Minimum Ratio Canceling is Oracle Polynomial for Linear Programming, but Not Strongly Polynomial, Even for Networks*

S. Thomas McCORMICK† Akiyoshi SHIOURA‡

January, 1999; Revised December, 1999

Abstract

This paper shows that the minimum ratio canceling algorithm of Wallacher (1989) (and a faster relaxed version) can be generalized to an algorithm for general linear programs with geometric convergence. This implies that when we have a negative cycle oracle, this algorithm will compute an optimal solution in (weakly) polynomial time. We then specialize the algorithm to linear programming on unimodular linear spaces, and to the minimum cost flow and (dual) tension problems. We construct instances proving that even in the network special cases the algorithm is not strongly polynomial.

Keywords: negative cycle canceling algorithm, minimum ratio cycle, linear programming problem, unimodular linear space, minimum cost flow, minimum cost tension.

1 Introduction

A popular class of algorithms for the minimum cost flow problem is the cycle canceling algorithms. This concept originates with Klein [11], and is based on the idea that pushing flow around negative-cost residual cycles will improve the primal objective. The dual version of cycle canceling is cut canceling, originating with Hassin [10]. Here we change node potentials across positive-valued cuts to improve the dual objective. These algorithms are attractive since they are flexible: algorithms with different behaviors can result from choosing different classes of negative cycles or positive cuts to cancel. See Shigeno, Iwata, and McCormick [20] for a recent survey of such algorithms.

A naive idea is to cancel most negative cycles, but these are NP-hard to compute, and do not lead to polynomial algorithms [26]. We could instead use the polynomially-computable most negative

*An extended abstract of this paper appeared in *Proceedings of SODA 2000*, 944–952.

†Faculty of Commerce and Business Administration, University of British Columbia, Vancouver, BC, V6T 1Z2, Canada stmv@adk.commerce.ubc.ca. Part of this research was done while visiting SORIE at Cornell University, and was supported by an NSERC Research Grant; corresponding author.

‡Department of Mechanical Engineering, Sophia University, 7-1 Kioi-cho, Chiyoda-ku, Tokyo 102-8554, Japan, shioura@me.sophia.ac.jp. Part of this research was done while visiting the Faculty of Commerce and Business Administration, University of British Columbia.

family of cycles, but this algorithm is again not polynomial, and the same is true for canceling a most positive (family of) cuts [20] (though canceling relaxed versions of these objects is polynomial [20]). Goldberg and Tarjan [8] popularized the idea of instead choosing to cancel a cycle whose ratio of cost to weight was minimum. They chose the weight of each arc to be one, and showed that the resulting minimum mean cycle canceling algorithm is strongly polynomial.

Wallacher [23] considered several other possibilities for arc weights, and developed polynomial algorithms. One of his polynomial algorithms uses the arc weight given by the reciprocal of the residual capacity of the arc; we call this *minimum ratio* cycle canceling. It turns out that this choice of arc weights makes it easy to prove a weakly polynomial iteration bound on the algorithm. Later, his algorithm was extended to linear programs with totally unimodular matrices by McCormick and Shioura [13], and to general integer programs by Schulz and Weismantel [21]. Recently, Wayne [24] applied the minimum ratio canceling technique to generalized minimum cost flow to give the first polynomial combinatorial algorithm for that problem. We combined Wayne's ideas from [24] and our ideas from [13] to create the present paper. Subsequent to this paper a common generalization of it and [21] showed that the same ideas also work for mixed integer programs [14].

One aim of this paper is to show that Wallacher's minimum ratio canceling algorithm generalizes to an algorithm for general linear programs that geometrically reduces the gap between the objective value of our current solution and the optimal objective value. When we have rational data and an oracle for finding negative cycles we can show that this gap can be driven small enough that it is possible to round to an exact optimal solution in weakly polynomial time. We show that a faster, relaxed version of Wallacher's algorithm also generalizes to linear programming.

An important special case of this is linear programming over a unimodular linear space, namely the solution set of the system $Mx = 0$ given by a totally unimodular matrix M. This is a common generalization of both primal and dual network flow. Karzanov and McCormick [12] showed that minimum mean cycle canceling can be generalized to a polynomial algorithm for linear programming in unimodular linear spaces. We show that the specialization of minimum ratio canceling to this case has a faster iteration bound.

Sections 3 and 4 show how our minimum ratio canceling algorithm further specializes to both the primal and dual of the minimum cost flow problem. The primal specialization is just Wallacher's algorithm [23]. Although most algorithms for the primal minimum cost flow problem have been dualized (e.g., [10, 3, 4, 20]), we believe that our minimum ratio cut canceling tension algorithm is the first dualization of the minimum ratio cycle canceling algorithm. We consider a problem equivalent to dual minimum cost flow called the minimum cost tension problem, which asks for a minimum cost set of potential differences. This problem has a number of applications (see Rockafellar [18, Chapter 7F]) and algorithms of its own (see Hadjiat [9], or Karzanov and McCormick [12]).

Goldberg and Tarjan's minimum mean cycle canceling algorithm [8], and Ervolina and McCormick's dual maximum mean cut canceling algorithm [4], differ from our algorithms only in the choice of weights for the denominator of the ratio. Those algorithms are both strongly polynomial.

