Relaxed Most Negative Cycle and Most Positive Cut Canceling Algorithms for Minimum Cost Flow

Maiko Shigeno *  Satoru Iwata †  S. Thomas McCormick ‡

August 9, 2007

Abstract

This paper presents two new scaling algorithms for the minimum cost network flow problem, one a primal cycle canceling algorithm, the other a dual cut canceling algorithm. Both algorithms scale a relaxed optimality parameter, and create a second, inner relaxation. The primal algorithm uses the inner relaxation to cancel a most negative node-disjoint family of cycles w.r.t. the scaled parameter, the dual algorithm uses it to cancel most positive cuts w.r.t. the scaled parameter. We show that in a network with \( n \) nodes and \( m \) arcs, both algorithms need to cancel only \( O(mn) \) objects per scaling phase.

Furthermore, we show how to efficiently implement both algorithms to yield weakly polynomial running times that are as fast as any other cycle or cut canceling algorithms. Our algorithms have potential practical advantages compared to some other canceling algorithms as well.

Along the way, we give a comprehensive survey of cycle and cut canceling algorithms for min-cost flow. We also clarify the formal duality between cycles and cuts.

Key words: minimum cost flow, scaling, relaxed optimality, cycle canceling, cut canceling

---

*Institute of Policy and Planning Sciences, University of Tsukuba, Tsukuba, Ibaraki 305, Japan. Research supported in part by a Grant-in-Aid of the Ministry of Education, Science, Sports and Culture of Japan, and University of Tsukuba Research Projects.

†Division of Systems Science, Graduate School of Engineering Sciences, Osaka University, Toyonaka 560, Japan. Research supported in part by a Grant-in-Aid of the Ministry of Education, Science, Sports and Culture of Japan.

‡Faculty of Commerce and Business Administration, University of British Columbia, Vancouver, BC V6T 1Z2 Canada. Research supported by an NSERC Operating Grant; part of this research was done during a visit to RIMS at Kyoto University.
1 Introduction

Finding a feasible flow in a network of minimum cost, the Minimum-Cost Flow (MCF) problem, is among the most important discrete optimization problems. It has many practical applications, and a large number of algorithms have been developed for it, see, e.g., Ahuja, Magnanti, and Orlin (1997).

The conceptually simplest MCF algorithms are based on pushing flow along residual (augmentable) cycles with net negative cost (“negative cycle canceling”) in the primal case, and changing dual variables across cuts with a net positive contribution to the dual objective (“positive cut canceling”) in the dual case. The generic negative cycle canceling algorithm is sometimes called Klein’s Algorithm because it was first formalized in Klein (1967). The generic positive cut canceling algorithm is sometimes called Hassin’s Algorithm because it was first formalized in Hassin (1983).

This paper proposes new cycle and cut canceling algorithms. We choose to cancel cycles and cuts generically called relaxed min/max cycles/cuts, which are a compromise between min mean cycles/max mean cuts, and most negative cycles/most positive cuts (see below for definitions of these). This allows us to achieve the small number of iterations associated with mean cycles and cuts, together with the efficient time per iteration associated with most negative cycles/most positive cuts. We end up with algorithms whose weakly polynomial running times are as fast as the fastest known cycle/cut canceling algorithms.

Our paper makes three additional contributions. First, we give the most comprehensive survey of cycle and cut canceling algorithms we know of, summarized in Table 1 below. Second, we clarify the extent to which cycles and cuts are dual objects, i.e., while cycles have minimal support, the same is not generally true of cuts. Third, using this clearer view of cycles and cuts, we propose using an assignment subproblem as a way to compute a most negative node-disjoint family of cycles, as a(n imperfect) dual notion to computing a most positive cut.

1.1 Are Cycles and Cuts Minimal Support Directions?

We first deal with an important technicality concerning cycles and cuts. Let $x$ be a flow that satisfies conservation and $A$ be the node-arc incidence matrix of the network. If $d$ is a non-zero vector satisfying $Ad = 0$, then $x + \alpha d$ also satisfies conservation for any scalar $\alpha$, i.e., $d$ is a direction vector. Cycles are minimal in the sense that any (non-zero) direction vector with minimal support must be a scalar multiple of the incidence vector of a cycle. We call a cycle a circuit when we want to emphasize this minimality. Some algorithms we consider here actually cancel node-disjoint families of cycles; we often call these “cycle canceling algorithms” anyway for simplicity.

On the dual side, let $\tau_{ij} = \pi_j - \pi_i$ be a potential difference, or tension. Let $B$ be a matrix whose rows are a basis for the subspace generated by all incidence vectors of cycles, so that $B\tau = 0$. As above, if $Bd = 0$, then $\tau + \beta d$ is again a tension, and such non-zero $d$’s are (dual) direction vectors.

A cut in a network is a nonempty proper subset of nodes. The subset of arcs with one end
in a cut and the other in its complement is called a cutset. The incidence vector of a cutset is a (dual) direction vector. Every minimal-support direction vector is a scalar multiple of the incidence vector of a cutset, but the converse is not true. For example, consider the $k$-star which has central node $s$ and arcs $s \to 1, \ldots, s \to k$. Then the cutset associated with the cut $\{s\}$ is not minimal, since its support is a superset of the support of the cutsets associated with the cuts $\{i\}, i = 1, \ldots, k$. We call a minimal-support cutset a cocircuit. Thus a cut corresponds to a family of cocircuits.

Though it is an easy observation that cycles are minimal but that the cutsets associated with cuts need not be minimal, we managed to overlook this fact before now. This blind spot led to statements such as “Most helpful cycles are NP-Hard to find, whereas most helpful total cuts are easy to find” (Ervolina and McCormick (1993b)), implying that there is an asymmetry in the complexity of cycles and cuts. We will in fact show below in Section 3.1 that both computing a most negative circuit and computing a most positive cocircuit are NP Hard. By contrast, it is known that computing a most negative family of cycles and computing a most positive cut are both easy. This restores the missing symmetry.

1.2 A Survey of Cycle and Cut Canceling Algorithms

Throughout this paper, complexities are expressed in terms of $n$, the number of nodes; $m$, the number of arcs; $C$, the largest absolute cost; and $U$, the largest absolute capacity. The discussion in this and the next section is summarized in Table 1 below.

First we note that there is not a clear distinction between cycle and cut canceling algorithms, and other min cost flow algorithms. The canceling algorithms we survey and develop here all start out with idea of canceling cycles and cuts with particular properties, but (for the sake of efficiency) some of them end up using operations that no longer resemble classical cycle and cut canceling. On the other side, some min cost flow algorithms are not motivated by canceling, but end up using similar operations as our algorithms here. For example, our Blocking Approximate MNDC canceling algorithm ends up looking similar to one of the min cost flow algorithms in Goldberg’s scaling shortest paths paper Goldberg (1995), see Section 6. It could be argued that Goldberg’s algorithms should also be covered here, since his Cut-Relabel operation is a type of cut canceling, and since his min cost flow algorithms cancel cycles. In this sense Goldberg’s algorithms are successive approximation versions of the out-of-kilter method (see Fulkerson (1961) or Minty (1960)), which could also be categorized as both a cycle and cut canceling algorithm. We have decided to not include this type of algorithm in order to keep our survey short.

When all data are integral, then canceling arbitrary cycles/cuts does at least improve the objective function value by one at each iteration, leading to the pseudo-polynomial bound of $O(mCU)$ iterations. We will see below that examples exist showing that this bound can be exponential in both cases, even for reasonable classes of cycles/cuts.

Perhaps the simplest reasonable class is to choose a most negative circuit or most positive cocircuit, or generically, a min/max minimal-support object. This is a non-basic version of Dantzig’s pivot rule. However, most negative circuits are NP Hard to compute (by an easy
reduction from Directed Hamiltonian Path), and Zadeh (1973) has shown that canceling most negative circuits can take an exponential number of iterations. We show in Section 3.1 that it is NP Hard to compute a most positive cocircuit, and Hassin’s (1983) examples show that canceling these is also exponential.

However, there are closely-related objects that are computable in polynomial time. We can use a 0-1 minimum-cost flow subproblem to find a most negative arc-disjoint family of cycles, or an assignment subproblem to find a most negative node-disjoint family of cycles (MNDC), see Section 3.2. However, Zadeh’s (1973) examples still show that canceling each of these is exponential. Similarly, a most positive cut (MPC) can be computed in polynomial time using a max flow subproblem (see Section 3.3), but Hassin (1983) showed that canceling these is exponential.

In Iwata, McCormick, and Shigeno (1999c) we consider two generalizable relaxation possibilities. One possibility leads to thinking that the dual object of MPC is a most negative family of arc-disjoint cycles, and the other leads to a less familiar family of cycles. Neither of these is a family of node-disjoint cycles, or MNDC. However, a MNDC is not only more efficiently computable than a family of arc-disjoint cycles, it also gives a better bound on the number of cancellations. We prefer computational efficiency over formal duality, so we shall use MNDCs as our most negative families of cycles.

For most cycle and cut canceling algorithms, computing the negative cycle or positive cut to be canceled is quite expensive. Network simplex algorithms restrict their choice to fundamental cycles w.r.t. a basic tree, and so achieve quite fast cycle/cut selection times. The trade-off is that they can choose cycles/cuts with residual capacity zero, and so are prone to degenerate pivots. This can even cause cycling, i.e., an infinite sequence of degenerate pivots, see Gassner (1964). Cunningham (1979) developed a simple rule that avoids cycling in primal network simplex; Goldfarb, Hao, and Kai (1990) have more complicated rules to prevent cycling in dual network simplex. Using the classic Dantzig’s Rule for pivot selection, it takes only $O(m)$ time to find a most negative fundamental cycle in the primal, and $O(n)$ time to find a most positive fundamental cut in the dual. However, Zadeh (1973) showed that these algorithms take exponential time in general.

