The Multiroute Maximum Flow Problem Revisited

Donglei Du[*] R. Chandrasekaran[†]

January 20, 2005

Abstract

We are given a directed network $G = (V, A, u)$ with vertex set V, arc set A, a source vertex $s \in V$, a destination vertex $t \in V$, a finite capacity vector $u = \{u_{ij}\}_{ij \in A}$, and a positive integer $m \in Z_+$. The multiroute maximum flow problem (m-MFP) generalizes the ordinary maximum flow problem by seeking a maximum flow from s to t subject to not only the regular flow conservation constraints at the vertices (except s and t) and the flow capacity constraints at the arcs, but also the extra constraints that any flow must be routed along m arc-disjoint s-t paths. In this paper, we devise two new combinatorial algorithms for m-MFP. One is based on Newton's method and another is based on augmenting-path technique. We also show how the Newton-based algorithm unifies two existing algorithms, and how the augmenting-path algorithm is strongly polynomial for case $m = 2$.

Keywords Newton's method, augmenting-path, multiroute flow, parametric flow

1 Introduction

The Problem: In the ordinary (single commodity) maximum flow problem, the objective is to send the maximum amount of a single commodity from a single source to a single destination through a network without violating the flow conservation constraints at the vertices (except the source and the destination) and the capacity constraints at the arcs. Since its initiation by Ford and Fulkerson [12], the maximum flow problem has been receiving extensive treatments in the literature because of its vast applications. The reader is referred to the book by Ahuja et al. [2] (Chapters 6-8) for further information on this problem.

There has been recent interest in extending the ordinary maximum flow problem to the *multi-route* version in order to provide fault-tolerance against arc failures in the network ([1, 4, 5, 7, 15,

[*]Corresponding author: Faculty of Administration, University of New Brunswick, P.O. Box 4400, Fredericton, NB, Canada, E3B 5V4, e-mail: ddu@unb.ca. Research supported in part by NSERC grant 283103

[†]Department of Computer Sciences, University of Texas at Dallas, Richardson, TX 75083, U.S.A., e-mail: chandra@utdallas.edu

17], et al.). Consider, for example, a communication network whose arcs are liable to physical failures. In the ordinary maximum flow problem, any generic unit of flow is routed along a single path from the source to the destination, and this unit is lost whenever an arc on the path fails. Therefore this routing method is vulnerable even to a single arc failure. A more robust strategy is to have equal flow in multiple disjoint routes (multiroute), i.e., m copies of the same generic unit flow in m disjoint paths (routes) from the source to the destination, where $m \in Z_+$. This guarantees that even in the case of $m-1$ arc failures, at least one route will survive, and hence one copy of the generic unit can be routed to the destination. Further applications can be found in [1].

We now formally define the problem. We introduce some notations first. For any finite set S, let $|S|$ denote the cardinality of S, and let $R_+^{|S|}$ denote the set of nonnegative real column vectors with dimension $|S|$.

Definition 1.1 ***(Multiroute Maximum Flow Problem*** *(m-MFP)): Given a directed network $G = (V, A, u)$ with vertex set V, arc set A, a source vertex $s \in V$, a destination vertex $t \in V : t \neq s$, and an arc capacity vector $u = \{u_{ij}\}_{(i,j)\in A} \in R_+^{|A|}$, let $m \in Z_+$ be any positive integer.*

(1) *An m-**route** from s to t consists of m arc-disjoint s-t paths. Arc $(i,j) \in A$ is said to be contained in an m-route if it is contained in some s-t path therein.*

(2) *An m-**route path-flow** from s to t is an assignment of nonnegative weights to m-routes.*

(3) *The **value** of the an m-route path-flow is the sum of the weights assigned.*

The problem is to find an m-route path-flow of maximum value.

Denote $\mathcal{P}$ to be the set of all m-routes from s to t, and denote further $\mathcal{P}_{ij}$ to be the set of m-routes that contain a given arc $(i,j) \in A$. Define vector variable $x = \{x_p\}_{p\in\mathcal{P}} \in R_+^{|\mathcal{P}|}$, where each x_p is the weight assigned to m-route $p \in \mathcal{P}$. Then we can formulate m-MFP as a linear program in the *path-flow* form:

$$\max \sum_{p\in\mathcal{P}} x_p \tag{1}$$

$$\sum_{p\in\mathcal{P}_{ij}} x_p \leq u_{ij}, \qquad \forall (i,j) \in A \tag{2}$$

$$x(p) \geq 0, \qquad \forall p \in \mathcal{P} \tag{3}$$

Although linear program (1)-(3) could have exponential number of variables, it is still solvable in weakly polynomial time based on the *Ellipsoid* algorithm [13], although it is not an efficient procedure in practice. The idea is to provide a polynomial-time subroutine that solves the *separation* problem of the linear program, that is, given a solution, either proves the solution is feasible, or else finds a violated constraint. In our case, we solve the dual program of (1)-(3). The separation problem of the dual program is just the minimum cost m-route problem (where the cost of each arc is the dual variable associated with that arc); we only need to verify that the minimum cost m-route between s and t is at least 1; this problem is solvable in both weakly and strongly polynomial time (e.g., it can be viewed as a unit-capacity minimum cost flow problem, for which many efficient algorithms are available; see Chapters 9-11 of [2]).

An equivalent arc-flow linear programming formulation lends itself to more efficient algorithms, as we introduce next. Let $\delta_i^- = \{j : (j,i) \in A\}$ and $\delta_i^+ = \{j : (i,j) \in A\}$ denote the sets of in-neighbors and out-neighbors of vertex $i \in V$, respectively.

Definition 1.2 *An m-**route arc-flow** $f \in R_+^{|A|}$ of value F/m satisfies the following flow conservation constraints (4), capacity constraints (5), as well as the extra capacity constraints (6):*

$$\sum_{j\in\delta_i^+} f_{ij} - \sum_{j\in\delta_i^-} f_{ji} = \begin{cases} F, & i = s \\ 0, & \forall i \neq s,\ t \\ -F, & i = t \end{cases} \tag{4}$$

$$0 \leq f_{ij} \leq u_{ij} \qquad \forall (i,j) \in A \tag{5}$$

$$0 \leq f_{ij} \leq \frac{F}{m} \qquad \forall (i,j) \in A \tag{6}$$

One advantage of an arc-flow formulation over a path-flow formulation is that its specification involves polynomial number of variables. Note that 1-MFP (i.e., $m = 1$) is exactly the ordinary maximum flow problem. By specifying $m = 1$ in Definition 1.1 and 1.2, we can represent any regular 1-route flow using two forms: the path-flow and the arc-flow. It is well-known that these two forms are equivalent in the following sense. Given any 1-route path-flow, one can construct an 1-route arc-flow with the same value in $O(|A|)$ time by summing up all the flow across any arc, and the converse can also be done using the well-know path-decomposition in $O(|V||A|)$ time (see e.g. [2], Pages 79-83). We have a similar relationship between m-route path-flows and m-route arc-flows. On one hand, given any m-route path-flow $x \in R_+^{|\mathcal{P}|}$, it is easy to deduce an m-route arc-flow $f \in R_+^{|A|}$ with the same value by taking $f_{ij} = \sum_{p\in\mathcal{P}_{ij}} x_p$, for any arc $(i,j) \in A$. However, the other direction is much more involved, and it is first established by Kishimoto [17].

Theorem 1.3 *(Kishimoto [17]) (i) Given any m-route arc-flow $f \in R_+^{|A|}$ of value F/m, we can construct an m-route path-flow of the same value; (ii) moreover, this construction can be done in*

$O(|A|^2 T_1)$ time, where T_1 is the time to find a perfect matching in a bipartite graph with $2(m+|A|)$ nodes.

For example, the current best value of T_1 is $O(|A|(m+|A|)^{1/2}))$ ([11, 14]), and hence implies an overall time bound of $O(|A|^3(m+|A|)^{1/2})$ for the construction in Theorem 1.3.

Aggarwal and Orlin [1] show that the time bound in Theorem 1.3 can be improved by using a more efficient process.

Theorem 1.4 *(Aggarwal and Orlin [1]) The construction of an m-route path flow from an m-route arc-flow can be done in $O(|A|T_2)$, where T_2 is the time to solve a unit-capacity 1-MFP.*

For example the current best value of T_2 is $O(\min\{|V|^{2/3}|A|, |A|^{3/2}\})$ ([3, 11]), and hence implies an overall time bound of $O(\min\{|A|^2|V|^{2/3}, |A|^{5/2}\})$.

Actually, the time bounds in these two theorems are absent in both [17] and [1], we establish and include them here, and further repeat [1]'s construction in Appendix A for the purpose of completeness and self-explanation. Recently Du and Kabadi [9] further improve the bound in Theorem 1.4 to $O(|A|^2)$.