On the other hand, the running time analysis of our algorithms is quite similar to analyses of the most helpful cycle canceling algorithm of Weintraub [25] and Barahona and Tardos [2], and the dual most helpful total cut canceling algorithm of Ervolina and McCormick [3]. Queyranne [16], and [3] show that these algorithms are not in general strongly polynomial.

This makes it natural to wonder (at least in the primal and dual network flow cases) if our weakly polynomial bound for minimum ratio canceling can be strengthened to strongly polynomial bound as is the case for many other network flow algorithms. Sections 3 and 4 construct instances, based on the instances in [16] and [3], showing that our minimum ratio canceling algorithms are also not in general strongly polynomial. We follow the proof technique of [16]: a strongly polynomial algorithm is in particular finite even for irrational data. We show that our algorithms take an infinite number of iterations when applied to our instances, thereby proving that the algorithms are not strongly polynomial. This makes it tempting to conjecture that any algorithm based on geometrically decreasing the objective value gap cannot be strongly polynomial.

Proving that the minimum ratio canceling algorithms are not strongly polynomial has implications beyond this paper. Our network flow algorithms are special cases of the Schulz and Weismantel [21], Wayne [24], and mixed integer program [14] results, so our proofs of non-strong polynomiality show that all of these algorithms are also not strongly polynomial.

2 A Minimum Ratio Cycle Canceling Algorithm

In this section, we present a minimum ratio cycle canceling algorithm for the linear programming problem. Let M be an $n \times m$ $(m \geq 2)$ real matrix (we notate M as $n \times m$ instead of the more usual $m \times n$ to match the canonical case where M is the node-arc incidence matrix of a directed graph). Put $J = \{1, 2, \cdots, m\}$. We are given *costs* $c \in \mathbf{R}^J$, and *upper* and *lower bounds* $u \in (\mathbf{R} \cup \{+\infty\})^J$ and $l \in (\mathbf{R} \cup \{-\infty\})^J$. We deal with the following linear programming problem:

$$\text{(LP)} \quad \text{Minimize} \quad c^T x = \sum_{j=1}^{m} c(j)x(j) \quad \text{subject to} \quad x \in P \equiv \{y \in \mathbf{R}^J \mid My = 0,\ l \leq y \leq u\}.$$

We assume that $P \neq \emptyset$ and that (LP) has an optimal solution.

Let $x \in P$. For each $j \in J$, we define *forward* and *back residual capacities* by $r_x^+(j) = u(j) - x(j)$ and $r_x^-(j) = x(j) - l(j)$, respectively. A non-zero vector $d \in \mathbf{R}^J$ is called a *cycle* if it satisfies $Md = 0$ and its support $\mathrm{supp}(d) = \{j \in J \mid d(j) \neq 0\}$ is minimal. The (residual) *capacity* of a cycle $d \in \mathbf{R}^J$ w.r.t. x is defined as

$$\mathrm{cap}_x(d) = \min\left[\min_{j \in J}\left\{\frac{r_x^+(j)}{d(j)}\middle| d(j) > 0\right\}, \min_{j \in J}\left\{-\frac{r_x^-(j)}{d(j)}\middle| d(j) < 0\right\}\right].$$

A cycle d is called *augmenting* (w.r.t. x) if $\mathrm{cap}_x(d) > 0$, and *negative* if $c^T d < 0$. In this paper, we assume that there exists no cycle $d \in \mathbf{R}^J$ such that $\mathrm{supp}(d) \subseteq \{j \in J \mid l(j) = -\infty,\ u(j) = +\infty\}$ and $c^T d = 0$. If such a cycle exists, then we can modify the problem (LP) without changing the optimal value by putting $l(j) = u(j) = 0$ for some $j \in \mathrm{supp}(d)$. Note that this assumption assures the existence of an optimal extreme point of (LP).

Conditions for optimality and unboundedness of (LP) can be stated in terms of cycles:

Theorem 2.1
(i) *$x \in P$ is optimal if and only if there is no negative augmenting cycle w.r.t. x.*
(ii) (LP) *is unbounded if and only if there is a negative cycle with capacity $+\infty$.*

Theorem 2.1 (i) gives a generic algorithm for (LP): we can find an optimal solution by iteratively finding and canceling negative cycles. The performance of this algorithm depends on the class of negative cycles we choose to cancel. We now describe the class of cycles we will choose to cancel.

For $x \in P$, define $w_x^+(j) = 1/r_x^+(j)$ and $w_x^-(j) = 1/r_x^-(j)$ $(j \in J)$. If $r_x^+(j) = +\infty$ (resp. 0), then we define $w_x^+(j) = 0$ (resp. $+\infty$). For any $y \in \mathbf{R}^J$, we define the *weight* $w_x(y)$ of y w.r.t. x as

$$w_x(y) = \sum_{j\in J}\{w_x^+(j)y(j) \mid y(j) > 0\} - \sum_{j\in J}\{w_x^-(j)y(j) \mid y(j) < 0\}.$$

If there is a negative cycle d with $w_x(d) = 0$, then d has infinite capacity, and (LP) is unbounded by Theorem 2.1 (ii), and we have ruled out such instances. Thus $w_x(d) > 0$ for all negative cycles d. If x is feasible but non-optimal to (LP) (so that some negative augmenting cycle w.r.t. x exists), then a *minimum ratio cycle* w.r.t. x is a (negative) augmenting cycle d minimizing the ratio $c^T d/w_x(d)$. Note that if d is a negative augmenting cycle, then $-\infty < c^T d/w_x(d) < 0$.