More recent work has developed more sophisticated pivot rules that are provably polynomial time. Orlin (1997) gives a cost scaling primal network simplex algorithm that takes $O(\min(mn \log (nC), m^2 n \log n))$ pivots. Each pivot takes $O(\log n)$ amortized time using a data structure due to Tarjan (1997). Orlin, Plotkin, and Tardos (1993) developed an excess scaling dual network simplex algorithm that runs in $O(m^2 (m + n \log n) \log n)$ time. Armstrong and Jin (1997) improved this to $O(nn(m + n \log n) \log n)$ time. Their algorithm aggregates $O(n)$ pivots into a block pivot, which costs $O(m + n \log n)$ to perform. They do not give a weakly polynomial version of their algorithm, but it is easy to see that the pseudo-polynomial version of their algorithm can be embedded in a scaling framework. Each scaling phase does $O(m)$ block pivots, and there are $O(\log U)$ scaling phases. The strongly polynomial version of their algorithm performs $O(mn \log n)$ block pivots. These Scaling Network Simplex algorithms equal the best running time of any cycle or cut canceling algorithm, and Armstrong and Jin’s strongly polynomial bound is faster than any other cut canceling algorithm.
A third idea is to cancel *mean* cycles or cuts, i.e., objects whose average change in objective value per arc is largest. This is analogous to a steepest edge pivot rule in the $\ell_1$ norm. These methods are guaranteed to produce minimal support objects, i.e., circuits or cocircuits. This approach was originated by Goldberg and Tarjan (1989) with their Min Mean Cycle Canceling algorithm. A dual Max Mean Cut Canceling algorithm was developed by Ervolina and McCormick (1993a). (Hassin (1992) tried a cut canceling algorithm using mean value per node, and showed that it can be exponential; Hadjiat (1994) proposed an algorithm similar to Max Mean Cut Canceling for the closely related max-cost tension problem.) Both algorithms turn out to be strongly polynomial. These are perhaps the conceptually simplest strongly polynomial min-cost flow algorithms known, in part because they do not rely on any sort of scaling.

The worst-case running times of these algorithms is quite large. But Goldberg and Tarjan noticed that the key factors in the convergence proof of Min Mean Cycle Canceling can be achieved in a computationally much more efficient manner. This led them to propose the *(Primal) Cancel and Tighten* (PCT) algorithm, which has a quite good theoretical running time. This was dualized into a *Dual Cancel and Tighten* (DCT) algorithm by Ervolina and McCormick, again with a good running time. (See also Radzik and Goldberg (1994), who improve the bounds on both PCT and DCT and show that these bounds can be tight, and Karzanov and McCormick (1997), who show that all of this can be extended to optimizing separable convex objectives over any totally unimodular matrix.)

PCT and DCT are quite good in theory: they match the best weakly polynomial running time bounds of any known cycle or cut canceling algorithm. Their weakly polynomial bounds are within log factors of the fastest running time of any known min-cost flow algorithm. However, they have some practical difficulties. Any implementation of them must deal with arbitrary fractions in its intermediate computations, even if the original data are all integral. Also, their convergence seems to be quite slow in practice (see McCormick and Liu (1993) for evidence of this for DCT), possibly due to the fact that their very small $(1 - 1/n)$ or $(1 - 1/m)$ factors of decrease in convergence parameter can get lost in round-off error.

A fourth idea is to cancel *most helpful*, or *most improving* cycles or cuts, i.e., those whose cancellation improves the objective value as much as possible. Here it is easy to get a polynomial bound on the number of iterations. Barahona and Tardos (1989) show how to implement a most helpful cycle canceling algorithm proposed by Weintraub (1974) in polynomial time per iteration by canceling most helpful node-disjoint cycles. Similarly, Ervolina and McCormick (1993b) show how a version of canceling most helpful cuts can be implemented in polynomial time.

However, these algorithms have rather slow worst-case complexity. Furthermore, it was shown by Queyranne (1980) that even for the case of max flow problems, Most Helpful Cycle Canceling is not *strongly* polynomial, and by Ervolina and McCormick (1993b) that even for the case of shortest path problems, Most Helpful Cut Canceling is also not strongly polynomial.

A fifth idea is to cancel *min/max ratio* cycles or cuts. These are a generalization of mean cycles/cuts where the denominator is a sum of the inverses of residual capacities. These algorithms always produce circuits/cocircuits. These algorithms have the same polynomial bound on iterations as the most helpful algorithms. The primal version, *Min Ratio* Cycle Canceling was developed by Wallacher (1991), and a dual version, *Max Ratio* Cut Canceling, was developed
by McCormick and Shioura (1999) for the max-cost tension problem (which is closely related to
dual min-cost flow). These algorithms are sometimes faster per iteration than the most helpful
algorithms, but they are still less efficient in theory than PCT and DCT. They are not strongly
polynomial, but (unlike most other network flow algorithms) they generalize to solving mixed
integer programs (with an appropriate oracle, McCormick et al. (1999)).

1.3 Overview of Our Algorithms

With this background we can now describe our algorithms more accurately. The convergence
proofs of Min Mean Cycle/Max Mean Cut Canceling depend on a duality with the notion of
relaxed optimality (also called approximate optimality; see Lemmas 4.2 and 5.2 below). This
is a relaxation of the usual complementary slackness conditions for optimality by a parameter.
The Mean Canceling algorithms converge because canceling these objects produces geometric
decrease in this parameter. It is often said that these algorithms implicitly scale this parameter.

We propose instead to explicitly scale this parameter. Other min-cost flow algorithms do this
also, such as Goldberg and Tarjan’s Successive Approximation algorithm (1990). However, these
algorithms use more complicated operations than just cycle and cut canceling. We then apply a
second, inner relaxation. In the primal case this is an assignment subproblem that allows dual
variables at nodes to have different values for head-incident arcs and tail-incident arcs; in the dual
case it is a max flow subproblem that allows violations of conservation. These allow our Relaxed
Min/Max algorithms to compute a most negative node-disjoint family of cycles/most positive
cut w.r.t. the current scale of the parameter, and cancel that object. Although this is a (simple)
MNDC/MPC computation, we will later show that it is nonetheless a good approximation to min
mean cycle/max mean cut. We will obtain bounds of $O(mn)$ objects canceled per scaling phase
for both primal and dual cases, which already gives us a weakly polynomial bound. Standard
techniques of interspersing a min mean cycle/max mean cut computation from time to time will
give us a strongly polynomial bound. These bounds match the iteration bounds for Min Mean
Cycle/Max Mean Cut Canceling up to a $\log n$ factor.

We then modify our algorithms to efficiently cancel enough cuts/cycles in one operation
that we are able to make significant progress in our inner relaxation’s subproblem. We call
this a Blocking Cancel by analogy with blocking flow methods for max flow (Ahuja, Magnanti,
and Orlin (1993)), which augment on enough paths in one operation to significantly change the
distance label of the source.

To make this “significant progress” large enough, we must either round the data, or further
relax our notion of which arcs are allowed to participate in our cycles/cuts to get approximate
Min/Max canceling algorithms. We prefer the second option since it avoids rounding, thereby
easily allowing the strongly polynomial techniques to carry over. The resulting Blocking Cancel
Approximate Min/Max primal and dual algorithms have the same complexity as PCT and DCT.

Finally, we consider the extent to which our techniques extend to classic pure scaling versions
(i.e., scaling the data directly, instead of an optimality parameter). We find that pure capacity
scaling in our framework yields a slight speed-up that matches the fastest known weakly poly-
nomial capacity scaling algorithms (Edmonds (1972), Armstrong and Jin (1997)), but that pure
cost scaling in our framework appears not to work.

Table 1 summarizes this discussion. For each algorithm considered, we give the number of iterations (number of cycles/cuts canceled) and the time per iteration (complexity of computing this class of cycles/cuts), in both weakly and strongly polynomial versions where applicable. SP is the time to solve a shortest path problem with non-negative costs, MF is the time to solve a max flow problem, and AP is the time to solve an assignment problem. The best current bounds on SP are $O(m + n \log n)$ (Fredman and Tarjan (1987)), $O(m \log \log C)$ (Johnson (1982)), and $O(m + n\sqrt{\log C})$ (Ahuja et al. (1990)). The best current bounds on MF are $O(mn \log(n^2/m))$ (Goldberg and Tarjan (1988)), $O(nm \log(n/m \log n))$ (King, Rao, and Tarjan (1994)), $O(nm \log(2n\sqrt{\log U/m})$ (Ahuja, Orlin, and Tarjan (1989)), and $O(\min\{n^{2/3}, \sqrt{m}\}m \log(n^2/m) \times \log U)$ (Goldberg and Rao (1997)). The best current bounds on AP are $O(nSP)$, and $O(\sqrt{nm \log(nC)})$ (Gabow and Tarjan (1989), Goldberg (1995), Orlin and Ahuja (1992)).

Table 1 also illustrates our original motivation for this research. Note that the best known strongly polynomial time for computing a min mean cycle, $O(nm)$, is the same as the best known time for computing general shortest paths, whereas the best known strongly polynomial bound for max mean cut is $O(nMF)$, which is $n$ times larger than the time to compute a max flow. By contrast, the complexity of most negative node-disjoint cycles is $O(AP)$, which is about the same as most positive cut, i.e., $O(MF)$.

As noted, our Relaxed Min/Max canceling algorithms have about the same iteration bounds as the Mean canceling algorithms, but this apparent computational asymmetry between Mean Cycle/Cut and Min/Max Cycle/Cut has allowed us to reduce the time per iteration for cut canceling from $O(nMF)$ to $O(MF)$, whereas the cycle canceling per-iteration times are essentially the same. In the same way that the Cancel and Tighten algorithms have essentially the same iteration bounds as the Mean Canceling algorithms, but much better times per iteration, our Blocking Cancel algorithms have the same iteration bounds as our Relaxed Min/Max algorithms, but much better times per iteration.

However, in all cases cycle canceling looks like a better choice than cut canceling. The fastest weakly polynomial algorithms in Table 1 are the Scaling Network Simplex algorithms, the Cancel and Tighten algorithms, and the Blocking Cancel Min/Max algorithms, which have exactly the same time bounds in all cases. The difference between the primal and dual time bounds for these three classes of algorithms is that the $O(m)$ cycle cancels which are done in a single operation in cycle canceling are amortized over the $O(m \log n)$ time for that operation, leading to $O(\log n)$ time per cycle, whereas the $O(n)$ cut cancels done in a single operation in cut canceling are amortized over the $O(SP)$ time for that operation, leading to the $O(SP/n)$ time per cut. Since $O(SP/n)$ is $O(m/n)$ up to log factors, which is usually much larger than $O(\log n)$, cycle canceling is usually faster than cut canceling.