According to the above discussion, as long as we can solve the following arc-flow linear program, we can always obtain an optimal m-route path-flow by Theorem 1.3 or Theorem 1.4.

$$\max \tfrac{F}{m} \tag{7}$$

$$\text{subject to} \quad (4), (5), \text{ and } (6) \tag{8}$$

This linear program is of polynomial size and the coefficients of the constraint matrix are bounded by m, therefore it can be solved in strongly polynomial time using the technique of Tardos [26]. However, this result is impractical because of its poor worst-case complexity. We will be interested in efficient strongly polynomial algorithms in this paper.

Previous Work: Kishimoto and Takeuchi [18], and Kishimoto et al [20] investigate the special 2-MFP. Kishimoto and Takeuchi [19], and Kishimoto [17] study the more general m-MFP by giving a strongly polynomial combinatorial algorithm which amounts to solving m ordinary 1-MFP's, and their algorithm also implies a max-flow-min-cut relationship. While their algorithm is simple, its analysis is complicated. Aggarwal and Orlin [1] give an improved strongly polynomial combinatorial primal-dual algorithm which solves m ordinary 1-MFP's only in the worst case. Besides, they propose to solve m-MFP by binary search which results in calling the ordinary 1-MFP's in a number of $O(\log(|V|U_{\max}))$, where $U_{\max} = \max_{(i,j)\in A} u_{ij}$. This binary-search type method is *weakly* polynomial. Although this kind of algorithm can perform well in some practical cases, particularly for small $U_{\max}$, its drawbacks are also obvious. First of all it may not terminate if $U_{\max}$ is irrational. Although the fact that an algorithm may not ternimate with irrational data is of little practical concern, the fact that the complexity can be very high for large $U_{\max}$ *is* of practical concerns. Consequently, there have also been many interests in designing *strongly* polynomial algorithms that serve both purposes of being theoretically and practically efficient (e.g. [10, 26]). Throughout this paper we will only focus on strongly polynomial algorithms.

New Results: Both Kishimoto's algorithm [17] and Aggarwal and Orlin's primal-dual algorithm [1] are based on the idea of reducing m-MFP to solving a sequence of ordinary 1-MFP's. This is not coincidental, and as a matter of fact both of them can be unified into one common framework based on Newton's method, as we will report in this paper. An immediate benefit of this new fact is that proofs of correctness and complexities of the previous two algorithms in [17] and [1] become much easier.

Although these algorithms are efficient for this particular problem, many combinatorial properties were not exploited and the techniques used there cannot be extended to deal with more general problems, such as the *multiroute minimum cost flow problem* in [7] (Chapter 4) and the *multiroute two-commodity maximum flow problem* in [7] (Chapter 5) and [8]. This justifies searching for alternative methods to solve m-MFP, like an augmenting-path algorithm. However, a direct generalization of Ford and Fulkerson augmenting-path approach fails for this problem as observed in [1]. We shall devise a novel augmenting-path algorithm to solve m-MFP as our second major result in this paper. This augmenting-path algorithm will be the backbone for solving the aforementioned problems in [7, 8], where existing techniques cannot be applied.

We note that the two new algorithms proposed in this paper are mainly of theoretical significance. From practical point of view, they are not as efficient as the existing ones in [1] in terms of worst-case complexity. The main purpose of this paper is twofold. One is to simplify and extend the basic theory of multiroute flow by unifying existing results into one framework. This allows easier treatment and deeper understanding of the problem. Another is to explore the combinatorial structure of the problem. This exploitation results in an augmenting-path algorithm that serves as the foundation for solving more general problems, such as those considered in [7, 8]. To our best knowledge, this is the only currently known efficient (both theoretical and practical) procedure that can be applied to solve the aforementioned problems in [7, 8].

The rest of this paper is organized as follows. In Section 2, we present a new strongly polynomial combinatorial algorithm based on Newton's method. In Section 3, we explain why both Kishimoto's algorithm [17] and Aggarwal and Orlin's primal-dual algorithm [1] can be viewed as variants of Newton's method. In Section 4, we present an augmenting-path algorithm to directly solve m-MFP. We conclude the paper in Section 5.

2 Newton-Based Algorithm

For a given directed network $G = (V, A, u)$, an s-t cut of G is a partition $(S, \bar{S})$ of the vertices of V with $S \subset V$, $\bar{S} = V \backslash S$, $s \in S$, and $t \notin S$. We call (i, j) an arc of $(S, \bar{S})$ if $i \in S$ and $j \in \bar{S}$. We define next the capacity of an s-t cut, which will be used in the proofs of Theorems 3.2 and 4.3.

Definition 2.1 *Given an s-t cut $(S, \bar{S})$ of G, let $(i_1, j_1), \cdots, (i_\ell, j_\ell)$ be all arcs of $(S, \bar{S})$ ordered such that $u_{i_1 j_1} \geq \cdots \geq u_{i_\ell j_\ell}$. The capacity of $(S, \bar{S})$ is*

$$C(S, \bar{S}) = \min \left\{ \frac{\sum_{q=1}^{\ell} u_{i_q j_q}}{m}, \frac{\sum_{q=2}^{\ell} u_{i_q j_q}}{m-1}, \cdots, \frac{\sum_{q=m}^{\ell} u_{i_q j_q}}{1} \right\} \tag{9}$$

Any term on the right side of (9) is an upper bound on the maximum of all m-route path-flow values. This fact is easily verified as follows. Let F be the value of any given m-route path-flow f. Then the total flow of f across cut $(S, \bar{S})$ is mF. Because f is sent along m-routes, no cut arc (i_q, j_q) can route flow more than F in f for any $q \in \{1, \cdots, \ell\}$. Consider the k^{th} term $\sum_{q=k}^{\ell} u_{i_q j_q}/(m-k+1)$ of (9). We have $mF \leq (k-1)F + \sum_{q=k}^{\ell} u_{i_q j_q}/(m-k+1)$ when arc capacity $u_{i_q j_q}$ is replaced by F for all $q = 1, \cdots, k-1$. Rearranging the previous inequality, we obtain $(m-k+1)F \leq \sum_{q=k}^{\ell} u_{i_q j_q}/(m-k+1)$, which proves that the k^{th} term is indeed an upper bound for any $k \in \{1, \cdots, m-1\}$.

Actually Kishimoto [17] proves the bound in (9) is also a lower bound, which implies the following max-flow-min-cut theorem.

Theorem 2.2 *(Kishimoto [17]) The maximum of all m-route path-flow values equals to the minimum of all s-t cut capacities.*

The above result is also implied by the two algorithms that will be proposed in this paper.

Now we explain how our problem can be solved using Newton's method. We construct the following one-dimensional parametric 1-MFP by parameterizing the right-hand side of (6) in the aforementioned linear program (7)-(8), i.e., replacing F/m by a nonnegative parameter λ.

$$\mu(\lambda) = \max F \tag{10}$$

$$\sum_{j\in\delta_i^+} f_{ij} - \sum_{j\in\delta_i^-} f_{ji} = \left\{ \begin{array}{rl} F, & i = s \\ 0, & i \neq s,\ t \\ -F, & i = t \end{array} \right. \tag{11}$$

$$0 \leq f_{ij} \leq \min\{u_{ij},\ \lambda\}, \qquad \forall (i,j) \in A \tag{12}$$

This formulation falls with the framework studied by Megiddo [21]; so the problem can be solved using the technique there. However, Newton's method is much more efficient in this case. For comparison, we only present the worst-case complexity of Megiddo's method here, and refer the reader to the original paper of Megiddo [21] for further details: Megiddo's method solves (10)-(12) in $O(T_1(T_1 + T_2))$ time if 1-MFP is solvable within $O(T_1)$ comparisons and $O(T_2)$ additions.

Newton's method is a classical way of solving equations (see e.g. [23]). While Newton's method is normally used with smooth functions, it can also be made applicable to piece-wise smooth functions by some extra care of the breakpoints. Basically, Newton's method uses an iterative process to approach the roots of a function. This is achieved by proceeding along the gradient direction at each iteration. The efficiency of Newton′s method for combinatorial optimization is reported in [22, 23].

It is well-known, by the linear programming duality theory and the fact that the right-hand side of (12), i.e., $\min\{u_{ij},\ \lambda\}$ is piece-wise linear concave, the optimal objective value function $\mu(\lambda)$ of this linear program (10)-(12) is also a piece-wise linear concave function of $\lambda \geq 0$. Combining the previous construction (the parametrization), it is easy to see that the optimal value of the linear program (7)-(8) is the *unique* root λ^* (if exists) of the following equation:

$$H(\lambda) = m\lambda - \mu(\lambda) = 0 \tag{13}$$

See Figure 1 for an illustration on how $m\lambda$ and $\mu(\lambda)$ intersect to get the unique root. Now m-MFP is reduced to solving a single variable equation (13).