Any vector $y \in \mathbf{R}^J$ with $My = 0$ can be represented as a nonnegative combination of cycles $d_1, d_2, \cdots, d_k$ $(k \le m)$ such that for $i = 1, \cdots, k$ and $j \in J$, if $d_i(j) > 0$ (resp. < 0) then $y(j) > 0$ (resp. < 0). This representation is called a *conformal realization* of y [18]. This implies that a minimum ratio cycle w.r.t. a feasible, non-optimal x is an optimal solution of the following fractional problem:

$$\begin{array}{lll} (\mathrm{MRC}_x) & \text{Minimize} & c^T y/w_x(y) \\ & \text{subject to} & My = 0, \\ & & y(j) \le 0 \text{ if } r_x^+(j) = 0, \quad y(j) \ge 0 \text{ if } r_x^-(j) = 0, \\ & & y \in \mathbf{R}^J. \end{array}$$

That is, for an optimal solution $y^* \in \mathbf{R}^J$ of (MRC_x), if y^* is decomposed into cycles $d_1, d_2, \cdots, d_k$ in a conformal realization, then each d_i is also optimal for (MRC_x).

Algorithm MinRat is described as follows:

Algorithm MinRat
Step 0: Find a feasible solution $x = x^0 \in P$ of (LP).
Step 1: If there is no negative augmenting cycle, then stop; the vector x is optimal.
Step 2: Find a minimum ratio cycle $d \in \mathbf{R}^J$ w.r.t. x. Let $\alpha = \mathrm{cap}_x(d)$.
Step 3: Cancel d by setting $x' = x + \alpha d$. Go to Step 1.

The next lemma shows that MinRat geometrically reduces the gap between the current objective value and the optimal objective value, which is key to proving that it terminates in a weakly polynomial number of iterations.

Lemma 2.2 *Let $x, x' \in P$ be feasible solutions in the current and next iterations of* MinRat*, respectively, and $d \in \mathbf{R}^J$ be the cycle chosen in Step 2. Also, let $x^* \in \mathbf{R}^J$ be an optimal solution of* (LP). *Then we have*

$$\text{(i)} \quad \frac{c^T(x'-x^*)}{c^T(x-x^*)} \le 1 - \frac{1}{m}, \quad \textit{and} \quad \text{(ii)} \quad \frac{c^T d}{w_x(d)} \le \frac{c^T(x^*-x)}{m}.$$

Proof. Claim (i) is equivalent to

$$\frac{c^T(x'-x)}{c^T(x^*-x)} \ge \frac{1}{m}. \tag{1}$$

Since $-r_x^-(j) \le x^*(j) - x(j) \le r_x^+(j)$ for all $j \in J$, the vector $x^* - x$ is a feasible solution of (MRC$_x$) with

$$w_x(x^* - x) \le m. \tag{2}$$

Since $x^* - x$ is feasible for (MRC$_x$) and d is optimal for (MRC$_x$),

$$\frac{c^T(x'-x)}{w_x(x'-x)} = \frac{c^T d}{w_x(d)} \le \frac{c^T(x^*-x)}{w_x(x^*-x)} \quad \Longrightarrow \quad \frac{c^T(x'-x)}{c^T(x^*-x)} \ge \frac{w_x(x'-x)}{w_x(x^*-x)}. \tag{3}$$

The definition of α implies that

$$w_x(x'-x) = w_x(\alpha d) = \alpha w_x(d) \ge 1. \tag{4}$$

Combining (2), (3), and (4) yields (1). Claim (ii) follows from (2) and (3). ∎

Lemma 2.2 will allow us to get very close to an optimal extreme point. To get an exact optimal solution we will round an approximate solution using only m calls to the oracle (this idea originated in Wayne [24]). This is based on the following lemma.

Lemma 2.3 *A vector $x \in P$ is an extreme point of P if and only if there exists no cycle $d \in \mathbf{R}^J$ with* $\text{supp}(d) \subseteq \{j \in J \mid l(j) < x(j) < u(j)\}$.

Based on Lemma 2.3, we can round a non-extreme point $x \in P$ by iteratively finding a cycle $d \in \mathbf{R}^J$ such that $\text{supp}(d) \subseteq \{j \in J \mid l(j) < x(j) < u(j)\}$ and $c^T d \le 0$, and putting $x := x + \alpha d$ with $\alpha = \text{cap}_x(d)$. We see from our assumptions that $\text{cap}_x(d)$ is finite for such d. Since the cardinality of $\{j \in J \mid l(j) < x(j) < u(j)\}$ decreases in each iteration, this procedure terminates in at most m iterations.

From now on we assume that entries of the matrix M and the vectors l, u, and c are rational numbers. As in Grötschel, Lovász and Schrijver [5], we define the *encoding length* of numbers and vectors as follows. The encoding length of a rational number r is

$$\langle r \rangle = 2 + \lceil \log(|p|+1) \rceil + \lceil \log(|q|+1) \rceil,$$

where p and q (> 0) are relatively prime integers such that $r = p/q$. The encoding length of a rational vector $x \in \mathbf{Q}^n$ is $\langle x \rangle = \sum\{\langle x(j) \rangle \mid j \in J\}$. We put

$$\langle B \rangle = \max\{\langle M(i,j) \rangle \mid i = 1, \cdots, n,\ j \in J\}, \quad \langle C \rangle = \max\{\langle c(j) \rangle \mid j \in J\},$$
$$\langle U \rangle = \max\left[\max\{\langle l(j) \rangle \mid j \in J,\ l(j) > -\infty\},\ \max\{\langle u(j) \rangle \mid j \in J,\ u(j) < +\infty\}\right].$$

Put $\varphi = m\langle B \rangle + \langle U \rangle$. Grötschel et al. [5] shows that there exists an feasible solution x^0 and an optimal solution x^* of (LP) such that $\langle x^0 \rangle$ and $\langle x^* \rangle$ are at most $4m^2\varphi$. We assume that we have an oracle that computes such an x^0 in Step 0 (in practice x^0 might come from a Phase I procedure), and that computes a minimum ratio cycle in Step 2.