This paper is organized as follows. Section 2 gives our formal notation for min-cost flow, and introduces basic notions of cycle and cut canceling. Then Section 3 gives basic algorithms for computing most negative cycles and most positive cuts. The primal and dual canceling algorithms are explained in Sections 4 and 5, respectively. These sections are written so that Result 4.k is dual to Result 5.k where possible. Each has a subsection covering the appropriate
<table>
<thead>
<tr>
<th>Class of Objects</th>
<th>Cycle Canceling Algorithms</th>
<th>Cut Canceling Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># Cancellations</td>
<td>Time/Cancellation</td>
</tr>
<tr>
<td>Arbitrary</td>
<td>O(mCU)</td>
<td>O(√m log C) Goldberg (1995)</td>
</tr>
<tr>
<td></td>
<td>Exponential</td>
<td>O(nn) Hellman (1958)</td>
</tr>
<tr>
<td>Min/Max</td>
<td>O(mCU)</td>
<td>Exponential Zadeh (1973)</td>
</tr>
<tr>
<td></td>
<td>Exponential</td>
<td>O(√nm log C) Goldberg (1995)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>O(nm)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>O(m)</td>
</tr>
<tr>
<td></td>
<td>Cost Scaling Primal NS</td>
<td>O(mn log(nC)) O(log n)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>O(mn log(nC))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>O(m^2 n log n)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>O(m^2 n)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(no str. poly.)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tarjan (1997)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>Min Mean Cycle</td>
<td>O(mn log(nC)) O(√m log(nC))</td>
</tr>
<tr>
<td></td>
<td>Radzik and Goldberg (1994)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>O(mn log(nC))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>O(m^2 n log n)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>O(m^2 n)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(no str. poly.)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Queyranne (1986)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cancel and</td>
<td>(Primal) Cancel and Tighten</td>
<td>O(m)</td>
</tr>
<tr>
<td>Tighten</td>
<td>Goldberg and Turjan (1988)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>O(mn log(nC))</td>
<td></td>
</tr>
<tr>
<td></td>
<td>O(m^2 n log n)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>O(m^2 n)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(no str. poly.)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Queryranne (1986)</td>
<td></td>
</tr>
<tr>
<td>Most Help</td>
<td>Most Helpful Cycle</td>
<td>Barahona and Tardos (1989)</td>
</tr>
<tr>
<td>Helpful</td>
<td></td>
<td>O(mn log(nCU)) (no str. poly.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>O(mSP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(no str. poly. Queryranne (1986))</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ratio</td>
<td>Min Ratio Cycle</td>
<td>O(mn log(nC)) O(min(n^3 log^2 n, n^3 log n + mn log^2 n log log n))</td>
</tr>
<tr>
<td></td>
<td>Wallacher (1991)</td>
<td>O(mSP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(no str. poly. McCormick and Shiozaki (1999))</td>
</tr>
<tr>
<td></td>
<td>McCormick and Shioura (1999)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relaxed</td>
<td>Relaxed Most Negative Cycle</td>
<td>O(mn log(nC)) O(mn^2 log n)</td>
</tr>
<tr>
<td>Min/Max Blocking Approx.</td>
<td></td>
<td>O(AP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pure Scaling</td>
<td>Cost Scaling Most Negative Cycle</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Table showing the number of iterations and time per iteration for cycle and cut canceling algorithms. Where possible, we give known pseudo-polynomial bounds, weakly polynomial bounds, and strongly polynomial bounds in each box, in that order. The last row of boxes gives the new results in this paper.
notion of relaxed optimality, a subsection on the generic algorithm (and how to make it strongly polynomial), and a subsection on how to efficiently implement the algorithm. We end with Section 6, which compares and contrasts our new algorithms with previous algorithms, and which points out possible future improvements.

2 The Basics of Cycle and Cut Canceling

We first formally define the min-cost flow (MCF) problem. We are given a directed graph \( G = (N, A) \) with nodes \( N \) and arcs \( A \). We denote an arc from node \( i \) to node \( j \) by \( i \to j \). Each arc \( i \to j \) has lower and upper bounds \( l_{ij} \leq u_{ij} \) and a cost \( c_{ij} \). We assume without loss of generality and without loss of complexity that all supplies and demands are zero. We also assume throughout the paper, unless considering strongly polynomial bounds, that all data are integral.

The primal MCF problem also has flow \( x_{ij} \) on arc \( i \to j \), and the dual MCF problem has node potential \( \pi_i \) on node \( i \). The node potentials yield reduced costs on each arc defined by \( c_{\pi ij} = \pi_i + c_{ij} - \pi_j \). The primal MCF linear program is:

\[
(P) \quad \min \sum_{i \to j \in A} c_{ij} x_{ij} \\
\text{s.t.} \quad \sum_{k \to i \in A} x_{ki} - \sum_{j \to i \in A} x_{ij} = 0 \quad (i \in N) \\
\quad l_{ij} \leq x_{ij} \leq u_{ij} \quad (i \to j \in A).
\]

The dual LP to (P) has dual variables associated with the primal bounds, but it is easy to see from the dual constraints that these dual variables are just the positive and negative parts of the reduced costs, \((c^\pi)^+ = \max(0, c^\pi)\) and \((c^\pi)^- = \max(0, -c^\pi)\). Using this observation, the dual LP becomes:

\[
(D) \quad \max \sum_{i \to j \in A} (c_{\pi ij}^+) l_{ij} - \sum_{i \to j \in A} (c_{\pi ij}^-) u_{ij} \\
\text{s.t.} \quad \pi_i + c_{ij} - \pi_j = c_{\pi ij}^\pi \quad (i \to j \in A).
\]

2.1 Cycle Canceling

Primal complementary slackness conditions are that for all arcs \( i \to j \),

\[
\begin{align*}
   x_{ij} > l_{ij} & \implies c_{\pi ij}^+ \leq 0, \quad \text{and} \\
   x_{ij} < u_{ij} & \implies c_{\pi ij}^- \geq 0.
\end{align*}
\]

We now construct the residual network \( G(x) \) with costs \( \tilde{c} \) so that conditions (1) in the original network \( G \) are just shortest path optimality conditions in \( G(x) \). If \( i \to j \in A \) has \( x_{ij} < u_{ij} \), then put arc \( i \to j \) in \( G(x) \) as a forward arc with cost \( \tilde{c}_{ij} = c_{ij} \) and residual capacity \( r_{ij} = u_{ij} - x_{ij} \). If \( i \to j \in A \) has \( x_{ij} > l_{ij} \), then put arc \( j \to i \) in \( G(x) \) as a backward arc with cost \( \tilde{c}_{ji} = -c_{ij} \)
and residual capacity $r_{ji} = x_{ij} - l_{ij}$. Then a set of node potentials $\pi$ is a valid set of shortest path distances in $G(x)$ if and only if $\pi$ satisfies (1) w.r.t. $x$, i.e., if $\pi$ proves $x$’s optimality.

Recall that shortest path distances exist in $G(x)$ if and only if there are no negative directed cycles in $G(x)$, i.e., a directed cycle $\tilde{Q}$ such that $\sum_{i \rightarrow j \in \tilde{Q}} \tilde{c}_{ij} < 0$. Note that every directed cycle $\tilde{Q}$ in $G(x)$ corresponds to an undirected cycle $Q$ in $G$ that is augmentable in $G$ w.r.t. $x$ in such a way that

$$\text{cost}(Q, c) = \sum_{i \rightarrow j \text{ forw. in } Q} c_{ij} - \sum_{i \rightarrow j \text{ back. in } Q} c_{ij} = \sum_{i \rightarrow j \in \tilde{Q}} \tilde{c}_{ij} = \text{cost}(\tilde{Q}, \tilde{c}).$$

Thus a negative directed cycle $\tilde{Q}$ in $G(x)$ corresponds to a negative augmenting cycle in $G$ that proves that $x$ is not optimal. That is, choose $\alpha = \min_{i \rightarrow j \in \tilde{Q}} r_{ij}$ and increase $x$ by $\alpha$ on forward arcs of $Q$ and decrease $x$ by $\alpha$ on backward arcs of $Q$. This decreases the primal objective value by $\text{cost}(Q, c) \cdot \alpha$, and is called canceling the negative cycle $Q$. (If $\alpha = \infty$, then the primal objective value is unbounded.) To cancel node-disjoint cycles, we just cancel each individual cycle. We have effectively proved the theorem (Busacker and Saaty (1965)):

**Theorem 2.1** Flow $x$ is optimal if and only if there is no negative directed cycle in $G(x)$.

This theorem is the basis for Klein’s Algorithm: Find and cancel negative cycles until none remain, at which point $x$ must be optimal. All algorithms in the left-hand column of Table 1 are variants of Klein’s Algorithm that differ in the class of negative cycles chosen.

### 2.2 Cut Canceling

We now re-express the complementary slackness conditions (1) from a dual point of view:

$$c^\pi_{ij} < 0 \quad \Rightarrow \quad x_{ij} = u_{ij},$$
$$c^\pi_{ij} = 0 \quad \Rightarrow \quad l_{ij} \leq x_{ij} \leq u_{ij}, \quad \text{and}$$
$$c^\pi_{ij} > 0 \quad \Rightarrow \quad x_{ij} = l_{ij}. \quad (2)$$

We now define a network $G(\pi)$ whose modified bounds $l^\pi$ and $u^\pi$ have a feasible flow $x$ if and only if $x$ proves that $\pi$ is optimal through (2). If $c^\pi_{ij} < 0$ then $l^\pi_{ij} = u^\pi_{ij} = u_{ij}$; if $c^\pi_{ij} = 0$ then $l^\pi_{ij} = l_{ij}$ and $u^\pi_{ij} = u_{ij}$; and if $c^\pi_{ij} > 0$, then $l^\pi_{ij} = u^\pi_{ij} = l_{ij}$.

For a cut $S$, we denote by $\Delta^+ S$ the set of arcs with only their tail in $S$ (leaving $S$), and by $\Delta^- S$ the set of arcs with only their head in $S$ (entering $S$). We say that $i \rightarrow j$ crosses cut $S$ if $i \rightarrow j \in \Delta^+ S \cup \Delta^- S = \Delta S$. Define the value of cut $S$ w.r.t. $\pi$ by

$$V^\pi(S) = \sum_{\Delta^+ S} l^\pi_{ij} - \sum_{\Delta^- S} u^\pi_{ij}. \quad (3)$$

The next theorem (Hoffman’s (1960) Circulation Theorem) and the following paragraph motivate this definition of cut value.