Let $G_\lambda = (V, A, u(\lambda))$ be the network that is the same as the network $G = (V, A, u)$ but with capacities $u_{ij}(\lambda) = \min\{u_{ij}, \lambda\}$, for any $(i,j) \in A$. Note that $\mu(\lambda)$ is differentiable everywhere except at the breakpoints; let $\mu'(\lambda)$ denote the derivative (or slope) at any non-breakpoint λ. Based on the max-flow-min-cut theorem for the ordinary 1-MFP, slope $\mu'(\lambda)$ equals to the number of arcs with value λ in any minimum cut of network G_λ. To see this, for any minimum cut C in G_λ, we have $\mu(\lambda) = \sum_{(i,j)\in C} u_{ij}(\lambda)$ by the max-flow-min-cut theorem for the ordinary 1-MFP. Therefore increasing λ by a sufficiently small amount gives the desired result.

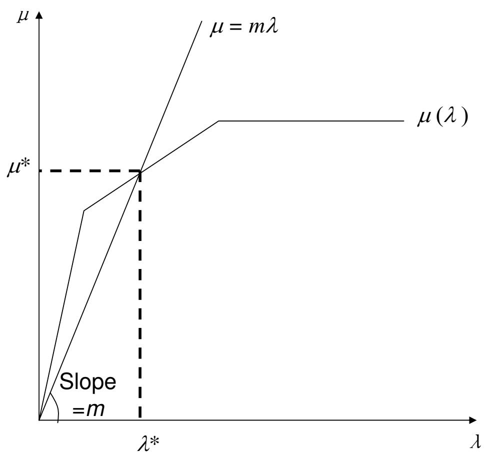

Figure 1: The piece-wise linear concave function $\mu(\lambda)$ is the optimal objective value of the linear program (10)-(12); $\mu = m\lambda$ is a line with slope m that passes through the origin, and (λ^*, μ^*) is the intersection point of these two functions. So λ^* is the unique root of equation (13).

Now we apply Newton's method to solve equation (13).

Newton's algorithm

Input: A directed network $G = (V, A, u)$ with vertex set V, arc set A, a source vertex $s \in V$, a destination vertex $t \in V$, a capacity vector $u \in R_+^{|A|}$, and a positive integer $m \in Z_+$.

Output: An optimal m-route arc-flow $f^* \in R_+^{|A|}$ and its value μ^*.

Step 0. (Initialization) Let $k = 0$, and let $\lambda_k = U_{\max} = \max_{(i,j)\in A} u_{ij}$.

Step 1. Solve an ordinary 1-MFP in G_{λ_k} to get an optimal 1-route arc-flow $f(\lambda_k) \in R_+^{|A|}$ and its corresponding value $\mu(\lambda_k)$. If $\mu(\lambda_k) = m\lambda_k$, then terminate and output $f^* = f(\lambda_k)$ and $\mu^* = \mu(\lambda_k)$.

Step 2. Otherwise, let

$$\begin{aligned} \lambda_{k+1} &= \lambda_k - \frac{H(\lambda_k)}{H'(\lambda_k)} && (14) \\ &= \lambda_k - \frac{m\lambda_k - \mu(\lambda_k)}{m - \mu'(\lambda_k)} \\ &= \frac{\mu(\lambda_k) - \mu'(\lambda_k)\lambda_k}{m - \mu'(\lambda_k)}, && (15) \end{aligned}$$

$k = k + 1$, and go to Step 1. ■

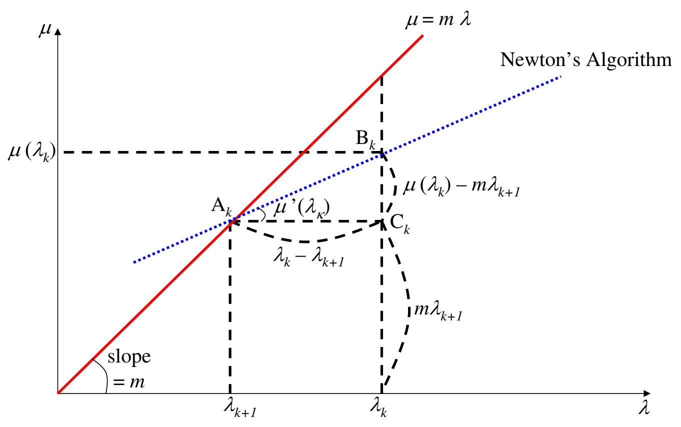

Figure 2: Point B_k has coordinates $(\lambda_k, \mu(\lambda_k))$, where λ_k is the current estimate of the root of equation (13) at the k^{th} iteration in G_{λ_k}; point A_k has coordinates $(\lambda_{k+1}, m\lambda_{k+1})$, where λ_{k+1} is the next estimate at the $(k+1)^{\text{th}}$ iteration in network $G_{\lambda_{k+1}}$, obtained by intersecting the line with slope $\mu'(\lambda_k)$ that passes through B_k with the line $\mu = m\lambda$; point C_k is the intersection of the horizontal line passing through A_k with the vertical line passing through B_k. The distance between A_k and C_k is $\lambda_k - \lambda_{k+1}$ and that between B_k and C_k is $\mu(\lambda_k) - m\lambda_{k+1}$.

Step 1 in the above algorithm requires solving an ordinary 1-MFP, for which efficient algorithms are available (see Chapters 6-8 of [2]).

A geometric explanation of one iteration in Newton's algorithm is illustrated in Figure 2. Starting from point $\mathrm{B}_k = (\lambda_k, \mu(\lambda_k))$ at the k^{th} iteration in the network G_{λ_k}, where λ_k and $\mu(\lambda_k)$ respectively are the current estimates of the root and its corresponding value. We draw a line with slope $\mu'(\lambda_k)$ that passes through point B_k to intersect $\mu = m\lambda$ at point $\mathrm{A}_k = (\lambda_{k+1}, m\lambda_{k+1})$. Checking the triangle $(\mathrm{A}_k, \mathrm{B}_k, \mathrm{C}_k)$ in Figure 2, we have

$$\mu'(\lambda_k)(\lambda_k - \lambda_{k+1}) = \mu(\lambda_k) - m\lambda_{k+1},$$

solving λ_{k+1}, we get (15). This is exactly how λ is updated in the above Newton's algorithm.

The following simple fact for $H(\lambda)$ will be used in the proof Theorem 2.3, where derivative H' is defined only at non-break points.

$$\forall\lambda \geq \lambda^* : H(\lambda) \geq 0,\ H'(\lambda) = m - \mu'(\lambda) \geq 0 \tag{16}$$

Theorem 2.3 *Newton's algorithm terminates with an optimal m-route arc-flow in at most m iterations, or equivalently, Newton's algorithm terminates with an optimal m-route arc-flow by solving the ordinary 1-MFP's at most m times.*

Proof: First, by (14) and (16) we know the estimated λ-values decrease iteration by iteration before the termination. Next we claim that, $\forall k \geq 0$, $\mu(\lambda_k) < m\lambda_k$ before termination, which implies the correctness of the algorithm. This can be done by induction on k. Initially, $\lambda_0 = U_{\max}$, the claim is true. Suppose $\mu(\lambda_k) < m\lambda_k$. We want to show $\mu(\lambda_{k+1}) \leq m\lambda_{k+1}$. Based on the previous geometric explanation (Figure 2), we know that λ_{k+1} is the intersection point of the line B_kA_k and the line $\mu = m\lambda$, and these two lines are never above $\mu(\lambda)$, $\forall \lambda \in [\lambda_{k+1}, \lambda_k]$. Therefore we must have $\mu(\lambda_k) \leq m\lambda_k$, which implies our claim since we cannot have $\mu(\lambda_k) = m\lambda_k$ before termination.