Theorem 2.4 *If the matrix M and the vectors l, u are rational, then* MinRat *finds an optimal solution in* $O(m^4\langle B\rangle + m^3\langle U\rangle + m^2\langle C\rangle)$ *iterations.*

Proof. This proof is similar to Wayne [24, Theorem 5].

Put $\epsilon = 2^{-(m\langle C\rangle + 4m^2\varphi)}$. We first find a vector $\tilde{x} \in \mathbf{R}^J$ with $c^T(\tilde{x} - x^*) < \epsilon^2/2$ by using MinRat. Lemma 2.2 (i) implies that MinRat requires at most k iterations to compute $\tilde{x}$, where k satisfies $(1 - 1/m)^k c^T(x^0 - x^*) < \epsilon^2/2$. This yields that $k = O(m^4\langle B\rangle + m^3\langle U\rangle + m^2\langle C\rangle)$.

Suppose that $\bar{x}$ is a non-optimal extreme point and that x^* is an optimal extreme point. Since the denominators of the rational numbers $c^T\bar{x}$ and $c^T x^*$ are at most $1/\epsilon$, we have $c^T\bar{x} > c^T x^* + \epsilon^2$. Thus any extreme point $\hat{x}$ with $c^T\hat{x} \le c^T\tilde{x}$ must be optimal. We can obtain such an optimal extreme point $\hat{x}$ of P by canceling cycles at most m times, as explained after Lemma 2.3. ∎

Therefore MinRat is an oracle weakly polynomial algorithm for linear programming. In some applications of this algorithm a special-purpose algorithm is available to solve the minimum ratio subproblem, and a reasonable initial solution is easily available. See Sections 3 and 4 below for such subroutines for primal and dual network flow, or Wayne [24] for such a subroutine for generalized network flow.

2.1 A Faster Relaxed Version of the Algorithm

Wallacher [23] showed that when canceling min ratio cycles for minimum cost flow, it is possible to replace the (bottleneck) min ratio cycle computation with a much faster negative cycle computation, while keeping essentially the same iteration bound. Wayne [24], and Schulz and Weismantel [21] showed that the same idea works for generalized flows and integer programming. Here we show that this idea extends to linear programming.

Each *scaling phase* has a fixed parameter value μ which is at most half the min ratio value at the beginning of the phase. For cycle d in $\mathbf{R}^J$, we define the *modified cost* of d by $c_\mu(d) = c^T d - \mu w_x(d)$. Then it is easy to see that d has negative modified cost if and only if $c^T d/w_x(d) < \mu$. Such a d can be computed by a fast negative cycle subroutine, which is typically much faster than a min ratio cycle subroutine. For example, for generalized flow a negative cycle is a factor of about $O(n)$ faster to compute than a min ratio cycle [24].

We keep canceling negative cycles w.r.t. c_μ in the scaling phase until none are left (i.e., μ is at most the current min ratio value), at which point we set $\mu \leftarrow \mu/2$, and start a new phase. We call this version *relaxed* MinRat, or RMinRat.

Lemma 2.5 *There are at most* $2m$ *cancellations per scaling phase of* RMinRat.

Proof. First we show that canceling negative cycle d (w.r.t. c_μ) reduces the objective value by less than μ. Rescale d so that $\mathrm{cap}_x(d) = 1$, implying that the objective value decreases by $c^T d$. In this rescaling $w_x(d) \ge 1$ since at least one j becomes saturated. Thus $c^T d < \mu w_x(d) \le \mu$.

Let x^0 be the solution at the beginning of the phase, x^* be an optimal solution, and μ^0 be the min ratio value for x^0. Lemma 2.2 (ii) implies that $\mu^0 \le c^T(x^* - x^0)/m$, or $c^T(x^* - x^0) \ge m\mu^0 \ge 2m\mu$. Since each cancellation in the phase brings $c^T x$ at least μ closer to $c^T x^*$, there can be at most $2m$ cancellations in the phase. ∎

Theorem 2.6 RMinRat *finds an optimal solution in* $O(m^4\langle B\rangle + m^3\langle U\rangle + m^2\langle C\rangle)$ *cancellations.*

Proof. It takes $O(m)$ cancellations in one scaling phase of RMinRat to reduce our upper bound on $c^T(x - x^*)$ from $2m\mu$ to $2m(\mu/2)$, i.e., by a factor of two. It also takes $O(m)$ cancellations of MinRat to achieve the same reduction, so they enjoy the same iteration bound. ∎

There are corresponding relaxed versions of all our subsequent specializations of MinRat.

