain $g$ by summing up $g^j$ for $1 \leq j \leq c$, we have, for each edge $e$, $h(e) \leq g(e) \leq 2f(e) + c$ and $f(e) \leq \text{cap}(e) \geq 1$ by the assumption, and therefore we obtain $(c + 2)$-congestion.

For the cost, only step 3(a) can increase costs since using Corollaries 1 and 2 does not increase costs. Thus, it follows that $\text{cost}(h) \leq \text{cost}(g) = \sum_{i=1}^{c} \text{cost}(g^j) \leq \sum_{j=1}^{c} 2 \cdot \text{cost}(f^j) \leq 2 \cdot \text{cost}(f)$.

For the time complexity, we can observe that the algorithm can be done in polynomial time by using the same argument in the proof of polynomiality of Kolliopoulos-Stein’s 3-congestion, 2-cost algorithm. □

Furthermore, we can generalize the 3.23-congestion, 1.68-cost algorithm of Kolliopoulos-Stein [6] to the multiple-source problem.

**Corollary 3** Given an instance of the minimum cost unsplittable flow problem with $c$ sources satisfying $d_{\text{max}} \leq \text{cap}_{\text{min}}$ and the fractional feasibility, we can find a $3.23c$-congestion and 1.68-cost approximate solution in polynomial time.

The proof can be obtained directly from the results of Kolliopoulos-Stein [6], and therefore we omit it. Note that we cannot generalize the 2-congestion algorithm of Dinitz-Garg-Goemans [1] to the minimum cost flow problem since the algorithm seems hard to bound the cost.
4 LP relaxation with randomized rounding

Raghavan and Thompson [9] proposed an LP relaxation with randomized rounding (for short, LPRR) method, which can be applied to several optimization problems formulated in terms of IP (integer programming) problems. Kleinberg [3] applies the method to the minimum congestion multiple-source unsplittable flow problem and shows that the method achieves $O(\log m)$-congestion w.h.p. (with high probability) on arbitrary graphs. We will apply the method to the minimum cost multiple-source unsplittable flow problem as follows.

The main idea of the LPRR method consists of following two phases. First, we relax a problem formulated by an IP problem to an LP problem and we obtain a solution to the relaxed LP problem which consists of one or more paths for each commodity. Let $f$ denote a fractional solution, and let us call this phase a relaxing phase. Next, we construct an unsplittable flow by randomly selecting a path for each commodity from the paths in the fractional solution. Let us call this phase a randomized rounding phase. More precisely, let $P_{i1}, \ldots, P_{i\rho(i)}$ denote the paths of the fractional solution for commodity $i$, where $\rho(i)$ denotes the number of the paths, and then, we select exactly one path $P_{ij}$ with probability $f(P_{ij})/d_i$ for each commodity $i$. Once we have selected $P_{ij}$ for $i$, we set a new flow $f'$ as $f'(P_{ij}) = d_i$ and $f'(P_{ih}) = 0$ for $1 \leq h \leq \rho(i)$, $h \neq j$. The above operation is called randomized rounding, and after the randomized rounding a desired unsplittable flow is obtained.

This LPRR method achieves the following bound, which can be shown using Chernoff’s bound.

**Corollary 4** Let $m$ and $k$ be sufficiently large. Then the above LPRR method satisfies the following (i)-(iii):

(i) $2\ln m$-congestion w.h.p. (with high probability) when $d_{\max} \leq cap_{\min}$.

(ii) $\varepsilon^2/e$-congestion w.h.p. when $cap_{\min} \geq \varepsilon\ln m$ ($\varepsilon > 0$) ($e$ denotes the base of the natural logarithm).

(iii) $2\ln k$-cost for arbitrary capacities and costs for the minimum cost multiple-source unsplittable flow problem.

We propose the following heuristics (in algorithms LP2, LP3) in the LPRR method to test them together with LP1, the above naive one.

**LP2** : When we relax the given problem to the LP problem, we use new cost $\text{cost}'(e) = \text{cost}(e)/\text{cap}(e)$ instead of the original costs.

**LP3** : When we relax the given problem to the LP problem, we use new cost $\text{cost}'(e) = \text{cost}(e)/\text{cap}(e)^2$ instead of the original costs.