For the total number of iterations, first note that $\mu'(\lambda_{k+1}) \geq \mu'(\lambda_k)$ because $\mu(\lambda)$ is a concave function and $\lambda_{k+1} \leq \lambda_k$. We claim $\mu'(\lambda_{k+1}) > \mu'(\lambda_k)$ before the termination. This fact directly implies the theorem because $\mu'(\lambda_k)$ is a nonnegative integer and $\mu'(\lambda_k) < m$ (by (16)) before the termination. Suppose on the contrary, $\mu'(\lambda_{k+1}) = \mu'(\lambda_k)$ before the termination. Therefore there is no breakpoint between λ_k and λ_{k+1}, therefore both $\mathrm{B}_k = (\lambda_k, \mu(\lambda_k))$ and $\mathrm{A}_k = (\lambda_{k+1}, \mu(\lambda_{k+1}))$ lie on a same line segment with slope $\mu'(\lambda_k)$. Therefore

$$\mu(\lambda_{k+1}) - \mu(\lambda_k) = \mu'(\lambda_k)(\lambda_{k+1} - \lambda_k) = m\lambda_{k+1} - \mu(\lambda_k),$$

where the second equality follows from (15). From this, we have $\mu(\lambda_{k+1}) = m\lambda_{k+1}$, a contradiction with the fact that the algorithm did not terminate yet. $\square$

Combining Theorems 1.4 and 2.3, we have:

Corollary 2.4 *m-MFP is solvable in $O(mT_3 + T_4)$ time, where T_3 is the time to solve a 1-MFP, and T_4 is the time to construct an m-route path-flow from the optimal m-route arc-flow.*

For example the current best value of T_3 is $O(|V||A| \log_{|A|/(|V| \log |V|)} |V|)$ ([16]), and the best value of T_4 is $O(|A|^2)$ [9], and hence implies an overall time bound of $O(m|V||A| \log_{|A|/(|V| \log |V|)} |V| + |A|^2)$.

3 Applications of Newton's Algorithm

Now we explain why both Kishimoto's algorithm [17] and Aggarwal and Orlin's primal-dual algorithm [1] can be viewed as variants of Newton's method.

3.1 Kishimoto's Algorithm

Note that Newton's algorithm and its analysis are still valid as long as the following two conditions are satisfied:

(i) The slopes increase at each iteration, i.e., $\mu'(\lambda_{k+1}) > \mu'(\lambda_k)$, for any k;

(ii) Line B_kA_k is either strictly below or the same as line $\mu(\lambda)$, for any $\lambda \in [\lambda_{k+1}, \lambda_k]$.

Obviously (i) and (ii) still hold if $\mu'(\lambda_k)$ is replaced by k in formula (15) at the k^{th} iteration, which is exactly the λ update in Kishimoto's algorithm [17]:

$$\lambda_{k+1} = \frac{\mu(\lambda_k) - k\lambda_k}{m - k} \tag{17}$$

Therefore we have:

Theorem 3.1 *Kishimoto's algorithm terminates in m iterations with an optimal m-route arc-flow.* $\square$

Note that the slope in Kishimoto's algorithm increases by one each iteration, while the slope in Newton's algorithm increases by at least one (since the slope $\mu'(\lambda)$ also equals the number of arcs with value λ in any minimum cut of network G_λ). This implies that the step size to obtain the next estimate of λ in Newton's algorithm is no less than that in Kishimoto's algorithm. Therefore Newton's algorithm terminates in no more iterations than Kishimoto's algorithm.

3.2 Aggarwal and Orlin's Primal-Dual Algorithm

Aggarwal and Orlin's primal-dual algorithm [17] is the same as Newton's algorithm except that (15) is replaced with

$$\lambda_{k+1} = C(S_k, \bar{S}_k), \tag{18}$$

where $(S_k, \bar{S}_k)$ is a minimum s-t cut in G_{λ_k} at iteration k, and $C(S_k, \bar{S}_k)$ is the cut capacity defined in (9) of Definition 2.1 using the original capacity vector u in G.

Theorem 3.2 *Aggarwal and Orlin's primal-dual algorithm terminates in at most m iterations with an optimal m-route arc-flow.*

Proof: First, it is easy to see $\forall k$, $\mu(\lambda_k) \leq m\lambda_k$, since $C(S_k, \bar{S}_k)$ defined in (9) is an upper bound on $\mu(\lambda_k)$. This implies the correctness of the algorithm. For the number of iterations, we prove

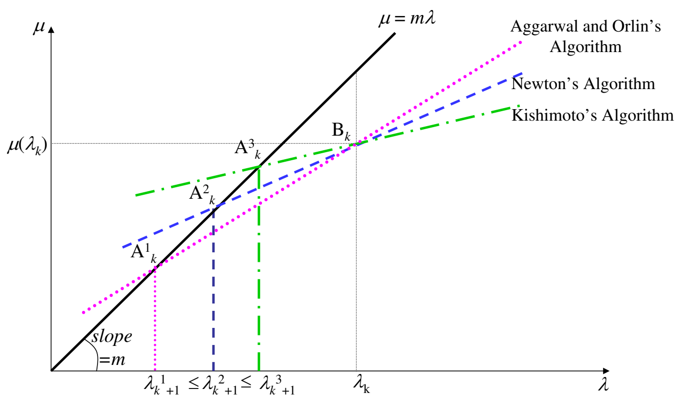

Figure 3: λ_k is the current estimate of the root of equation (13) at the k^{th} iteration in network G_{λ_k}; λ^1_{k+1}, λ^2_{k+1}, and λ^3_{k+1} are the next estimates at the $(k+1)^{\text{th}}$ iteration, updated respectively according to the formulas (18), (15), and (17). Points B_k, A^1_k, A^2_k, and A^3_k have coordinates $(\lambda_k, \mu(\lambda_k))$, $(\lambda^1_{k+1}, m\lambda^1_{k+1})$, $(\lambda^2_{k+1}, m\lambda^2_{k+1})$, $(\lambda^3_{k+1}, m\lambda^3_{k+1})$, respectively, where A^1_k, A^2_k, and A^3_k are obtained by intersecting line $\mu = m\lambda$ with the three lines that share the same point B_k, but have different slopes, corresponding to Aggarwal and Orlin's primal-dual algorithm, Newton's algorithm, and Kishimoto's algorithm, respectively.

this algorithm runs in no more iterations than Newton's algorithm. Let $(S_k, \bar{S}_k)$ be a minimum s-t cut in G_{λ_k} at iteration k, and let $(i_1, j_1), \cdots, (i_\ell, j_\ell)$ be all the arcs of $(S_k, \bar{S}_k)$ such that $u_{i_1j_1} \ge \cdots \ge u_{i_tj_t} \ge \lambda_k > u_{i_{t+1}j_{t+1}} \ge \cdots \ge u_{i_\ell j_\ell}$. So the number of arcs with capacities no smaller than λ_k is t. This implies $t = \mu'(\lambda_k)$. Let $Y = \sum_{q=t+1}^{\ell} u_{i_qj_q}$. Then $\mu(\lambda_k) = t\lambda_k + Y = \mu'(\lambda_k)\lambda_k + Y$. Therefore we have $\frac{Y}{m-t} = \frac{\mu(\lambda_k)-\mu'(\lambda_k)\lambda_k}{m-\mu'(\lambda_k)}$. However, by (9), we have

$$\begin{aligned}\lambda_{k+1} &= C(S_k, \bar{S}_k)\\ &= \min\left\{\frac{\sum_{q=1}^{\ell} u_{i_qj_q}}{m}, \cdots, \frac{Y}{m-t}, \frac{Y-u_{t+1}}{m-t-1}, \cdots, \frac{Y-\sum_{q=1}^{m-1} u_{i_{t+q}j_{t+q}}}{1}\right\}.\end{aligned}$$

Therefore $\lambda_{k+1} \le \frac{Y}{m-t} = \frac{\mu(\lambda_k)-\mu'(\lambda_k)\lambda_k}{m-\mu'(\lambda_k)}$, where the last term is exactly the iteration formula (15) used in Newton's Algorithm. Therefore Aggarwal and Orlin's primal-dual algorithm requires no more iterations than Newton's algorithm, which is at most m. Therefore the theorem follows. □

Figure 3 summarizes the relationship among the three algorithms. Note that $\lambda^1_{k+1} \le \lambda^2_{k+1} \le \lambda^3_{k+1}$. This fact implies that the step size in Kishimoto's algorithm is no more than that in Newton's algorithm, which again is no more than that in Aggarwal and Orlin's primal-dual algorithm. Therefore Kishimoto's algorithm is no more efficient than Newton's algorithm, which is no more efficient than Aggarwal and Orlin's primal-dual algorithm in terms of worst-case complexity.

4 Augmenting-path Algorithm for m-MFP

As observed by Aggarwal and Orlin in [1], the following direct extension of the Ford and Fulkerson augmenting path algorithm [12] from 1-MFP to m-MFP fails to find an optimal solution: identify an m-route from s to t, and simultaneously augment an appropriate amount of flow along each of these m paths subject to the flow capacity constraints; continue to augment until no m-route exists.

In this section we devise a novel augmenting-path algorithm by introducing a new concept of m-path (Definition 4.1 (3)). This allows us to solve m-MFP by augmenting flow along an m-path instead of an m-route in each iteration.