2.2 Specialization to Linear Programming on Unimodular Spaces

We now restrict M to be an $n \times m$ $(m \geq 2)$ totally unimodular matrix, i.e., any subdeterminant of M is equal to either 0, +1, or −1. We denote this specialization of (LP) by (LPU). We will use several well-known facts on totally unimodular matrices (see, e.g., Schrijver [19]). The feasibility and existence of an optimal solution for (LPU) can be checked in polynomial time with an algorithm of Tardos [22]. Now M being totally unimodular implies that any cycle is a multiple of a $\{0, \pm 1\}$ vector, and hence we regard a cycle as a $\{0, \pm 1\}$ vector. This implies that if the initial x^0 is integral and l and u are integral, then the current solution x is integral in each iteration of MinRat.

Define $C = \max_{j \in J} |c(j)|$,

$$J^L = \{j \in J \mid l(j) > -\infty\},\ J^U = \{j \in J \mid u(j) < +\infty\},\ U = \max\left\{\max_{j \in J^L} |l(j)|, \max_{j \in J^U} |u(j)|\right\}.$$

In this case we can get a sharper bound on the initial gap:

Lemma 2.7 *There exists an optimal solution* $x^* \in \mathbf{R}^J$ *of* (LPU) *with* $|x^*(j)| \leq mU$ *for all* $j \in J$.

Proof. Let $x' \in \mathbf{R}^J$ be an optimal solution of (LPU), and let $x' = \sum_{i=1}^k \beta_i d_i$ $(\beta_i \geq 0)$ be a conformal realization of x'. For $i = 1, \cdots, k$, define $J_i = \{j \in J^U \mid d_i(j) = +1\} \cup \{j \in J^L \mid d_i(j) = -1\}$. Then we claim that

(i) if $J_h \neq \emptyset$, then $\beta_h \leq U$,

(ii) if $J_h = \emptyset$ and $\beta_h > U$, then $x'' = \sum_{i \neq h} \beta_i d_i + U d_h$ is also an optimal solution.

The claim implies that $x^* = \sum_{i=1}^k \min\{\beta_i, U\} d_i$ is an optimal solution with $|x^*(j)| \leq mU$ for all $j \in J$.

To prove claim (i), let $j_0 \in J_h$ with $d_h(j_0) = +1$. Then $x'(j_0) > 0$ and $d_i(j_0) \geq 0$ for all i, which implies that $\beta_h \leq x'(j_0) \leq u(j_0) \leq U$. The case where $d_h(j_0) = -1$ is similar.

Now we show claim (ii). By Lemma 2.1 (ii), d_h is a nonnegative cycle, which implies $c^T x'' \leq c^T x'$. Hence, it suffices to show that $l(j) \leq x''(j) \leq u(j)$ for all $j \in J$. If $d_h(j) = +1$, then $l(j) \leq U \leq x''(j) < +\infty = u(j)$ holds. The other cases are similar. ∎

We now assume that our oracle finds an initial x^0 satisfying the bound of Lemma 2.7 and computes minimum ratio cycles.

Theorem 2.8 *If* c*,* u*, and* l *are integral and the initial feasible solution* x^0 *is integral, then* MinRat *terminates in* $O(m \log(mCU))$ *iterations.*

Proof. The assumption on x^0 and Lemma 2.7 imply that $c^T(x^0 - x^*)$ is bounded by $2m^2CU$. Since $c^T(x - x^*) < 1$ implies that x is optimal, Lemma 2.2 (i) implies that the algorithm terminates in at most k iterations, where k satisfies $(1 - 1/m)^k(2m^2CU) < 1$. This implies that $k < 3m \ln(mCU)$. ∎

Therefore MINRAT is a faster oracle weakly polynomial algorithm for (LPU) than for (LP). We again point out that special-purpose subroutines are available for important special cases.

3 Specialization to the Minimum Cost Flow Problem

We are given a directed graph $G = (N, A)$ with $|N| = n$ and $|A| = m$, upper and lower bounds $u \in (\mathbf{R} \cup \{+\infty\})^A$ and $l \in (\mathbf{R} \cup \{-\infty\})^A$, and costs $c \in \mathbf{R}^A$. For each node $v \in N$, we denote the sets of arcs leaving and entering v by $\delta^+ v$ and $\delta^- v$, respectively. Then the *minimum cost flow problem* is

$$\begin{array}{lll} \text{(MCF)} \quad \text{Minimize} & \sum_{a \in A} c(a)x(a) & \\ \text{subject to} & \sum_{a \in \delta^+ v} x(a) - \sum_{a \in \delta^- v} x(a) = 0 & (v \in N), \\ & l(a) \le x(a) \le u(a) & (a \in A). \end{array}$$

If $M \in \mathbf{R}^{n \times m}$ is the node-arc incidence matrix of G, then the flow conservation constraints can be written as $Mx = 0$. Since this M is totally unimodular, (MCF) is a special case of (LPU).

It is well-known that the (matrix) cycles of M are in 1-1 correspondence with the (graph) cycles of G. Thus the whole machinery of minimum ratio canceling for (LPU) carries over to (MCF) without even needing to change terminology. We call the version of MINRAT specialized to (MCF) MINRAT-F. The specialization of Theorem 2.8 to this case is:

Theorem 3.1 (Wallacher [23]) *If c, u, and l are integral, and the initial feasible flow x^0 is integral and satisfies $|x^0(a)| \le mU$ for all $a \in A$, then* MINRAT-F *terminates in* $O(m \log(nCU))$ *iterations.*

Such an x^0 can be found in strongly polynomial time via max flow (Ahuja, Magnanti, and Orlin [1]). A minimum ratio cycle can be found in $O(\min\{n^3(\log n)^2, n^3 \log n \log\log n\})$ time by Megiddo's parametric search [15]. Hence, MINRAT-F runs in (non-oracle) weakly polynomial time.