**Theorem 2.2** Bounds $l^\pi$ and $u^\pi$ are infeasible if and only if there is a cut $S$ with $V^\pi(S) > 0$. 


We “augment” cut $S$ by increasing $\pi$ by $\beta > 0$ on nodes in $S$. This has the effect of increasing reduced costs by $\beta$ on $\Delta^+ S$, decreasing reduced costs by $\beta$ on $\Delta^- S$, and leaving reduced costs unchanged outside of $\Delta S$. We increase $\beta$ until a reduced cost hits zero, i.e., choose

$$\beta = \min(\min\{|c_{ij}^\pi| : i \to j \in \Delta^+ S, c_{ij}^\pi < 0\}, \min\{|c_{ij}^\pi| : i \to j \in \Delta^- S, c_{ij}^\pi > 0\}). \tag{4}$$

It can be shown (see, e.g., Ervolina and McCormick (1993a) Section 3.2) that with this choice of $\beta$, the dual objective value increases by $V^\pi \cdot \beta$. This is called canceling cut $S$. (If $\beta = \infty$, then the dual objective value is unbounded.) Then we get (Hassin (1983) Theorem 2):

**Corollary 2.3** Node potentials $\pi$ are optimal if and only if the value of every cut w.r.t. $\pi$ is non-positive.

This corollary is the basis for Hassin’s Algorithm: Find and cancel positive cuts until none remain, at which point $\pi$ must be optimal. All algorithms in the right-hand column of Table 1 are variants of Hassin’s Algorithm that differ in the class of positive cuts chosen.

## 3 Computing Most Negative Cycles and Most Positive Cuts

### 3.1 Most Negative Circuit and Most Positive Cocircuit are NP Hard

It is well-known that it is strongly NP Hard to compute a most negative circuit, by an easy reduction from Directed Hamiltonian Cycle (DHC). We recall from Garey and Johnson (1979) Problem GT38 that DHC is NP Hard even for planar graphs.

We now use this to show that it is strongly NP Hard to compute a most positive cocircuit. Let $\mathcal{N}$ be an instance of Planar DHC, and let $\mathcal{N}'$ be its planar dual. Put a lower bound of 1 and an upper bound of, e.g., $n^2$ on each arc of $\mathcal{N}'$. Then the only positive cuts in $\mathcal{N}'$ are directed cuts, and the value of the cut is the number of arcs crossing it.

Suppose that $S$ is a minimal-support cut, i.e., a cocircuit. Then it is easy to see that both $S$ and $\bar{S}$ must induce connected subgraphs of $\mathcal{N}'$, and that this implies that the subset of arcs of $\mathcal{N}$ which is planar dual to the corresponding cutset must be a simple directed cycle. Thus, if we could find a most positive cocircuit in $\mathcal{N}'$, its dual set of arcs would be a longest circuit in $\mathcal{N}$, and this is strongly NP Hard.

### 3.2 Computing Most Negative Node-Disjoint Cycles

Since it is NP Hard to compute a most negative circuit, we cannot expect an efficient algorithm to do so. We instead follow Barahona and Tardos (1989) and use an assignment subproblem (finding a perfect matching of minimum cost) to find a most negative node-disjoint family of cycles (MNDC). Given the residual network $G(x)$ with costs $\tilde{c}$, construct an assignment problem $APN(x, \tilde{c})$ as follows. The nodes of $APN(x, \tilde{c})$ are two copies of the nodes of $G(x)$, one on the left, one on the right. Denote the left copy of node $i$ by $i'$, the right copy by $i''$. Each arc $i \to j$ of $G(x)$ gives an arc $i' \to j''$ of $APN(x, \tilde{c})$ with $\tilde{c}_{i'j''} = \tilde{c}_{ij}$. Each node $i$ of $G(x)$ also gives an artificial arc $i' \to i''$ with $\tilde{c}_{i'i''} = 0$ in $APN(x, \tilde{c})$. 

11
The key fact is that any perfect matching \( M \) in \( \text{APN}(x, \bar{c}) \) corresponds one-to-one with a family \( Q \) of node-disjoint directed cycles in \( G(x) \), where \( i \to j \in Q \in \mathcal{Q} \) if and only if \( i' \to j'' \) is a non-artificial arc of \( M \). This correspondence preserves costs, so that
\[
\sum_{i' \to j'' \in M} \bar{c}_{i'j''} = \sum_{Q \in \mathcal{Q}} \text{cost}(Q, \bar{c}) \equiv \text{cost}(Q, \bar{c}).
\]

We take advantage of this correspondence, and will talk about \( Q \) as a solution of \( \text{APN}(x, \bar{c}) \) instead of \( M \).

Since the all-artificial matching in \( \text{APN}(x, \bar{c}) \) has cost 0, the minimum cost perfect matching always has non-positive cost, and has cost 0 if and only if there are no negative directed cycles in \( G(x) \). In particular the (NP Hard to compute) most negative circuit \( Q^* \) in \( G(x) \) corresponds to a perfect matching in \( \text{APN}(x, \bar{c}) \), so the cost of the cycles in an optimal solution \( Q^* \) of \( \text{APN}(x, \bar{c}) \) is at most \( \text{cost}(Q^*, \bar{c}) \). This shows that MNDCs are a reasonable proxy for most negative circuits.

This also shows that we can use \( \text{cost}(Q^*, \bar{c}) \) as a measure of how close to optimality \( x \) is, since driving \( \text{cost}(Q^*, \bar{c}) \) to zero means that \( x \) is optimal (by Theorem 2.1). Note that if we cancel a cycle \( Q \in Q^* \) to transform \( x \) to \( x' \), then at least one arc in \( G(x) \) is not in \( G(x') \) (namely, an arc determining how much flow we push on \( Q \), and some new residual arcs could appear in \( G(x') \) that weren’t in \( G(x) \)). Thus we think of \( G(x) \) and \( \text{APN}(x, \bar{c}) \) as dynamic networks whose sets of arcs change as we cancel cycles.

### 3.3 Computing Most Positive Cuts

We now turn to computing a cut maximizing \( V^\pi(S) \), a most positive cut (MPC). Because of the close connection with Hoffman’s Circulation Theorem, the natural tool for computing MPCs is an algorithm for finding a feasible flow w.r.t. bounds \( l^\pi \) and \( u^\pi \). This tool produces a cutset which is in general a family of cocircuits.

The tool we use is sometimes called Phase I Max Flow (McCormick and Ervolina (1994)), see also Ahuja, Magnanti, and Orlin (1993) Application 6.1. Find any initial flow \( l^\pi \leq x^0 \leq u^\pi \), which will in general not satisfy conservation. Compute residual capacities \( r \) w.r.t. \( x^0 \), and excesses w.r.t. \( x^0 \) giving the amount of violation of conservation at each node via
\[
e_i(x^0) = \sum_j x_{ji}^0 - \sum_j x_{ij}^0.
\]

Now construct a max flow network \( \text{MFN}(l^\pi, u^\pi) \) as follows. Start by adding all the arcs of \( G(x^0) \) as internal arcs with residual capacities \( r \). Then add a source \( s \) and sink \( t \). For each node \( i \) with \( e_i(x^0) > 0 \), put an arc \( s \to i \) with capacity \( r_{si} = e_i(x^0) \), and for each node \( j \) with \( e_j(x^0) < 0 \) put an arc \( j \to t \) with capacity \( r_{jt} = |e_j(x^0)| \).

Now compute a maximum \( s \to t \) flow \( \tilde{x} \) in \( \text{MFN}(l^\pi, u^\pi) \); denote the value of \( \tilde{x} \) by \( \text{val}(\tilde{x}) \). Recall that there is a min cut \( \tilde{S} \) in \( \text{MFN}(l^\pi, u^\pi) \) with \( s \in \tilde{S} \) and capacity \( \text{cap}(\tilde{S}) = \text{val}(\tilde{x}) \). Then for each arc \( i \to j \) of \( G(\pi) \) set \( x_{ij} = x_{ij}^0 + \tilde{x}_{ij} - \tilde{x}_{ji} \) (where \( \tilde{x}_{ij} = 0 \) if \( i \to j \) is not forward residual, and \( \tilde{x}_{ji} = 0 \) if \( j \to i \) is not backward residual).

Note that if \( e_i(x^0) > 0 \), then \( e_i(x) = r_{si} - \tilde{x}_{si} \), if \( e_j(x^0) < 0 \), then \( e_j(x) = \tilde{x}_{jt} - r_{jt} \), and if \( e_i(x^0) = 0 \), then \( e_i(x) = 0 \) also. We can measure the extent of infeasibility of a flow \( x \) by the
sum of its positive excesses, \( P(x) = \sum \{ e_i(x) \mid e_i(x) > 0 \} \). Thus the max flow has reduced \( P(x^0) \) down to \( P(x) \), and \( P(x) \) measures by how much \( \tilde{x} \) fails to saturate \( s \) (by how much \( x \) fails to be feasible). We can summarize by noting that

\[
P(x) = P(x^0) - \text{val}(\tilde{x}).
\]

(Note that \( \text{cap}(\{s\}) = P(x^0) \), so \( \text{val}(\tilde{x}) \) can be at most \( P(x^0) \).)

Thus if \( \tilde{x} \) saturates \( s \) (i.e., \( \text{val}(\tilde{x}) = P(x^0) \) and \( \{s\} \) is a min cut), then \( P(x) = 0 \), so \( x \) is feasible to \( l^\pi \) and \( u^\pi \), and \( x \) proves the optimality of \( \pi \). Thus we can use the saturation gap \( P(x) \) as a measure of \( \pi \)'s closeness to optimality, since if we cancel cuts that modify \( \pi \) so as to drive \( P(x) \) to zero, then \( \pi \) is then optimal.