In Section 4.1, we describe the augmenting scheme, present the main algorithm and prove its correctness. We can implement the proposed algorithm in weakly polynomial time by the standard capacity-scaling technique for rational data (see e.g., [2] for further details on this technique). Once again, however, a direct generalization of the shortest-path idea by Dinic [6] and Edmonds and Karp [10] for 1-MFP fails to yield a strongly polynomial implementation even for $m = 2$. This is shown by a counterexample in Section 4.2. While we still do not know how to implement our algorithm in strongly polynomial time for general m, we manage to achieve this for $m = 2$ using a new method in Section 4.3.

4.1 Main Algorithm

First, we need some preliminary definitions and notations.

Definition 4.1 *Given a directed network $G = (V, A, u)$ with vertex set V, arc set A, a source vertex s, a destination vertex t, a capacity vector $u \in R_+^{|A|}$, and a positive integer $m \in Z_+$, let $f \in R_+^{|A|}$ be an m-route arc-flow with value F/m.*

(1) *The **residual graph** $G(f) = (V, A(f))$ is defined as follows: any arc $(i, j) \in A$ is replaced by two arcs: the **forward** arc (i, j) with residual capacity of $u_{ij} - f_{ij}$, and the **backward** arc (j, i) with residual capacity of f_{ij}. The residual graph contains only those arcs with positive residual capacities. Let $A_+(f)$ and $A_-(f)$ be the set of all forward and backward arcs in $G(f)$.*

(2) *A **tight** arc $(i, j) \in A(f)$ is a forward arc with $f_{ij} = F/m$ (therefore, all backward arcs are non-tight). Let $A_T(f)$ be the set of tight arcs in $G(f)$, so $A_T(f) \subset A_+(f)$.*

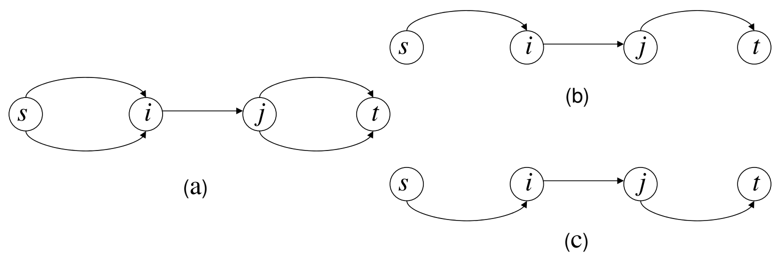

Figure 4: (a) is a 2-path, which contains two paths (b) and (c), where arc (i, j) is shared by both paths

(3) *An m-**path** consists of m s-t paths in $G(f)$ such that any tight arc appears in at most one of these m paths. Note that an m-path is not necessarily arc-disjoint; a different terminology, m-route, is reserved for that case as defined in Definition 1.1(1). Figure 4 is an example of 2-path.*

(4) *A **bridge** of $G(f)$ is an arc whose deletion leaves no directed s-t path.*

We identify an m-path with an integer solution of the following feasibility problem defined in the residual graph $G(f)$:

$$\sum_{j\in\delta_i^+} x_{ij} - \sum_{j\in\delta_i^-} x_{ji} = \begin{cases} m, & i = s \\ 0, & i \neq s,\ t \\ -m, & i = t \end{cases} \tag{19}$$

$$0 \leq x_{ij} \leq 1 \qquad (i,j) \in A_T(f) \tag{20}$$

$$x_{ij} \geq 0 \qquad (i,j) \in A \tag{21}$$

This is the feasibility problem of a typical minimum cost flow problem. Therefore we can check its feasibility by transforming it into a 1-MFP (see e.g. [2]). Whenever it is feasible, there exists an integer solution x such that $0 \leq x_{ij} \leq m$ which can be decomposed into m paths (not necessarily arc-disjoint) such that no tight arc appears more than once due to the constraints (20). This is exactly an m-path defined earlier. For simplicity, we refer any such solution x as an m-path from now on.

Now we state our main algorithm formally.

Algorithm AUG

Input: A directed network $G = (V, A, u)$ with vertex set V, arc set A, a source $s \in V$, a destination $t \in V : t \neq s$, a positive integer $m \in Z_+$, and a capacity vector $u \in R_+^{|A|}$.

Output: An optimal m-route arc-flow $f^* \in R_+^{|A|}$ and its value μ^*.

Step 0. (Initialization) let $k = 0$, and let $f^0 \in R_+^{|A|}$ be an m-route arc-flow such that $f_{ij}^0 = 0$, $\forall (i,j) \in A$, and its corresponding value $F^0/m = 0$.

Step 1. Find an m-path x^k in the current residual graph $G(f^k)$ by solving (19)-(21). If no such m-path exists (i.e., (19)-(21) is infeasible), then optimality is reached. The algorithm terminates and outputs $f^* = f^k$ and $\mu^* = F^k/m$. Otherwise let $A_+^k(\tau) = \{(i,j) \in F(f^k) : x_{ij}^k = \tau\}$ and $A_-^k(\tau) = \{(i,j) \in B(f^k) : x_{ij}^k = \tau\}$ be the sets of forward and backwards arcs in x^k with value τ, respectively, where $1 \le \tau \le m$.

Step 2. By default $c/0 = +\infty$, for any constant c, so the following first formula is well-defined even for $\tau = 1$. Let

$$\begin{aligned} \delta_{A_+^k(\tau)} &= \min_{(i,j)\in A_+^k(\tau)} [\min\{\frac{u_{ij} - f_{ij}}{\tau}, \frac{\frac{F}{m} - f_{ij}}{\tau - 1}\}], \\ \delta_{A_-^k(\tau)} &= \min_{(j,i)\in A_-^k(\tau)} \{\frac{f_{ij}}{\tau}\}, \\ \delta^k &= \min_{\tau=1,\cdots,m} [\min\{\delta_{A_+^k(\tau)}, \delta_{A_+^k(\tau)}\}]. \end{aligned} \tag{22}$$

Generate a new m-route arc-flow, and its value as follows:

$$f_{ij}^{k+1} = f_{ij}^k + \tau\delta^k,\ \forall (i,j) \in A_+^k(\tau), \tag{23}$$

$$f_{ij}^{k+1} = f_{ij}^k - \tau\delta^k,\ \forall (j,i) \in A_-^k(\tau), \tag{24}$$

$$F^{k+1} = F^k + m\delta^k.$$

Generate a new residual graph, and let $k = k + 1$, go to Step 1. ■

The correctness of algorithm AUG follows from the following two theorems: the first of which shows that the m-route arc-flow generated at each iteration in algorithm AUG is again an m-route arc-flow, provided that the previous one is; the second of which shows that the output is optimal whenever the algorithm terminates.

Theorem 4.2 *Given an m-route arc-flow $f^k \in R_+^{|A|}$ with value F^k/m at the k^{th} iteration, the new flow $f^{k+1} \in R_+^{|A|}$ generated in (22), (23) and (24) at the $(k+1)^{\text{th}}$ iteration is an m-route arc-flow with value $F^k/m + \delta^k$.*

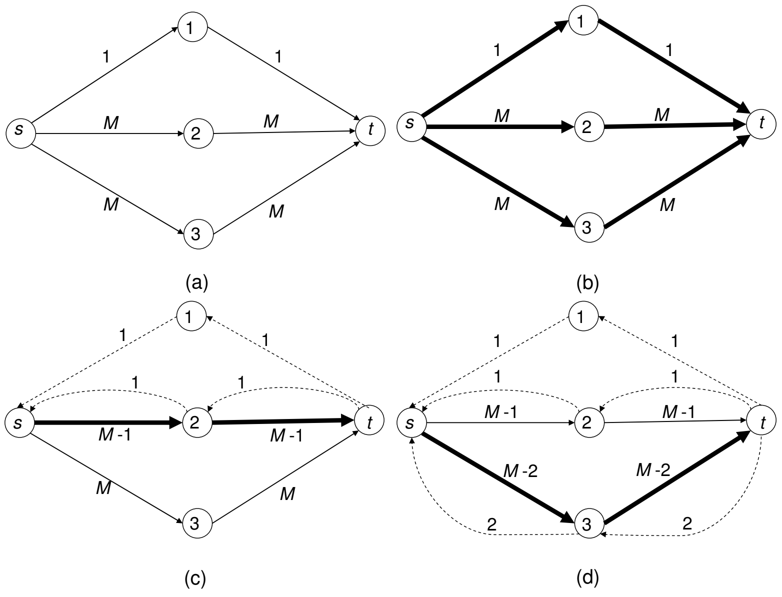

Figure 5: A counterexample for 2-MFP

Proof: See Appendix B. □

Theorem 4.3 *If algorithm* AUG *terminates, it outputs an optimal m-route arc-flow.*

Proof: See Appendix C. □

4.2 A Counterexample

From now on we focus on the case $m = 2$. We will show that simple extensions of the shortest-path (in terms of the number of arcs) idea by Dinic [6] and Edmonds and Karp [10] for 1-MFP, no longer yields strongly polynomial implementation.