In order to show that MINRAT-F is not strongly polynomial, we now show a real-valued instance for which MINRAT-F does not terminate. Consider the network shown in Figure 1, which is a modification of the one in Queyranne [16]. Each arc drawn by a bold arrow actually consists of three arcs in series with the same cost and bounds. The lower bound is 0 for each arc, and the upper bound is indicated in the figure. Define $r = (\sqrt{5} - 1)/2$. The symbols 'inf', S_1, S_2, and S_3 represent the values $+\infty$, $(1 + r)/2$, $1/2$, and $r/2$. These numbers satisfy the identities

$$S_1 - r = r^3 S_1, \quad S_2 - r^2 = r^3 S_2, \quad \text{and} \quad S_3 - r^3 = r^3 S_3,$$

and the inequalities $1 > S_1 > r > S_2 > r^2 > S_3 > r^3 > S_1 - r = r^3 S_1$, etc., which are needed to understand the behavior of MINRAT-F on the network. The arc (t, s) has cost -1, and the four arcs $(s, 4), (s, 7), (5, t)$, and $(8, t)$ have large negative costs. Each of the other arcs has cost 0.

When MINRAT-F is applied to this network with the initial feasible flow $x^0 = \mathbf{0} \in \mathbf{R}^A$, it is easy to see that the first iteration chooses the cycle with arcs $\{(s, 7), (7, 8), (8, t), (t, s)\}$, and the second iteration chooses the cycle with arcs $\{(s, 4), (4, 5), (5, t), (t, s)\}$. These two cancellations saturate the

Table 1: Change of flow when MinRat-F is applied to bad instance

iter.	MRC	$\mathrm{cap}_x(Q)$	flow at the beginning of the iteration					
			key arcs			arcs with upper bound		
			$(1,2)$	$(4,5)$	$(7,8)$	S_1	S_2	S_3
$3k$	Q_1	r^{3k-2}	0	r^{3k-2}	r^{3k-3}	$S_1(1-r^{3k-3})$	$S_2(1-r^{3k-3})$	$S_3(1-r^{3k-3})$
$3k+1$	Q_2	r^{3k-1}	r^{3k-2}	0	r^{3k-1}	$S_1(1-r^{3k})$	$S_2(1-r^{3k-3})$	$S_3(1-r^{3k-3})$
$3k+2$	Q_3	r^{3k}	r^{3k}	r^{3k-1}	0	$S_1(1-r^{3k})$	$S_2(1-r^{3k})$	$S_3(1-r^{3k-3})$
$3k+3$	Q_1	r^{3k+1}	0	r^{3k+1}	r^{3k}	$S_1(1-r^{3k})$	$S_2(1-r^{3k})$	$S_3(1-r^{3k})$

arcs $(s,4)$, $(s,7)$, $(5,t)$, and $(8,t)$. The flow on the three *key* arcs is now $x_{12}=0$, $x_{45}=r$, and $x_{78}=1$, and all other arcs have flow 0.

In the following iterations, the arcs with capacities 1 and r are never contained in canceled cycles, so the only negative-cost residual arc available is (t,s). Thus each further cycle Q consists of a shortest s-t path (w.r.t. lengths w_x), together with arc (t,s). Since $\frac{1}{r}+\frac{3}{S_1}<\frac{3}{S_2}$ (this is why each bold arc in Figure 1 is a series of three arcs), the subpath containing the backward $(4,5)$ arc and the forward S_1-arc $(4,6)$ is shorter than the $(5,6)$ S_2 arc, so that the first such shortest path uses only arcs with capacities S_1 and $+\infty$. Since $\frac{1}{r^2}+\frac{3}{S_2}<\frac{3}{S_3}$ and $\frac{1}{r^3}+\frac{3}{S_3}<\frac{3}{r^3S_1}$, the next two cycles use S_2 and S_3 arcs respectively, and we have a pattern that repeats blocks of three iterations. For $g=1,2,3$ the $3k-1+g$-th iteration $(k\geq 1)$ cancels the cycle

$$\begin{aligned}
Q_1 &= \{(s,1),(1,2),(2,3),(3,5),(4,5),(4,6),(6,8),(7,8),(7,t),(t,s)\},\\
Q_2 &= \{(s,2),(1,2),(1,3),(3,4),(4,5),(5,6),(6,8),(7,8),(7,t),(t,s)\},\\
Q_3 &= \{(s,2),(1,2),(1,3),(3,5),(4,5),(4,6),(6,7),(7,8),(8,t),(t,s)\},
\end{aligned}$$

where Q_g consists of all arcs with capacities S_g, the three key arcs, and (t,s). The cycle Q_g contains the key arc with flow 0 at the beginning of its iteration forward, and the two key arcs with positive flow backward. The iteration pushes r^{3k-3+g} units of flow on Q_g, which is determined by the backward key arc with a smaller flow hitting its lower bound of 0. The flow value on the key arcs and S_1, S_2, and S_3 arcs changes as noted in Table 1.