If \( \text{val}(\tilde{x}) < P(x^0) \), then \( \{s\} \) is not a min cut. Then \( \tilde{T} = N - \tilde{S} \) is a valid cut in \( G(\pi) \), and it can be shown (McCormick and Ervolina (1994) Lemma 3.2) that

\[
P(x) = V^\pi(\tilde{T}).
\]

We also have (McCormick and Ervolina (1994) Corollary 3.4) that:

**Lemma 3.1** When \( \text{val}(\tilde{x}) < P(x^0) \), cut \( \tilde{T} \) is a most positive cut.

**Proof:** The correspondence between \( \tilde{S} \) and \( \tilde{T} = N - \tilde{S} \) is a bijection between non-trivial \( s \)–\( t \) cuts in \( \text{MFN}(l^\pi, u^\pi) \) and cuts in \( G(\pi) \). Putting (6) and (7) together yields \( V^\pi(\tilde{T}) = P(x^0) - \text{cap}(\tilde{S}) \). The term \( P(x^0) \) is a constant. Thus minimizing \( \text{cap}(\tilde{S}) \) is equivalent to maximizing \( V^\pi(\tilde{T}) \).

If \( S^* \) is a most positive cocircuit, then \( S^* \) is a candidate for \( \tilde{T} \), so that \( V^\pi(\tilde{T}) \geq V^\pi(S^*) \). Thus \( \tilde{T} \) is a good proxy for a most positive cocircuit.

We use this Phase I Max Flow tool to compute and cancel MPCs. Suppose that we cancel such a MPC \( \tilde{T} \) to transform \( \pi \) to \( \pi' \). At least one arc in \( G(\pi') \) will have either its \( l^\pi \) decrease, or its \( u^\pi \) increase (namely an arc whose \( c^\pi \) has newly dropped to zero in determining \( \beta \) in (4)). There may also be some arcs whose \( l^\pi \) increase or \( u^\pi \) decrease (if canceling \( \tilde{T} \) caused this arc’s \( c^\pi \) to move away from zero). Thus we think of \( G(\pi) \) and \( \text{MFN}(l^\pi, u^\pi) \) as dynamic networks whose bounds change as we cancel cuts.

### 4 Primal Most Negative Cycle Canceling Algorithms

#### 4.1 Primal Relaxed Optimality

A very fruitful idea in min-cost flow algorithms is relaxed optimality (or approximate optimality), where we relax the primal complementary slackness conditions (1) by a non-negative parameter \( \varepsilon \), see, e.g., Tardos (1985), Bertsekas (1986), or Goldberg and Tarjan (1989, 1990). Formally, we say that flow \( x \) is \( \varepsilon \)-optimal if there is a set of node potentials \( \pi \) such that for all arcs \( i \to j \),

\[
\begin{align*}
    x_{ij} > l_{ij} & \Rightarrow c^\pi_{ij} \leq \varepsilon, \\
    x_{ij} < u_{ij} & \Rightarrow c^\pi_{ij} \geq -\varepsilon
\end{align*}
\]

(8)
One equivalent definition is that \( x \) is \( \varepsilon \)-optimal if there are no negative augmenting cycles w.r.t. the set of costs \( \tilde{c} + \varepsilon \). A more succinct way to re-express this is to require that \( \tilde{c}_{ij} \geq -\varepsilon \) for all arcs \( i \to j \) in \( G(x) \).

The general algorithmic idea is to start with \( \varepsilon \) large and drive \( \varepsilon \) towards zero, since \( x \) is 0-optimal if and only if it is optimal. The next lemma (Ahuja, Magnanti, and Orlin (1993) Lemma 10.2) says that \( \varepsilon \) need not start out too big, and need not end up too small.

**Lemma 4.1** Any flow \( x \) is \( C \)-optimal. Moreover, when \( \varepsilon < 1/n \), any \( \varepsilon \)-optimal flow is optimal.

This concept is closely tied to min mean cycles. Define \( \varepsilon(x) \) to be the minimum value of \( \varepsilon \) over all \( \pi \) such that \( x \) is \( \varepsilon \)-optimal. Moreover, when \( \varepsilon < 1/n \), any \( \varepsilon \)-optimal flow is optimal.

**Lemma 4.2** When \( x \) is not optimal, \( \varepsilon(x) = -\mu(x) \).

### 4.2 Generic Relaxed MNDC Canceling

Our generic Relaxed MNDC Canceling algorithm starts with \( \varepsilon = C \) and executes scaling phases, where each phase cuts \( \varepsilon \) in half. The input to a phase is a \( 2\varepsilon \)-optimal flow from the previous phase, and its output is an \( \varepsilon \)-optimal flow. This continues until \( \varepsilon < 1/n \), at which point Lemma 4.1 says that we are finished, having done \( O(\log(nC)) \) phases. During an \( \varepsilon \) scaling phase, we use \( \tilde{c}_{ij} = \tilde{c}_{ij} + \varepsilon \) as the set of costs in the residual network \( G(x) \). Note that this definition implies that when both arcs \( i \to j \) and \( j \to i \) belong to \( G(x) \), we have

\[
\tilde{c}_{ij} + \tilde{c}_{ji} = 2\varepsilon.
\]

Within a phase we cancel most negative node-disjoint cycles w.r.t. \( \tilde{c} \) by solving the assignment problem \( \text{APN}(x, \tilde{c}) \). We call such a family of cycles an \( \varepsilon \)-MNDC. Note that we have two different notions of “closeness to optimality”: We drive \( \varepsilon \) towards zero (and so \( x \) towards optimality) at the outer level of phases, and the value of \( \text{APN}(x, \tilde{c}) \) towards zero (and so \( x \) towards \( \varepsilon \)-optimality) within a phase. Observe that \( \text{APN}(x, \tilde{c}) \) is the problem resulting from (an inner) relaxation of the condition that \( x \) is \( \varepsilon \)-optimal, in that node \( i \) is associated with two dual variables, \( \sigma_i' \) and \( \sigma_i'' \), always satisfying the relaxed \( \varepsilon \)-optimality condition \( -\sigma_i' + \tilde{c}_{ij} - \sigma_j'' \geq -\varepsilon \) for all arcs of \( G(x) \). As the phase progresses, \( \sigma_i' + \sigma_i'' \) converges to \( c_{ij''} = 0 \), and the sum equals zero once the optimal value of \( \text{APN}(x, \tilde{c}) \) is zero. At this point we can set \( \pi_i = \sigma_i' \) for all nodes \( i \), and \( \pi \) proves the \( \varepsilon \)-optimality of \( x \).

Note that \( \varepsilon \)-MNDCs are good approximations to min mean cycles: If \( Q \) is an \( \varepsilon \)-MNDC, the definition of \( \tilde{c} \) implies that \( \text{cost}(Q, \tilde{c}) \) equals \( \text{cost}(Q, \tilde{c}) + \varepsilon \sum_{Q \in Q} |Q| \), which implies that the mean cost of \( Q \) is smaller than \( -\varepsilon \). The fact that \( x \) is \( 2\varepsilon \)-optimal at the beginning of this scaling phase implies by Lemma 4.2 that the min mean cycle value \( \mu(x) \) is at least \( -2\varepsilon \). Thus an \( \varepsilon \)-MNDC is within a factor of two of a min mean cycle.
\[ \varepsilon \text{-MNDC Canceling Algorithm:} \]

Initialize \( \varepsilon := C. \)
Find a feasible flow \( x. \)

While \( \varepsilon \geq 1/n \) do:
  - Put \( \varepsilon := \varepsilon / 2. \)
  - Put \( \tilde{c}_{ij} := \tilde{c}_{ij} + \varepsilon \) for all \( i \rightarrow j \) in \( G(x). \)
  - While there is a negative cycle w.r.t. \( \tilde{c}_e \) do:
    - Solve APN\((x, \tilde{c}_e)\) to find an \( \varepsilon \text{-MNDC} \) \( Q^*. \)
    - Cancel \( Q^*. \)

End.

End.

The next lemma shows that canceling an \( \varepsilon \text{-MNDC} \) does not decrease \( \text{cost}(Q^*, \tilde{c}) \), i.e., that our measure of closeness to optimality in a phase is non-decreasing.

**Lemma 4.3** Suppose that \( Q^* \) is optimal for APN\((x, \tilde{c}_e)\), that \( x' \) is the result of canceling \( Q^* \), and that \( Q' \) is optimal for APN\((x', \tilde{c}_e)\). Then \( \text{cost}(Q^*, \tilde{c}_e) \leq \text{cost}(Q', \tilde{c}_e) \).

**Proof:** Let \( \sigma \) be optimal dual variables (node potentials) for APN\((x, \tilde{c}_e)\), so that the reduced cost of each \( i' \rightarrow j'' \) in APN\((x, \tilde{c}_e)\), namely \( \tilde{c}_{ji} - \sigma_{i'} - \sigma_{j''} \), is non-negative. It suffices to show that \( \sigma \) is still dual feasible for APN\((x', \tilde{c}_e)\). The only possible violation of dual feasibility in APN\((x', \tilde{c}_e)\) is when a new residual arc \( v \rightarrow w \) appears due to canceling \( Q^* \). Since \( v \rightarrow w \) is new, \( Q^* \) must have contained \( w \rightarrow v. \)

Since \( w \rightarrow v \in Q^* \), \( \tilde{c}_{w'w''} - \sigma_{w'} - \sigma_{w''} = 0 \) by complementary slackness. Using (9) we get
\[
\tilde{c}_{v'w''} + \sigma_{w'} + \sigma_{w''} = 2\varepsilon. \tag{10}
\]

Dual feasibility of \( \sigma \) also implies that \( \sigma_{i'\prime} + \sigma_{j''} \leq \tilde{c}_{i'j''} = 0 \) and the same for \( w. \) Subtracting these from (10) yields \( \tilde{c}_{v'w''} - \sigma_{i'} - \sigma_{w''} \geq 2\varepsilon. \)

Recall that each cancellation makes at least one arc of \( Q \) hit either \( l_{ij} \) or \( u_{ij} \), and thus to drop out of \( G(x); \) call such an arc a tight arc. The intuition for the next lemma is that once a cancellation forces an arc to hit its bound, its reduced cost must change from one extreme of relaxed optimality to the other for it to be used in the other direction, a distance of \( 2\varepsilon \) by (9), see Figure 1. This then causes an increase of at least \( 2\varepsilon \) in the value of an \( \varepsilon \text{-MNDC}. \)