There are two ways to count the number of arcs on a shortest 2-path:

Version 1. Each shared arc in the 2-path is counted once.

Version 2. Each shared arc in the 2-path is counted twice since they are used in both paths.

The example in Figure 5(a) shows that both versions fail. In Figure 5(a), vertex s is the source, vertex t is the destination, and the numbers beside the arcs are the capacities, where M is a large number. We now apply algorithm AUG to this example by choosing a shortest 2-path (Versions 1 and 2) at each iteration.

Version 1:

Step 0. flow $f^0 = 0$ with value $F^0 = 0$. Figure 5(b) is the residual graph $G(f^0)$, where the heavy arcs are tight, and the numbers beside the arcs are the residual capacities. All arcs are tight initially. One shortest 2-path in the current residual graph $G(f^0)$ is $\{s-1-t, s-2-t\}$. Therefore $\delta^1 = 1$ (by (22)), and the new flow $f^1_{s1} = f^1_{1t} = f^1_{s2} = f^1_{2t} = 1$ (by (23)-(24)), $f^1_{ij} = f^0_{ij}$, ($\forall (i,j) \in A \backslash \{(s,1),(1,t),(s,2),(2,t)\}$) with $F^1 = 2$.

Step 1. Figure 5(c) is the residual graph $G(f^1)$, where the dotted arcs are backward, the heavy arcs are tight, and the numbers beside the arcs are the residual capacities. One shortest 2-path in $G(f^1)$ is $\{s-3-t, s-3-t\}$. Therefore $\delta^2 = 1$, and $f^2_{s3} = f^2_{3t} = 3$, $f^2_{ij} = f^1_{ij}$ ($\forall (i,j) \in A \backslash \{(s,3),(3,t)\}$) with $F^2 = 4$.

Step 2. Figure 5(d) is the residual graph $G(f^2)$, where the dotted lines are backward arcs, and the numbers beside the arcs are the residual capacities. One shortest 2-path in $G(f^2)$ is $\{s-2-t, s-2-t\}$. Therefore $\delta^3 = 1$, and $f^3_{s2} = f^3_{2t} = 2$, $f^3_{ij} = f^2_{ij}$ ($\forall (i,j) \in A \backslash \{(s,2),(2,t)\}$) with $F^3 = 6$.

Steps 1 and 2 will repeat $O(M)$ times before the optimal solution is found. ■

Version 2: In Step 1, there are only two 2-paths: one is $\{s-3-t,\ s-3-t\}$ with four arcs (each shared arc counted twice), the other is $\{s-2-t,\ s-3-t\}$ with the same number of arcs. Therefore if we are allowed to choose any minimum 2-path, the algorithm is the same as Version 1. ■

However, if we choose the second 2-path $\{s-2-t,\ s-3-t\}$ in Step 1. Then $\delta^2 = 1$, and $f^2_{s2} = f^2_{2t} = M$, $f^2_{s3} = f^2_{3t} = M-1$, $f^2_{ij} = f^1_{ij}$ ($\forall (i,j) \in A \backslash \{(s,2),(2,t),(s,3),(3,t)\}$) with $F^2 = M$. Then in Step 2, the only 2-path is $\{s-3-t,\ s-3-t\}$ in $G(f^2)$, so $\delta^3 = 1$, and $f^3_{s2} = f^3_{2t} = M$, $f^3_{ij} = f^2_{ij}$ ($\forall (i,j) \in A \backslash \{(s,2),(2,t)\}$) with $F^3 = 2M+1$. We obtain the optimal solution after three iterations.

This particular example suggests two key points that actually lead to strongly polynomial termination as we shall prove in the next section: first, each shared arc is counted twice; second, any 2-route has higher priority to be chosen than any 2-path (a 2-path is chosen only if there is no 2-route available). For example, we should choose $\{s-2-t,\ s-3-t\}$ over $\{s-3-t,\ s-3-t\}$ in Step 1 of Version 2 above; and since there is no 2-route available in Step 2, the only 2-path $\{s-3-t,\ s-3-t\}$ is chosen.

4.3 Strongly Polynomial Termination for 2-MFP

We highlight the main idea on how to implement the two points observed earlier. At each iteration: (1) if there exists some 2-route, then choose a shortest one (in terms of minimum number of arcs); (2) otherwise there must exist some bridges before optimality is reached, choose a shortest 2-path.

Actually we can unify cases (1) and (2) by defining a modified residual graph. Suppose f^k is the m-route arc-flow with value F^k/m at the k^{th} ($k \geq 0$) iteration in algorithm AUG. Denote $A_B(f^k)$ to be the set of all bridges in the k^{th} residual graph $G(f^k)$. Obviously all bridges must be on every 2-path from s to t in $G(f^k)$. We duplicate each bridge $(i,j) \in A_B(f^k)$ by two parallel arcs to construct a new residual graph $G'(f^k) = (V'(f^k), A'(f^k))$. There always exists a 2-route in the new residual graph $G'(f^k)$ before optimality is reached. For any 2-route in $G'(f^k)$, we can recover a 2-path in $G(f^k)$ by merging back the parallel arcs.

Now we show that if we choose a shortest 2-route in $G'(f^k)$ at each iteration of algorithm AUG (therefore the corresponding 2-path in $G(f^k)$), then we obtain a strongly polynomial algorithm. The resulting algorithm will be denoted as AUG$'$.

To prove this, we note that at each iteration we can identify the bridges set $A_B(f^k)$ in $O(|A(f^k)|)$ time, and obtain a shortest 2-route in $O(|A| + |V|\log|V|)$ time by applying the method of Suurballe and Tarjan [25]. Therefore, it suffices to show that the number of iterations is bounded by a polynomial of $|V|$ and $|A|$.

Let α_k denote the number of arcs of a shortest 2-route in $G'(f^k)$, and let β_k be the number of arcs in the union of all shortest 2-route in $G'(f^k)$. So $\alpha_k \leq 2|V|$ before optimality is reached (by default $\alpha_k = +\infty$ at the optimality), and $\beta_k \leq 2|A|$. The following result is an extension of a similar result of Shrijver [24] for the ordinary 1-MFP.

Lemma 4.4 *For any k,*

(i) $\alpha_{k+1} \geq \alpha_k$*;*

(ii) *If $\alpha_{k+1} = \alpha_k$ then $\beta_{k+1} < \beta_k$.*

Proof: See Appendix D. □

Now the worst-case complexity follows immediately from the above lemma.

Theorem 4.5 *Algorithm* AUG$'$ *solves 2-MFP in $O(|V||A|^2 + |V|^2|A|\log|V|)$ time.*

Proof: By Lemma 4.4, there is no 2-route in the residual graph after at most $O(|V||A|)$ iterations; For each iteration, we take $O(|A|)$ to identify all the bridges, and $O(|A| + |V| \log |V|)$ to find the shortest 2-route using the method in [25]. In total we have $O(|V||A|^2 + |V|^2|A| \log |V|)$. $\square$

5 Conclusions

In this paper, we present two new algorithms to solve m-MFP. One is based on Newton's method, and another is based on the augmenting-path technique. We show that two existing algorithms for m-MFP, the algorithm by Kishimoto [17] and the primal-dual algorithm by Aggarwal and Orlin [1], can both be viewed as variants of Newton's method. An immediate benefit of this new fact is that proofs of correctness and complexities of the previous two algorithms in [17] and [1] become much easier. The augmenting-path algorithm proposed in this paper has potential applications in more general problems, such as the *multiroute minimum cost flow problem* [7] (Chapter 4) and the *multiroute two-commodity maximum flow problem* in [7] (Chapter 5) and [8], to which existing techniques cannot be applied. However, the potential applications are limited as we are only able to implement the augmenting-path algorithm in strongly polynomial time for the special 2-MFP. An obvious open question is therefore to design a strongly polynomial implementation for the general m-MFP ($m > 2$).

Acknowledgement: We thank the anonymous referees for their numerous and constructive suggestions in improving the presentation of this paper. We also thank one of the referees for pointing out the important reference [21], which is absent in the first version.

References

[1] C. C. Aggarwal and J. B. Orlin, On Multiroute Maximum Flows in Networks, *Networks,* **39**(1), 2002, 43-52.

[2] R. K. Ahuja, T. L. Maganti, and J. B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice-Hall, Engle-Wood Cliffs, NJ, 1993.