The aim of the heuristics is to give a priority to edges with larger capacities. We cannot analyze the performance of the heuristics theoretically, and therefore we estimate them by computational experiments. See Section 6.
5 Greedy algorithms

The greedy method is used by Kleinberg [3] and Kolliopoulos-Stein [7] in the approximation algorithms for the edge disjoint paths problem, and by Guruswani, et al. [2] in the algorithms for the maximum unsplittable flow problem and the maximum integral multicommodity flow problem, although it produces no theoretical bounds of approximation ratios better than $\Omega(1/\sqrt{m})$.

In this section, we propose greedy algorithms for the minimum congestion and the minimum cost flow problems for multiple-source unsplittable flows. The following algorithm also assumes that $d_{\text{max}} \leq cap_{\text{min}}$ and fractional feasibility as before (these restrictions can be removed with a few modifications).

Algorithm Greedy

1. Sort $d_1, d_2, \ldots, d_k$ and assume $d_1 \geq d_2 \geq \cdots \geq d_k$ and make a commodity list $L = (1, 2, 3, \ldots, k)$ (1 is the top element of $L$).
2. Repeat the following until a path for every commodity is found.
   (a) Make a copy $G'$ of $G$. Define the capacity function $\text{cap}'$ of $G'$ to be $\text{cap}'(e) = \text{cap}(e)$ for each $e$.
   (b) Set $i$ to the top element of the list $L$, and repeat the following until $L$ becomes empty.
      i. Find a shortest path from $s_i$ to $t_i$ on the graph $G'$ on the basis of a length function $\text{length}(e)$ described below. Note that we use only edges $E' = \{e \mid \text{cap}'(e) \geq d_i\} \subset E$.
      ii. If a shortest path $P$ is found, set $f(P) = d_i$ and for any edge $e \in P$, decrease $\text{cap}'(e)$ by $d_i$. Moreover, delete $i$ from $L$ and go to (b).
      iii. Otherwise, finish the scanning of $L$ and go back to 2.

For the above algorithm, we use the following length functions. The bold name indicates the greedy algorithm using the right-hand length function. Note that each algorithm uses exactly one length function.

Length Functions

- gr10 : Set $\text{length}(e) = 1/\text{cap}'(e)$.
- gr20 : Set $\text{length}(e) = \text{cost}(e)$.
- gr30 : Set $\text{length}(e) = \text{cost}(e)/\text{cap}'(e)$.
- gr40 : Set $\text{length}(e) = \text{cost}(e)/(\text{cap}'(e)^2)$.
- gr50 : Set $\text{length}(e) = 1$.

We could not analyze the performance of the above algorithms theoretically, and thus we will estimate the performance of the algorithms by computational experiments. See Section 6.
6 Experimental results

For experiments, we use the following environments. We write programs by using the programming language C++, and the compiler GNU g++, version 2.95.2 (option : -g -Wall), and SIMPLE to solve LP, which is an original C++ library using the internal point method in NUOPT developed by Suuri-system corporation in Japan. We use a machine with UltraSPARCII 400MHz, and 4GB main memory.

We describe how we construct instances for the experiments. For each capacity, we generate a uniform random number in the interval \([cap_{min}, cap_{max}]\). For every cost we generate it in the same way as the capacities, and multiply it by the capacity of the edge to prevent that algorithms which do not consider the cost perform well on cost. For every demand we generate it in the interval \([d_{min}, d_{max}]\) in the same way as the capacities (note that all demands, capacities and costs are integer). For graphs, we use undirected random graphs. For edges of random graphs, given \(n, m\), we assume edge \((i, i+1)\) exists. For each pair \((i, j)\) \((i + 1 < j)\) we give an equal probability to exist for such an edge. For each commodity we select uniformly at random a source and a sink. Since we assume that the problem is fractional feasible, once we solve the maximum flow problem by LP and we remove unroutable demands from the given demands.