**Lemma 4.4** Suppose \( v \rightarrow w \) is a tight arc of an \( \varepsilon \text{-MNDC} \) \( Q \) and that \( R \) is a later \( \varepsilon \text{-MNDC} \) that uses \( w \rightarrow v. \) Then \( \text{cost}(R, \tilde{c}_e) \geq \text{cost}(Q, \tilde{c}_e) + 2\varepsilon. \)

**Proof:** Fix \( \sigma \) as dual variables proving the optimality of \( Q. \) If \( \sigma \) is infeasible for \( R, \) then there is an \( \varepsilon \text{-MNDC} \) \( Q' \) between \( Q \) and \( R, \) and an arc \( i \rightarrow j \) of \( Q', \) such that the new arc \( j \rightarrow i \) created by canceling \( Q' \) has \( \tilde{c}_{ij} - \sigma_{i'} - \sigma_{j''} < 0. \) Now \( \sigma_{i'} + \sigma_{j''} \leq 0, \sigma_{j'} + \sigma_{j''} \leq 0, \) and (9) imply that \( \tilde{c}_{i'j''} - \sigma_{i'} - \sigma_{j''} > 2\varepsilon. \)

If instead \( \sigma \) is feasible for \( R, \) then put \( Q' = R. \) The proof of Lemma 4.3 shows that \( \tilde{c}_{v'w''} - \sigma_{v'} - \sigma_{w''} \geq 2\varepsilon. \)
We write $i \in Q'$ to mean that node $i$ is the tail of an arc in $Q'$. In either case we get
\[ \text{cost}(Q', \tilde{c}^\varepsilon) - \sum_{i \in Q'} (\sigma_i + \sigma_i') \geq \sum_{i \rightarrow j \in Q'} (\tilde{c}_{ij} - \sigma_i - \sigma_j) \geq 2\varepsilon. \]
Since $\sigma_i + \sigma_i' \leq 0$ for all $i$ we get
\[ \text{cost}(Q', \tilde{c}^\varepsilon) - \sum_{\text{all } i} (\sigma_i + \sigma_i') \geq \text{cost}(Q', \tilde{c}^\varepsilon) - \sum_{i \in Q'} (\sigma_i + \sigma_i'). \]
Optimality of $\sigma$ and $Q$ implies that $\text{cost}(Q, \tilde{c}^\varepsilon) = \sum_{i} (\sigma_i + \sigma_i')$. Putting these together yields that $\text{cost}(R, \tilde{c}^\varepsilon) \geq 2\varepsilon + \text{cost}(Q, \tilde{c}^\varepsilon)$, as desired.

![Figure 1: Kilter diagram showing why Lemma 4.4 is true.](image)

This now puts us in a position to prove the key lemma in the convergence proof of $\varepsilon$-MNDC Canceling. It shows that we must be in a position to apply Lemma 4.4 reasonably often.

**Lemma 4.5** After at most $m - n + 2$ cancellations of node-disjoint cycles, the value of an $\varepsilon$-MNDC increases by at least $2\varepsilon$.

**Proof:** Suppose that the first cancellation is $\varepsilon$-MNDC $Q^1$ which transforms $x^0$ into $x^1$, the next cancels $Q^2$ to get $x^2$, ..., and the $(m - n + 2)$nd cancels $Q^{m-n+2}$ to get $x^{m-n+2}$.

We claim that there must be two $\varepsilon$-MNDCs $Q^h$ and $Q^k$ with $h < k$ such that $Q^h$ has $v \rightarrow w$ as a tight arc, and $Q^k$ uses arc $w \rightarrow v$. If not, then each tight arc effectively disappears from further consideration after the iteration where it becomes tight: It is no longer residual in the forward direction, and we are assuming that it is not used in the backward direction.

16
Arbitrarily designate one tight arc from each $Q^k$ as a *special* arc, and consider the subgraph of special arcs. As long as families of cycles avoid re-using tight arcs, the complement of this subgraph must always contain a spanning tree: If not, suppose that the special arc from canceling $Q^k$ was the first to disconnect the complement. Then $Q^k$ must contain another special arc (crossing the cut induced by the two connected components of the complement), a contradiction. Since the complement of a spanning tree has $m - n + 1$ arcs, there can be at most $m - n + 1$ iterations before the next iteration must re-use a tight arc.

So let $Q^h$ and $Q^k$ be two such node-disjoint families of cycles. Then Lemma 4.4 says that $\text{cost}(Q^h, \bar{c}^\varepsilon) + 2\varepsilon \leq \text{cost}(Q^k, \bar{c}^\varepsilon)$. But now, by Lemma 4.3, we have $\text{cost}(Q^1, \bar{c}^\varepsilon) + 2\varepsilon \leq \text{cost}(Q^h, \bar{c}^\varepsilon) \leq \text{cost}(Q^{m-n+2}, \bar{c}^\varepsilon)$.

**Lemma 4.6** The cost of every $\varepsilon$-MNDC in a scaling phase is at least $-n\varepsilon$.

**Proof:** Let $\sigma$ be dual variables for $\text{APN}(x, \bar{c}^{2\varepsilon})$ proving the $2\varepsilon$-optimality of the final $x$ of the $2\varepsilon$ scaling phase, so that $\sum_{\nu} \sigma_{\nu} = 0$. Define $\sigma'_{\nu} = \sigma_{\nu} - \varepsilon$ and $\sigma''_{\nu} = \sigma_{\nu}$. Then $\sigma'$ is dual feasible for $\text{APN}(x, \bar{c}^\varepsilon)$ with objective value $-n\varepsilon$. By Lemma 4.3, the cost of all $\varepsilon$-MNDCs in this phase are at least $-n\varepsilon$.

Putting Lemmas 4.5 and 4.6 together yields our first bound on the running time of $\varepsilon$-MNDC Canceling:

**Theorem 4.7** A scaling phase of $\varepsilon$-MNDC Canceling cancels at most $(m - n + 2)n/2 \varepsilon$-MNDCs. Thus the running time of $\varepsilon$-MNDC Canceling is $O(mn\log(nC)\text{AP})$.

**Proof:** Lemma 4.6 says that the $\varepsilon$-MNDC value of the first family of cycles in a phase is at least $-n\varepsilon$. It takes at most $(m - n + 2)$ iterations to reduce this by $2\varepsilon$, so there are at most $(m - n + 2)n/2 = O(mn)$ iterations per phase. The time per iteration is dominated by computing a $\varepsilon$-MNDC, which is $O(\text{AP})$. The number of phases is $O(\log(nC))$ by Lemma 4.1.

A Strongly Polynomial Bound

We can apply standard techniques to make $\varepsilon$-MNDC Canceling run in strongly polynomial time. We first quote the “proximity lemma” (Goldberg and Tarjan (1989), Radzik and Goldberg (1994)) we will need to fix flows.

**Lemma 4.8** Suppose that $\pi$ proves that $x$ is $\varepsilon$-optimal. If $c^u_{ij} > n\varepsilon$, then every optimal $x^*$ has $x^*_{ij} = l_{ij}$. If $c^u_{ij} < -n\varepsilon$, then every optimal $x^*$ has $x^*_{ij} = u_{ij}$.

We now group the scaling phases of $\varepsilon$-MNDC Canceling into *groups*, where each group consists of $b = \lceil \log_2 n + 1 \rceil$ consecutive scaling phases. We then replace the first $\varepsilon$-MNDC cancel in each group by canceling a min mean cycle $Q^\ast$.

Our choice of $b$ ensures that the value of $\varepsilon$ at the end of a group is smaller than $1/n$ of the absolute value of the minimum cycle mean $\mu$. Let $\pi'$ be the set of node potentials obtained at the end of the group. Then we have $\text{cost}(Q^\ast, \bar{c}^{\varepsilon'}) = \text{cost}(Q^\ast, \bar{c}) = \mu|Q^\ast| < -n\varepsilon|Q^\ast|$. From this
it follows that at least one arc $i \rightarrow j$ of $Q^*$ must satisfy one of the hypotheses of Lemma 4.8, so that we can fix $x_{ij}$ to one of its bounds. It can be shown (Goldberg and Tarjan (1989), Radzik and Goldberg (1994)) that the flow on $i \rightarrow j$ was changed by canceling $Q^*$, so that $x_{ij}$ is newly fixed to a bound. After $O(m)$ groups, all $x_{ij}$'s are fixed, so we must be optimal. This yields

**Theorem 4.9** Canceling $\varepsilon$-MNDCs by groups with a min mean cycle cancellation at the beginning of each group takes $O(m^2 n \log n \, \text{AP})$ time overall.

**Proof:** Each group takes $O(\log n)$ scaling phases, each of which costs $O(mn \, \text{AP})$ time, and we need $O(m)$ groups to reach optimality. The cost of the min mean cycle calls is $(O(m) \text{ calls}) \times (O(mn) \text{ time/call})$, or $O(m^2 n)$ time, which is not a bottleneck.

### 4.3 Efficient Implementation of $\varepsilon$-MNDC Canceling

Our improvement borrows an idea from Goldberg and Tarjan’s (1989) (Primal) Cancel and Tighten algorithm that allows us to cancel many $\varepsilon$-MNDCs at once. Suppose that we solve $\text{APN}(x, \tilde{c}, \varepsilon)$ and cancel the $\varepsilon$-MNDC associated with its optimal matching to get $x'$, and let $\sigma$ be its optimal dual variables. Note that there may still be cycles $Q$ in $G(x')$ whose arcs all have reduced cost zero in $\text{APN}(x', \tilde{c})$ w.r.t. $\sigma$, or zero reduced cost cycles. It would then be a waste of time to solve $\text{APN}(x', \tilde{c})$, when it would be much faster to just cancel its zero reduced cost cycles.

Ideally, we want a procedure that cancels zero reduced cost cycles in $G(x)$ until none remain. Such a procedure was developed by Sleator and Tarjan (1983) as an application of their data structure called dynamic trees. Their procedure takes as input any flow network with bounds, and cancels directed cycles in the network until none remain (the original application was turning a feasible flow into a basic feasible flow), all in $O(m \log n)$ time.