[3] R. K. Ahuja and J. B. Orlin, Distance-directed Augmenting Path Algorithms for Maximum Flow and Parametric Maximum Flow Problems, *Naval Research Logistics Quarterly,* **38**, 1991, 413-430.

[4] Y. P. Aneja, R. Chandrasekaran, S. N. Kabadi, and K. P. K. Nair, Flows over Edge-Disjoint Mixed Multipaths and Applications, *Working Paper, Faculty of Administration, University of New Brunswick*, 2002.

[5] R. Chandrasekaran, K. P. K. Nair, Y. P. Aneja, and S. N. Kabadi, Multi-Terminal Multipath Flows: Synthesis, *Discrete Applied Mathematics*, **143**, 2004, 182-193.

[6] E. A. Dinic, Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation, *Soviet Mathematics Doklady,* **11**, 1970, 1227-1280.

[7] D. Du, Multiroute Flow Problem, *PhD Thesis, The University of Texas at Dallas*, 2003.

[8] D. Du and R. Chandrasekran, The Multiroute Two-commodity Maximum Flow Problem, *Working Paper, The University of Texas at Dallas*, 2002.

[9] D. Du and S. N. Kabadi, An Improved Algorithm for Decomposing Arc Flows into Multipath Flows, accepted for publication in *Operations Research Letters*, 2005.

[10] J. Edmonds and R. M. Karp, Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems, *Journal of ACM,* **19**, 1972, 248-264.

[11] S. Even and R. E. Tarjan, Network Flow and Testing Graph Connectivity, *Siam Journal on Computing,* **4**, 1975, 507-518.

[12] L. R. Ford and D. R. Fulkerson, Maximal Flow Through a Network, *Canadian Journal of Mathematics* , **39**, 1956, 399-404.

[13] M. Grotschel, L. Lovasz, and A. Shrijver, Geometric Algorithms and Combinatorial Optimization, Springer-Verlag, Berlin, 1988.

[14] J. E. Hopcroft, R. E. Tarjan, An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs, *SIAM Journal on Computing*, **2**, 1973, 135-158.

[15] S. N. Kabadi, R. Chandrasekaran, K. P. K. Nair, and Y. P. Aneja, Integer Version of Multipath Flow Network Synthesis Problem, *Working Paper, Faculty of Administration, University of New Brunswick*, 2003.

[16] V. King, S. Rao, and R. E. Tarjan, A Faster Deterministic Maximum Flow Algorithm, *Journal of Algorithms*, **17**, 1994, 447-474.

[17] W. Kishimoto, A Method for Obtaining the Maximum Multiroute Flows in a Network, *Networks*, **27**(4), July 1996, 279-291.

[18] W. Kishimoto and M. Takeuchi, On Two-route Flows in an Undirected Network, *IEICE Technical Report*, CAS90-19, DSP-90-23, 1990 (in Japanese).

[19] W. Kishimoto and M. Takeuchi, On M-route Flows in a Network, *IEICE Trans J-76-A*, 1993, 1185-1200.

[20] W. Kishimoto, M. Takeuchi, and G. Kishi, Two-route Flows in an Undirected Flow Network, *IEICE Trans J-75-A*, 1992, 1699-1717 (in Japanese).

[21] N. Megiddo, Combinatorial Optimization with Rational Objective Functions, *Mathematics of Operations Research*, **4**(4), 1979, 414-424.

[22] T. Radzik, Newton's Method for Fractional Combinatorial Optimization, *Proceedings of 33rd IEEE Symposium on FOCS*, 1992, 659-669.

[23] T. Radzik, Fractional Combinatorial Optimization, In *Handbook of Combinatorial Optimization*, editors D-Z. Du and P. Pardalos, Vol.1, Kluwer Academic Publishers, 1998, 429-478.

[24] A. Shrijver, Theory of Linear Programming and Integer Programming, John Wiley, New York, 1986, 153-155.

[25] J. W. Suurballe and R. E. Tarjan, A Quick Method for Finding Shortest Pair of Disjoint Paths, *Networks*, **14**, 1984, 325-331.

[26] E. Tardos, A Strongly Polynomial Algorithm to Solve Combinatorial Linear Programs, *Operations Research*, **34**, 1986, 250-256.

APPENDIX A

Proof of Theorem 1.4: [1] The correctness of the following algorithm is implied by Lemmas 1 and 2 in [1], although not explicitly stated.

Input: An m-route arc-flow $f \in R_+^{|A|}$ with value F/m on a directed network $G = (V, A, u)$ with capacity $u \in R_+^{|A|}$. Based on the path-decomposition mechanism we can make the following assumption without loss of generality: the subgrpah induced by deleting all the arcs with zero flow is acyclic (see e.g. [2], Pages 79-83).

Output: An m-route path-flow $x \in R_+^{|\mathcal{P}|}$ with value F/m, where $\mathcal{P}$ is the set of all m-routes from s to t in G.

Step 1. Solve a unit-capacity 1-MFP by replacing the lower bound on arc (i,j) by $\lfloor mf_{ij}/F \rfloor$, the upper bound by $\lceil mf_{ij}/F \rceil$. Let y be an integer solution of this unit-capacity 1-MFP. Then y corresponds to an m-route such that $y_{ij} = 0$ when $f_{ij} = 0$, and $y_{ij} = 1$ when $f_{ij} = F/m$ (Lemma 1 in [1]).

Step 2. Let $\Delta_1 = \min\{F/m - f_{ij} : y_{ij} = 0\}$, and $\Delta_2 = \min\{f_{ij} : y_{ij} = 1\}$, and let $\Delta = \min\{\Delta_1, \Delta_2\}$. Update f_{ij} and F as follows:

$$\begin{aligned} f : &= f - \Delta y \\ F : &= F - m\Delta. \end{aligned}$$

Let P be the m-route induced by y. Let $x(P) = \Delta y$. If $F = 0$, then terminate and output an m-route path-flow constituted by all the $x(P)$'s; otherwise, go to Step 1. ■.

An arc $(i,j) \in A$ is *intermediate* if $0 < f_{ij} < u_{ij}$. By Lemma 2 in [1], the above process can only iterate no more than the number of intermediate arcs in the initial network. Moreover each iteration involves solving a unit-capacity 1-MFP. So the overall time bound follows. □

Remark 5.1 *Strictly speaking, the unit-capacity 1-MFP solved at each iteration in Step 1 of the above algorithm may also has nonzero lower bounds (either 0 or 1). However standard technique ([2], Pages 191-192) can reduce this problem to solving two 1-MFP's with zero lower bounds without affecting the worst-case complexity.*

APPENDIX B

Proof of Theorem 4.2: First the flow conservation constraints are easy to verify. For the capacity constraint $0 \le f_{ij}^{k+1} \le u_{ij}$, we need to show the following:

$$0 \le f_{ij}^k + \tau\delta^k \le u_{ij}, \quad \text{if} \quad (i,j) \in A_+^k(\tau) \tag{25}$$

$$0 \le f_{ij}^k - \tau\delta^k \le u_{ij}, \quad \text{if} \quad (j,i) \in A_-^k(\tau). \tag{26}$$

By the definition of δ^k in (22), we have

$$\tau\delta^k \le u_{ij} - f_{ij}^k, \quad \text{if} \quad (i,j) \in A_+^k(\tau) \tag{27}$$

$$\tau\delta^k \le f_{ij}^k, \quad \text{if} \quad (j,i) \in A_-^k(\tau). \tag{28}$$

Therefore (25) and (26) follow from (27) and (28), respectively, and the fact that f^k is an m-route arc-flow.