Now we present results of the experiments. Figure 1 shows congestions and costs of the algorithms respectively, for instances with \(d_{max} = d_{min} = 1, k = 80, 1\leq cost(e)\leq 100\) and capacities changing from \(cap_{max} = cap_{min} = 1\) to \(cap_{max} = cap_{min} = 5\). Similarly, Figure 2 shows results for instances with \(d_{max} = 5, d_{min} = 1, k = 150, 1 \leq cost(e) \leq 100\) and capacities changing from \(5 \leq cap(e) \leq 10\) to \(20 \leq cap(e) \leq 25\). Figure 3 shows results for instances with \(d_{max} = 5, d_{min} = 1, 15 \leq cap(e) \leq 20, 1 \leq cost(e) \leq 100\) and the number of commodities \(k\) changing from 50 to 250. Note that the maximum number of the variables in the figure is 150000 (\(m = 300\), one undirected edge corresponds to two directed edges, and \(k = 250, 300 \times 2 \times 250 = 150000\)).

![Figure 1: Congestion and cost (demand=1, k = 80)](image)

As a result, we can observe the following. First, for the running time of the algorithms, the greedy algorithms are fast and take at most 1 second for each experiment. On the other hand, the LPRR algorithms and our \((c + 2)\)-cost 2-cost algorithm are very slow since a part to solve LP takes much time, about
one of the greedy algorithms. The order of performance among the LPRR algorithms is 1 to 2.14 times worse compared with the best performance of the greedy algorithms. The best algorithms show the performance better than the theoretical bound mentioned above, but worse than the best performance of the greedy algorithms. The best algorithms perform very nicely and achieve the cost near to the optimal fractional cost, the cost of the 500 seconds for $k = 80$, about 1000 seconds for $k = 150$, about 1700 seconds for $k = 200$. The rounding phase in the LPRR and the remaining steps in our $(c + 2)$-congestion algorithm are relatively fast than the part of LP.

Next, for the congestion, the algorithm \textbf{gr10} performs best and \textbf{gr50} performs second best (the averages are 1.13 and 1.18, respectively). The worst congestion of the greedy algorithms is at most 3 in these experiments. The LPRR algorithms show the performance better than the theoretical bound mentioned above, but worse than the best performance of the greedy algorithms. The best performance of the LPRR algorithms is 1 to 2.14 times worse compared with the best one of the greedy algorithms. The order of performance among the LPRR algorithms \textbf{LP1}, \textbf{LP2}, \textbf{LP3} is not fixed, and therefore, we cannot assure that the heuristics we proposed (the modifications in the relaxation phase) are effective. The worst congestion of the LPRR algorithms is at most 4 in these experiments. Our $(c + 2)$-congestion algorithm performs much better than the theoretical bound $(c + 2)$ (note that in the experiments, $c \geq 30$ and $c \geq k/3$), and slightly better than the LPRR algorithms for most instances, but worse than \textbf{gr10} and \textbf{gr50}. The worst congestion of the $(c + 2)$-congestion algorithms is at most 3.2 in these experiments. If capacities are fixed, all algorithms perform better when $k$ is smaller, and if $k$ is fixed, they perform better when capacities are larger. The results came up to our expectation, since capacities are the larger and $k$ is the smaller, the constraints will be the looser.

Finally, for the cost, \textbf{LP1}, \textbf{LP2}, \textbf{LP3} and our 2-cost algorithm perform very nicely and achieve the cost near to the optimal fractional cost, the cost of the
minimum cost fractional flow. Even in the worst case, LP1, LP2 show only 1.07 times of the optimal fractional cost, LP3 shows only 1.13 times, and our 2-cost algorithm shows only 1.06 times. The order of performance among the LPRR algorithms is LP1, LP2, LP3, as expected, since the heuristics used in LP2, LP3 are disadvantageous modifications for the cost. The greedy algorithms show 8 to 70% worse than the LPRR algorithms. gr20, gr30 perform comparatively well (even in the worst case, 1.48). gr10, gr50, which perform best for the congestion, show the second worst cost, 1.71 and 1.68 in the worst case respectively, since the length functions used in them do not consider the cost.

Acknowledgement: The author wishes to thank Prof. H. Imai for valuable discussions and suggestions.

References

Yasuhito Asano is with the Department of Information Science at the University of Tokyo, Hongo 7-3-1, Bunkyo-ku, Japan (ZIP: 113-0033). E-mail: asano@is.s.u-tokyo.ac.jp