Therefore, MinRat-F requires an infinite number of iterations. This proves:

Theorem 3.2 *Algorithm* MinRat-F *is not a finite algorithm for real data, so it is not a strongly polynomial algorithm.*

4 Specialization to the Minimum Cost Tension Problem

The minimum cost tension problem has an identical setup to the minimum cost flow problem of the previous section, except that here we are looking for an optimal set of *node potentials* $\pi\in\mathbf{R}^N$ instead of an optimal flow $x\in\mathbf{R}^A$. Given potentials π, we define the associated *potential difference*, or *tension* $\tau\in\mathbf{R}^A$ by $\tau(a)=\pi_j-\pi_i$ for $a=(i,j)\in A$.

The *minimum cost tension problem* is

$$\begin{array}{lll}\text{(MCT)} & \text{Minimize} & \sum_{a\in A} c(a)\tau(a) \\ & \text{subject to} & l(a) \le \tau(a) \le u(a) \quad (a \in A), \\ & & \tau \in \mathbf{R}^A : \text{a tension.}\end{array}$$

Rockafellar [18] shows that (MCT) is equivalent to the dual of (MCF). Define T to be the matrix whose rows are the incidence vectors of all fundamental cycles w.r.t. a maximal forest of G. Then τ is a tension if and only if $T\tau = 0$, and T is a totally unimodular matrix. Therefore (MCT) is also a special case of (LPU).

To describe the cycles of T we need some notation. A *cut* is a nonempty proper subset of N. For any cut $C \subset N$, denote by δ^+C (resp. δ^-C) the sets of arcs leaving (resp. entering) C. Then a vector $d \in \{0, \pm 1\}^A$ is a cycle of T if and only if there is some cut C such that $d(a) = +1$ exactly on δ^+C, and $d(a) = -1$ exactly on δ^-C [18].

With this understanding, given a tension $\tau \in \mathbf{R}^A$, we can now define residual capacities for arcs w.r.t. τ, the capacity of a cut C w.r.t. τ, the weight of C w.r.t. τ, and the cost of C. A cut is *augmenting* w.r.t. τ if $r_\tau(a) > 0$ on δ^+C and $r_\tau(a) < 0$ on δ^-C. We cancel an augmenting cut by increasing π_i by $\text{cap}_\tau(C)$ for $i \in C$, and leaving π_i the same for $i \notin C$. Algorithm MinRat specializes into canceling minimum ratio cuts, which we call MinRat-T.

The specialization of Theorem 2.8 to this case is:

Theorem 4.1 *If c, u, and l are integral, and the initial tension τ^0 is integral and satisfies $|\tau^0(a)| \le mU$ for all $a \in A$, then* MinRat-T *terminates in* $O(m \log(nCU))$ *iterations.*

Such a τ^0 can be found in strongly polynomial time (Ahuja, Magnanti, and Orlin [1]). The current best (strongly) polynomial bound for finding a minimum ratio cut is $O(mT_{MC}(m,n))$ by Radzik [17]. Here $T_{MC}(m,n)$ is the time required for computing a minimum cut in a directed graph, and $T_{MC}(m,n) = O(\min\{mn\log(n^2/m), \min\{n^{2/3}, \sqrt{m}\}m\log(n^2/m)\log U\})$ by Goldberg–Tarjan [7] and Goldberg–Rao [6]. Therefore, MinRat-T runs in (non-oracle) weakly polynomial time.

In order to show that MinRat-T is not strongly polynomial, we now show a real-valued instance for which MinRat-T does not terminate. The network in Figure 2 is just the planar dual of the one in Figure 1. Each arc drawn by a bold arrow actually consists of three parallel arcs with the same capacity and cost. The lower bound is 0 for each arc, and the upper bound is shown on each arc. The symbols 'inf', S_1, S_2, and S_3 again represent the values $+\infty$, $(1+r)/2$, $1/2$, and $r/2$. The arc (s,t) has cost -1, and four arcs $(2,1), (s,2), (15,t)$, and $(16,15)$ have large negative cost. Each of the other arcs has cost 0.

Planar duality says that the dual of cycles are cuts, and the dual of a flow is a tension, so that the planar dual of a shortest path w.r.t. weights w_x is a minimum weight cut w.r.t. weights w_τ. Therefore the behavior of MinRat-T on the network in Figure 2 exactly mimics the behavior of MinRat-F on the network in Figure 1. Thus MinRat-T also requires an infinite number of iterations, proving

Theorem 4.2 *Algorithm* MinRat-T *is not a finite algorithm for real data, so it is not a strongly polynomial algorithm.*

Acknowledgments

We are grateful to Tomomi Matsui, Maurice Queyranne, Andreas Schulz, Maiko Shigeno, and Kevin Wayne for valuable discussion and comments.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows — Theory, Algorithms, and Applications*, Prentice Hall (1993).

[2] F. Barahona and É. Tardos, "Note on Weintraub's minimum-cost circulation algorithm," *SIAM J. Comp.* **18**, 579–583 (1989).

[3] T. R. Ervolina and S. T. McCormick, "Canceling most helpful total cuts for minimum cost network flow," *Networks* **23**, 41–52 (1993).

[4] T. R. Ervolina and S. T. McCormick, "Two strongly polynomial cut canceling algorithms for minimum cost network flow," *Discrete Appl. Math.* **46**, 133–165 (1993).

[5] M. Grötschel, L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, Berlin (1988).