Finally, the route constraint $0 \leq f_{ij}^{k+1} \leq \frac{F^k}{m} + \delta^k$ is equivalent to

$$0 \leq f_{ij}^k + \tau\delta \leq \frac{F^k}{m} + \delta^k, \quad \text{if} \quad (i,j) \in A_+^k(\tau) \tag{29}$$

$$0 \leq f_{ij}^k - \tau\delta \leq \frac{F^k}{m} + \delta^k \quad \text{if} \quad (j,i) \in A_-^k(\tau). \tag{30}$$

By the definition of δ^k in (22), we have

$$(\tau - 1)\delta^k \leq \frac{F^k}{m} - f_{ij}^k, \quad \text{if} \quad (i,j) \in A_+^k(\tau) \tag{31}$$

$$\tau\delta^k \leq f_{ij}^k, \quad \text{if} \quad (j,i) \in A_-^k(\tau). \tag{32}$$

Therefore (29) and (30) follow from (31) and (32), respectively, by the same argument as before. □

APPENDIX C

Proof of Theorem 4.3: The algorithm stops when there is no m-path in the residual graph $G(f)$, where f is the current m-route arc-flow with value F/m. This is equivalent to the infeasibility of (19)-(21). Therefore there must exist an s-t cut $(S, \bar{S})$ in the original network G such that the cut arcs are divided into two disjoint groups

$$\begin{aligned} A_1 &= \{(i,j) \in (S,\bar{S}) : f_{ij} = \tfrac{F}{m} < u_{ij}\} \\ A_2 &= \{(i,j) \in (S,\bar{S}) : f_{ij} = u_{ij} \leq \tfrac{F}{m}\} \end{aligned}$$

satisfying

$$\begin{aligned} m &> |A_1| \\ \sum_{(i,j)\in A_2} u_{ij} &= (1 - \frac{|A_1|}{m})F. \end{aligned} \tag{33}$$

Let $u_{i_1j_1} \geq \cdots \geq u_{i_{|A_1|}j_{|A_1|}}$ and $u_{i_{|A_1|+1}j_{|A_1|+1}} \geq \cdots \geq u_{i_\ell j_\ell}$ be the arc capacities in A_1 and A_2 respectively. By (9), the capacity of $(S, \bar{S})$ is the following:

$$C(S,\bar{S}) = \min\{\frac{\sum_{q=1}^{\ell} u_q}{m}, \frac{\sum_{q=2}^{\ell} u_q}{m-1}, \cdots, \frac{\sum_{q=m}^{\ell} u_q}{1}\}. \tag{34}$$

First by (33), we know the $(m - |A_1| + 1)^{\text{th}}$ term in (34) is equal to:

$$\frac{\sum_{q=|A_1|+1}^{\ell} u_i}{m - |A_1|} = \frac{F}{m}. \tag{35}$$

For any z^{th} term with $z < |A_1|$, we have

$$\begin{aligned}\frac{\sum_{i=z+1}^{\ell} u_q}{m-z} &\geq \frac{(|A_1| - z)\frac{F}{m} + \sum_{q=|A_1|+1}^{\ell} u_q}{m-z} \\ &\geq \frac{(|A_1| - z)\frac{F}{m} + (m - |A_1|)\frac{F}{m}}{m-z} = \frac{F}{m},\end{aligned}$$

where the first inequality follows from the definition of A_1 in which each arc capacity is no smaller than F/m, and the second inequality follows from (35).

For any z^{th} term with $z > |A_1|$, we have

$$\begin{aligned}\frac{\sum_{q=z+1}^{l} u_i}{m-z} &= \frac{(m - |A_1|)\frac{F}{m} - \sum_{q=|A_1|+1}^{z} u_q}{m-z} \\ &\geq \frac{(m - |A_1|)\frac{F}{m} - (z - |A_1|)\frac{F}{m}}{m-z} = \frac{F}{m},\end{aligned}$$

where the first equality follows from (35) and the inequality follows from the definition of A_2 in which each arc capacity is no greater than F/m. The above argument implies that the capacity of $(S, \bar{S})$ is equal to F/m. This again implies that we have found an m-route arc-flow whose value achieves the upper bound in (34), and hence the optimality follows. $\square$

APPENDIX D

Proof of Lemma 4.4: Consider the following linear program:

$$\min \sum_{(i,j)\in A'(f^k)} y_{ij} \tag{36}$$

$$\sum_{j\in\delta_i^+} y_{ij} - \sum_{j\in\delta_i^-} y_{ji} = \begin{cases} 2, & i = s \\ 0, & i \neq s,\ t \\ -2, & i = t \end{cases} \tag{37}$$

$$0 \leq y_{ij} \leq 1 \qquad \forall (i,j) \in A'(f^k). \tag{38}$$

It is obvious that any optimal integer solution of this linear program corresponds to a 2-route in $G'(f^k)$ with a minimum number of arcs.

The dual program of (36)-(38) is:

$$\max\ 2(\pi_s - \pi_t) - \sum_{(i,j)\in A'(f^k)} \gamma_{ij} \tag{39}$$

$$\pi_i - \pi_j - \gamma_{ij} \le 1 \qquad \forall (i,j) \in A'(f^k) \tag{40}$$

$$\gamma_{ij} \ge 0 \qquad \forall (i,j) \in A'(f^k) \tag{41}$$

$$\pi_i \text{ unrestricted}, \qquad \forall i \in V. \tag{42}$$

Let y be an optimal integer solution to the linear program (36)-(38), and (π, γ) be an optimal solution for the dual program (39)-(42). The complementary slackness conditions are the following:

(S_1) $y_{ij} = 0$, i.e., $-y_{ij} = 0 > -1$, implies $\gamma_{ij} = 0$.

(S_2) $y_{ij} = 1 > 0$, implies $\pi_i - \pi_j - \gamma_{ij} = 1$.

We first show the following tow facts. For any arc $(i,j) \in A'(f^{k+1})$:

(C_1) $\pi_i - \pi_j - \gamma_{ij} \le 1$, if $(i,j) \in A'(f^k)$.

(C_2) $\pi_i - \pi_j \le -1 < 1$, if $(i,j) \notin A'(f^k)$.

Indeed if $(i,j) \in A'(f^k)$, then (C_1) follows directly from the constraints of the dual problem (D). If $(i,j) \in A'(f^{k+1}) \backslash A'(f^k)$, then $(j,i) \in A'(f^k)$, and (j,i) is on a shortest 2-route in $G'(f^k)$ according to the flow-updating formulas (23)-(24). Thus by the complementary slackness condition (S_2), $\pi_j - \pi_i - \gamma_{ji} = 1$. Combining the fact that $\gamma_{ji} \ge 0$, we obtain $\pi_i - \pi_j = -1 - \gamma_{ij} \le -1$. This implies ($C_2$).

Now let $\bar{y}$ be an optimal solution in $G'(f^{k+1})$, which corresponds to a 2-route containing two s-t paths P_1 and P_2, and let y^1_{k+1}, y^2_{k+1} be the number of arcs on P_1 and P_2, respectively. Then

$$\alpha_{k+1} = \sum_{(i,j)\in A'(f^{k+1})} \bar{y}_{ij} = y^1_{k+1} + y^2_{k+1}. \tag{43}$$

Let $P_1 = (s = i_0, i_1), (i_1, i_2), \ldots, (i_{\ell_1 - 1}, t = i_{\ell_1})$. Then

$$\begin{aligned} y^1_{k+1} &\ge \sum_{(i_q,i_{q+1})\in P_1 \cap A'(f^k)} \left[\pi_{i_q} - \pi_{i_{q+1}} - \gamma_{i_q i_{q+1}}\right] \\ &+ \sum_{(i_q,i_{q+1})\in P_1 - A'(f^k)} \left[\pi_{i_q} - \pi_{i_{q+1}}\right] \\ &= \pi_s - \pi_t - \sum_{(i_q,i_{q+1})\in P_1 \cap A'(f^k)} \gamma_{i_q i_{q+1}}. \end{aligned}$$

The first inequality comes from (C_1) and (C_2). Similarly, let $P_2 = (s = j_0, j_1), (j_1, j_2), \ldots, (j_{\ell_2 - 1}, t = j_{\ell_2})$. Then we have $y^2_{k+1} \geq \pi_s - \pi_t - \sum_{(j_q, j_{q+1}) \in P_2 \cap A'(f^k)} \gamma_{i_q i_{q+1}}$. So (i) follows by summing up y^1_{k+1} and y^2_{k+1}, and combining (43).

For (ii), first note that we can have equality $\alpha_{k+1} = \alpha_k$ only if each arc in paths P_1 and P_2 is also an arc of $A'(f^k)$ due to (C_2). So if $\alpha_{k+1} = \alpha_k$ then $\beta_{k+1} \leq \beta_k$. Consider the 2-path $\{P_1, P_2\}$ (after merging the parallel arcs generated from the bridges). By the flow updating formulas (23)-(24), at least one arc (i, j) in this 2-path either increases its flow to full capacity (i.e., $f^{k+1}_{ij} = u_{ij}$), or decreases to zero (i.e., $f^{k+1}_{ij} = 0$), or becomes a tight arc (i.e., $f^{k+1}_{ij} = F^{k+1}/2 < u_{ij}$). In the first two cases, (i, j) no longer belongs to $A(f^{k+1})$, and hence no longer belongs to $A'(f^{k+1})$. So $\beta_{k+1} < \beta_k$. We show that the third case cannot happen. Suppose otherwise, we must have $(i, j) \in A_B(f^k)$, the set of bridges at the k^{th} iteration. So there is a bridge in $A(f^{k+1})$ that is tight, and hence the algorithm reaches the optimality. This means $\alpha_{k+1} = +\infty > \alpha_k$, a contradiction with the assumption of $\alpha_{k+1} = \alpha_k$. $\square$