To adapt this to our situation, from the optimal solution of $\text{APN}(x, \tilde{c})$ we construct the subnetwork $G(x, \sigma)$ of $G(x)$ consisting of all arcs whose reduced costs w.r.t. $\sigma$ in $\text{APN}(x, \tilde{c})$ is zero, and hand it to Sleator and Tarjan’s algorithm. It will cancel all zero reduced cost cycles in $O(m \log n)$ time. An upper bound on the total number of cycles canceled is $O(m)$. Note that the input $G(x, \sigma)$ contains at least one cycle that comes from an optimal matching in $\text{APN}(x, \tilde{c})$ unless $x$ is $\varepsilon$-optimal.

**Lemma 4.10** Canceling all zero reduced cost cycles strictly increases the optimal value of $\text{APN}(x, \tilde{c})$.

**Proof:** The proof of Lemma 4.3 shows that any new residual arcs which appear during cancellation have positive reduced costs, so the subnetwork $G(x, \sigma)$ of zero reduced cost arcs has no directed cycles. Hence $\text{APN}(x, \tilde{c})$ has no perfect matching with reduced cost zero, which means the optimal value must be strictly larger than the objective value of $\sigma$.

Unfortunately, although using Sleator and Tarjan’s algorithm could cancel as many as $O(m)$ cycles and must cause $\text{cost}(Q, \tilde{c})$ to strictly increase, we have no guarantee that we have canceled
enough families of cycles to be able to apply Lemma 4.5, so we cannot ensure that $\text{cost}(Q, \tilde{c}^\varepsilon)$ increases by at least $2\varepsilon$.

We propose two ways to deal with this problem. One is rounding the costs to multiples of $\varepsilon$: Define $\hat{c}^\varepsilon = \varepsilon \lfloor \tilde{c}^\varepsilon / \varepsilon \rfloor$, and use $\hat{c}^\varepsilon$ in place of $\tilde{c}^\varepsilon$ in the algorithm. Since each $\hat{c}^\varepsilon$ is an integral multiple of $\varepsilon$, if we start out an $\varepsilon$ scaling phase with $\pi_i$'s that are integral multiples of $\varepsilon$, then all reduced costs in both $G(x)$ and $\text{APN}(x, \hat{c}^\varepsilon)$ will also be multiples of $\varepsilon$. Therefore, when the cost of an $\varepsilon$-MNDC strictly increases after applying Sleator and Tarjan's algorithm, it must increase by at least $\varepsilon$. The rounding of the costs means that the cost of the first $\varepsilon$-MNDC in the phase is now lower-bounded by $-2n\varepsilon$ instead of $-n\varepsilon$, but we still need to apply Sleator and Tarjan only $O(n)$ times before the phase ends.

The other way is canceling approximate $\varepsilon$-MNDCs. Instead of restricting the Sleator-Tarjan cycle canceling to only those arcs with zero reduced cost in $\text{APN}(x, \tilde{c}^\varepsilon)$, we now allow Sleator-Tarjan to use any arc with reduced cost strictly less than $\varepsilon$ in $\text{APN}(x, \tilde{c}^\varepsilon)$ (these cycles might no longer belong to $\varepsilon$-MNDCs, but are only approximations to them). This implies that every family of node-disjoint cycles after the cancellation must include at least one arc with reduced cost at least $\varepsilon$, implying that the optimal value of $\text{APN}(x, \hat{c}^\varepsilon)$ must increase by at least $\varepsilon$, so we still need only $O(n)$ Sleator-Tarjan calls. The lack of rounding makes it easier to adapt this method to get a strongly polynomial bound. However, Iwata, McCormick, and Shigeno (1999a) shows that it is still possible to get a strongly polynomial bound with rounding, and that rounding seems to adapt better to cycle canceling for submodular flow than approximate $\varepsilon$-MNDCs.

The bottleneck operation in a phase is now solving the assignment problems. However, recall that we can re-optimize an assignment problem with all non-negative reduced costs via Dijkstra-type shortest path operations. In the rounding case, Lemma 4.3 shows that reduced costs in $\text{APN}(x, \hat{c}^\varepsilon)$ after each Sleator-Tarjan operation are non-negative. If we use approximate $\varepsilon$-MNDCs we can show that reduced costs in $\text{APN}(x, \hat{c}^\varepsilon)$ after each Sleator-Tarjan operation are non-negative as in Lemma 4.3. In either case, each shortest path costs only $O(\text{SP})$ and increases the cost of the assignment by at least $\varepsilon$, so we need only $O(n)$ shortest paths in a phase.

**Theorem 4.11** When we implement $\varepsilon$-MNDC Canceling using the Sleator-Tarjan algorithm to cancel cycles together with shortest path updates, the running time is $O(mn \log n \log(nC))$. If we use approximate $\varepsilon$-MNDCs and the strongly polynomial time technique, the algorithm is $O(m^2n \log^2 n)$.

**Proof:** We again have $O(\log(nC))$ scaling phases. Each phase now involves solving only one initial assignment problem, $O(n)$ calls to Sleator-Tarjan, and $O(n)$ shortest path updates. The bottleneck operation is the calls to Sleator-Tarjan, which cost $O(m \log n)$ each. In the strongly polynomial time version, each of the $O(m)$ groups computes one min mean cycle and runs $O(\log n)$ scaling phases. The scaling phases dominate the running time as before.  

It is natural to wonder if we can reduce the number of scaling phases to $O(\log C)$, as is possible for cut canceling, by using the the rounded costs $\varepsilon \lfloor \hat{c}/\varepsilon \rfloor$ instead of $\hat{c}^\varepsilon$. This would be called a pure cost scaling approach. However, seems not to work because Lemma 4.10 no longer
holds. Indeed, it is easy to construct examples where canceling zero reduced cost cycles causes a new zero reduced cost cycle to appear.

5 Dual Most Positive Cut Canceling Algorithms

5.1 Dual Relaxed Optimality

In the same way that we relaxed the primal complementary slackness of (1) to primal relaxed complementary slackness in (8), we now relax the dual complementary slackness of (2) by a parameter \( \delta \geq 0 \) (see, e.g., Ervolina and McCormick (1993a)). Formally, we say that node potentials \( \pi \) are \( \delta \)-optimal if there is a flow \( x \) such that for all arcs \( i \rightarrow j \),

\[
\begin{align*}
    c^\pi_{ij} < 0 & \Rightarrow u_{ij} - \delta \leq x_{ij} \leq u_{ij} + \delta, \\
    c^\pi_{ij} = 0 & \Rightarrow l_{ij} - \delta \leq x_{ij} \leq u_{ij} + \delta, \quad \text{and} \\
    c^\pi_{ij} > 0 & \Rightarrow l_{ij} - \delta \leq x_{ij} \leq l_{ij} + \delta
\end{align*}
\]

(compare with (2)). Equivalently, \( \pi \) is \( \delta \)-optimal if there are no positive cuts w.r.t. the bounds \( l^\pi - \delta \) and \( u^\pi + \delta \).

We start out with \( \delta \) large and drive \( \delta \) towards zero, since \( \pi \) is 0-optimal if and only if it is optimal. The next lemma (Ervolina and McCormick (1993a) Lemma 5.1) says that \( \delta \) need not start out too big, and need not end up too small.

Lemma 5.1 Any node potentials \( \pi \) are \( U \)-optimal. Moreover, when \( \delta < 1/m \), any \( \delta \)-optimal node potentials are optimal.

This concept is closely tied to max mean cuts. Define \( \delta(\pi) \) to be the minimum value of \( \delta \) over all \( x \) such that \( \pi \) is \( \delta \)-optimal. Also define \( \mu(\pi) \) to be the maximum of \( V^\pi(S)/|\Delta S| \) over all cuts \( S \). It turns out that \( \delta(\pi) \) and \( \mu(\pi) \) are essentially the objective values of dual linear programs (Hassin (1992) Theorem 4.1, Ervolina and McCormick (1993a) Theorem 5.3), yielding

Lemma 5.2 When \( \pi \) is not optimal, \( \delta(\pi) = \mu(\pi) \).

5.2 Generic Relaxed MPC Canceling

Our generic relaxed MPC Canceling algorithm starts with \( \delta = U \) and executes scaling phases, where each phase cuts \( \delta \) in half. The input to a phase is a \( 2\delta \)-optimal set of node potentials from the previous phase, and its output is a \( \delta \)-optimal set of node potentials. This continues until \( \delta < 1/m \), at which point Lemma 5.1 says that we are finished, having done \( O(\log(nU)) \) phases.

Based on (11), during a \( \delta \) scaling phase, we would like to use the bounds \( l^{\pi,\delta} = l^\pi - \delta \) and \( u^{\pi,\delta} = u^\pi + \delta \) for network \( G(\pi) \). However, the proof of Lemma 5.4 below will need the property that for the flow \( x \) corresponding to a max flow in \( \text{MFN}(l^{\pi,\delta}, u^{\pi,\delta}) \),

\[
\text{an arc } i \rightarrow j \text{ determining } \beta \text{ via (4) has } l_{ij} \leq x_{ij} \leq u_{ij}.
\]
Note that the optimality condition for $\tilde{x}$ implies that the cut $\tilde{T}$ in (7) satisfies
\begin{align}
  x_{ij} &= l_{ij}^{\pi,\delta} \quad \text{for all } i \to j \in \Delta^+ \tilde{T}, \text{ and} \\
  x_{ij} &= u_{ij}^{\pi,\delta} \quad \text{for all } i \to j \in \Delta^- \tilde{T}.
\end{align}
(13)

Unfortunately, when $\delta > u_{ij} - l_{ij}$, then (13) forces, e.g., $x_{ij} = l_{ij}^{\pi,\delta} - \delta = u_{ij} - \delta < l_{ij}$, violating (12).

There are two ways to fix this. One way is to re-define $l_{ij}^{\pi,\delta}$ and $u_{ij}^{\pi,\delta}$ as follows: When $c_{ij}^{\pi} < 0$, then $l_{ij}^{\pi,\delta} = \max(l_{ij}, u_{ij} - \delta)$ and $u_{ij}^{\pi,\delta} = u_{ij} + \delta$; when $c_{ij}^{\pi} = 0$, then $l_{ij}^{\pi,\delta} = l_{ij} - \delta$ and $u_{ij}^{\pi,\delta} = u_{ij} + \delta$; when $c_{ij}^{\pi} > 0$, then $l_{ij}^{\pi,\delta} = l_{ij} - \delta$ and $u_{ij}^{\pi,\delta} = \min(u_{ij}, l_{ij} + \delta)$ (see Figure 2 (a)). It is then easy to check that (12) can never be violated. However, this method does not even extend to the case of piece-wise linear convex costs with more than one piece per arc (Iwata, McCormick, and Shigeno (1999c)).