[6] A. V. Goldberg and S. Rao, "Beyond the flow decomposition barrier," *J. ACM* **45**, 753–782 (1998).

[7] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum flow problem," *J. ACM* **35**, 921–940 (1988).

[8] A. V. Goldberg and R. E. Tarjan, "Finding minimum-cost circulations by canceling negative cycles," *J. ACM* **36**, 873–886 (1989).

[9] M. Hadjiat, "Un algorithme fortement polynomial pour la tension de coût minimum basé sur les cocycles de coûts moyens minimums," Technical Report, Groupe Intelligence Artificielle, Faculté des Sciences de Luminy, Marseille, France (1994). [In French.]

[10] R. Hassin, "The minimum cost flow problem: a unifying approach to dual algorithms and a new tree-search algorithm," *Math. Programming* **25**, 228–239 (1983).

[11] M. Klein, "A primal method for minimal cost flows," *Management Sci.* **14**, 205-220 (1967).

[12] A. V. Karzanov and S. T. McCormick, "Polynomial methods for separable convex optimization in unimodular spaces with applications," *SIAM J. Comp.* **26**, 1245–1275 (1997).

[13] S. T. McCormick and A. Shioura, "A minimum ratio cycle canceling algorithm for linear programming problems with application to network optimization problems," manuscript, November 1996.

[14] S.T. McCormick, A. Schulz, A. Shioura, and R. Weismantel, "An oracle-polynomial primal augmentation algorithm for mixed integer programs," manuscript, May 2000.

[15] N. Megiddo, "Applying parallel computation algorithms in the design of serial algorithms," *J. ACM* **30**, 852–865 (1983).

[16] M. Queyranne, "Theoretical efficiency of the algorithm 'Capacity' for the maximum flow problem," *Math. Oper. Res.* **5**, 258–266 (1980).

[17] T. Radzik, "Parametric flows, weighted means of cuts, and fractional combinatorial optimization," in: P. Pardalos (ed.), *Complexity in Numerical Optimization*, World Scientific, 351–386 (1993).

[18] R. T. Rockafellar, *Network Flows and Monotropic Optimization*, John Wiley & Sons (1984).

[19] A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley & Sons (1986).

[20] M. Shigeno, S. Iwata, and S.T. McCormick, "Relaxed most negative cycle and most positive cut canceling algorithms for minimum cost flow," *Math. of OR*, **25**, 76–104 (2000).

[21] A. S. Schulz and R. Weismantel, "A polynomial-time augmentation algorithm for integer programming," Manuscript, Fachbereit Mathmatik, Technische Universität Berlin (1998).

[22] É. Tardos, "A strongly polynomial algorithm to solve combinatorial linear programs," *Oper. Res.* **34**, 250–256 (1986).

[23] C. Wallacher, "A generalization of the minimum-mean cycle selection rule in cycle canceling algorithms," unpublished manuscript, Institute für Angewandte Mathematik, Technische Universität Braunschweig (1989).

[24] K. Wayne, "A polynomial combinatorial algorithm for generalized minimum cost flow," *Proceedings of the 31st Annual ACM Symposium on the Theory of Computing* (1999).

[25] A. Weintraub, "A primal algorithm to solve network flow problems with convex costs," *Management Sci.* **21**, 87–97 (1974).

[26] N. Zadeh, "More pathological examples for network flow problems," *Math. Programming* **5**, 217–224 (1973).

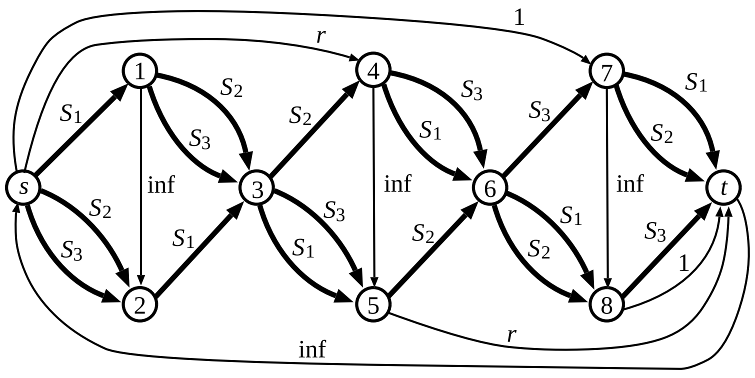

Figure 1: A bad instance for MinRat-F

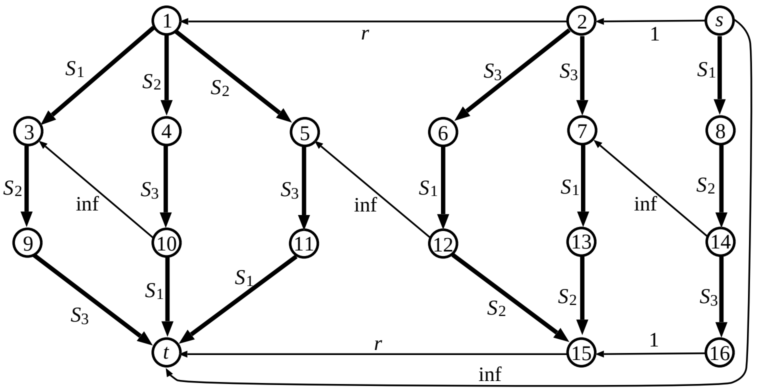

Figure 2: A bad instance for MinRat-T