A more generalizable method is to keep the definitions $l_{ij}^{\pi,\delta} = l_{ij}^{\pi} - \delta$ and $u_{ij}^{\pi,\delta} = u_{ij}^{\pi} + \delta$, and to re-define our notion of cut canceling instead. Now, when $x_{ij}$ is such that $i \to j$ might determine
\( \beta \) in (4) while violating (12), then we no longer let such an arc participate in determining \( \beta \). That is, we replace (4) with

\[
\beta = \min \left( \min \{ |c_{ij}^\pi| \mid i \to j \in \Delta^+ S, \; c_{ij}^\pi < 0, \; \text{and} \; x_{ij} \geq l_{ij} \}, \right.
\min \{ |c_{ij}^\pi| \mid i \to j \in \Delta^- S, \; c_{ij}^\pi > 0, \; \text{and} \; x_{ij} \leq u_{ij} \} \right). \tag{14}
\]

This idea (apparently first used in Ervolina and McCormick (1993a)) allows the reduced cost of an arc whose \( u_{ij} - l_{ij} < \delta \) to pass through zero, see Figure 2 (b). This changes the value of the cut partway through the cancellation, but the cut does remain positive. This sort of cut cancellation is called total cut cancellation in Ervolina and McCormick (1993b). (If \( \beta = \infty \) in (14), then the dual problem is again unbounded.) We use this definition of \( l_{\pi,\delta}^\pi \) and \( u_{\pi,\delta}^\pi \) and assume that \( \beta \) is computed via (14) from now on. We denote the value of cut \( S \) w.r.t. \( l_{\pi,\delta}^\pi \) and \( u_{\pi,\delta}^\pi \) by \( V_{\pi,\delta}(S) \).

### \( \delta \)-MPC Canceling Algorithm:

Initialize \( \delta := U, \pi_i := 0 \).

While \( \delta \geq 1/m \) do:
- Put \( \delta := \delta/2 \).
  - While there is a positive cut w.r.t. \( l_{\pi,\delta}^\pi \) and \( u_{\pi,\delta}^\pi \) do:
    - Solve \( \text{MFN}(l_{\pi,\delta}^\pi, u_{\pi,\delta}^\pi) \) to find \( \delta \)-MPC \( \tilde{T} \).
    - Cancel \( \tilde{T} \) using (14).

End.

End.

Within a phase we cancel most positive cuts w.r.t. \( l_{\pi,\delta}^\pi \) and \( u_{\pi,\delta}^\pi \) by solving the max flow problem \( \text{MFN}(l_{\pi,\delta}^\pi, u_{\pi,\delta}^\pi) \) and getting a MPC from the min cut as suggested by Lemma 3.1. We call such a cut a \( \delta \)-MPC. Note that we have two different notions of “closeness to optimality”: We drive \( \delta \) towards zero (and so \( \pi \) towards optimality) at the outer level of phases, and the \( \delta \)-MPC value towards zero (and so \( \pi \) towards \( \delta \)-optimality) within a phase. Observe that \( \text{MFN}(l_{\pi,\delta}^\pi, u_{\pi,\delta}^\pi) \) is the problem resulting from (an inner) relaxation of the condition that \( \pi \) is \( \delta \)-optimal, in that we keep a flow satisfying the bounds \( l_{\pi,\delta}^\pi \) and \( u_{\pi,\delta}^\pi \), but violating conservation. As the phase progresses the violations of conservation are driven to zero, until the final flow proves \( \delta \)-optimality of the final \( \pi \).

Note that \( \delta \)-MPCs are good approximations to max mean cuts: If \( \tilde{T} \) is a \( \delta \)-MPC, the definitions of \( l_{\pi,\delta}^\pi \) and \( u_{\pi,\delta}^\pi \) imply that \( V_{\pi,\delta}(\tilde{T}) \) equals \( V_{\pi}(\tilde{T}) - \delta |\Delta \tilde{T}| \), which implies that the mean cost of \( \tilde{T} \) is at least \( \delta \). The fact that \( \pi \) is 2\( \delta \)-optimal at the beginning of this scaling phase implies by Lemma 5.2 that the max mean cut value \( \mu(\pi) \) is at most 2\( \delta \). Thus, since \( \delta \) is within a factor of two of \( \delta(\pi) \) during a scaling phase, a \( \delta \)-MPC is close to a max mean cut.

The next lemma shows that canceling a \( \delta \)-MPC does not increase \( V_{\pi,\delta}(\tilde{T}) \), i.e., that our measure of closeness to optimality in a phase is non-decreasing. It generalizes Hassin (1983) Theorem 4.

**Lemma 5.3** Suppose that \( \tilde{T} \) is a \( \delta \)-MPC for \( \pi \), that \( \pi' \) is the result of canceling \( \tilde{T} \), and that \( \tilde{T}' \) is a \( \delta \)-MPC for \( \pi' \). Then \( V_{\pi,\delta}(\tilde{T}) \geq V_{\pi',\delta}(\tilde{T}') \).
Proof: Let \( x \) be a flow in \( G(\pi) \) proving that \( \tilde{T} \) is a \( \delta \)-MPC. It suffices to show that \( x \) is a feasible flow for \( l_{\pi}^{\pi,\delta} \) and \( u_{\pi}^{\pi,\delta} \) (since \( V_{\pi}^{\pi,\delta}(\tilde{T}) \leq P(x) = V_{\pi}^{\pi,\delta}(\tilde{T}) \)).

To show that \( x \) is feasible for \( l_{\pi}^{\pi,\delta} \) and \( u_{\pi}^{\pi,\delta} \), we only have to worry about arcs \( i \to j \) where either \( l_{ij}^{\pi,\delta} > u_{ij}^{\pi,\delta} \), or where \( u_{ij}^{\pi,\delta} < u_{ij}^{\pi,\delta} \). In both cases this happens because \( c_{ij}^\pi = 0 \) and \( i \to j \) crosses cut \( \tilde{T} \), so that \( c_{ij}^\pi \neq 0 \). If \( i \to j \in \Delta^+\tilde{T} \), then (13) says that \( x_{ij} = l_{ij}^{\pi,\delta} \). Canceling \( \tilde{T} \) caused \( c_{ij}^\pi \) to be positive, implying that \( l_{ij}^{\pi,\delta} = x_{ij} \). Thus \( x_{ij} \) is still feasible in \( MFN(l_{\pi}^{\pi,\delta}, u_{\pi}^{\pi,\delta}) \). The case where \( i \to j \in \Delta^-\tilde{T} \) is similar.

Suppose that we cancel \( \delta \)-MPC \( \tilde{T} \) w.r.t. \( \pi \) to get \( \pi' \). Call an arc determining \( \beta \) in (14) a determining arc. Then the next lemma shows that the value w.r.t. \( \pi' \) of any other cut crossed by a determining arc of \( \tilde{T} \) must be smaller than \( V_{\pi}(\tilde{T}) \) by at least \( \delta \). The intuition is that once a cancellation forces the reduced cost on an arc to hit zero, the next time that this arc participates in a cut its flow must move from an interior value of \( l_{\pi}^{\pi,\delta} \) or \( u_{\pi}^{\pi,\delta} \) to an extreme value of \( l_{\pi}^{\pi,\delta} \) or \( u_{\pi}^{\pi,\delta} \), a distance of at least \( \delta \) by (12), see Figure 2 (c).

Lemma 5.4 Suppose that \( v \to w \) is a determining arc of \( \tilde{T} \), and that \( S \) is another cut with \( v \to w \in \Delta S \). Then \( V_{\pi'}(S) \leq V_{\pi}(\tilde{T}) - \delta \).

Proof: Let \( \tilde{x} \) be a max flow in \( MFN(l_{\pi}^{\pi,\delta}, u_{\pi}^{\pi,\delta}) \), and \( x \) be the corresponding flow in \( G(\pi) \). The proof of Lemma 5.3 showed that \( x \) is feasible to \( l_{\pi}^{\pi,\delta} \) and \( u_{\pi}^{\pi,\delta} \).

Case 1: \( v \to w \in \Delta^+\tilde{T} \cap \Delta^+S \) (the case where \( v \to w \in \Delta^-\tilde{T} \cap \Delta^-S \) is similar). Then \( v \to w \in \Delta^+\tilde{T} \), \( v \to w \) determining, (13), and (12) imply that \( l_{vw}^{\pi,\delta} = l_{vw} - \delta \leq x_{vw} - \delta \).

Case 2: \( v \to w \in \Delta^+\tilde{T} \cap \Delta^-S \) (the case where \( v \to w \in \Delta^-\tilde{T} \cap \Delta^+S \) is similar). Now, \( v \to w \in \Delta^+\tilde{T} \), \( v \to w \) determining, (13), and (12) imply that \( u_{vw}^{\pi,\delta} = u_{vw} + \delta \geq x_{vw} + \delta \).

Thus in either case, from the feasibility of \( x \) we obtain

\[
V_{\pi',\delta}(S) = \sum_{i \to j \in \Delta^+S} l_{ij}^{\pi,\delta} - \sum_{i \to j \in \Delta^-S} u_{ij}^{\pi,\delta} \\
\leq \left( \sum_{i \to j \in \Delta^+S} x_{ij} - \sum_{i \to j \in \Delta^-S} x_{ij} \right) - \delta \\
= -\sum_{i \in S} e_i(x) - \delta.
\]

Since \( N - \tilde{T} \) is a min cut of \( MFN(l_{\pi}^{\pi,\delta}, u_{\pi}^{\pi,\delta}) \), no nodes with positive excess w.r.t. \( x \) are contained in \( \tilde{T} \), and all nodes with negative excess w.r.t. \( x \) are contained in \( \tilde{T} \). Thus, from (7) we get \( V_{\pi,\delta}(\tilde{T}) = P(x) = -\sum_{i \in \tilde{T}} e_i(x) \geq -\sum_{i \in S} e_i(x) \geq V_{\pi',\delta}(S) + \delta \).

This now allows us to prove the key lemma in the convergence proof of \( \delta \)-MPC Canceling. It shows that we must be in a position to apply Lemma 5.4 reasonably often. It generalizes Hassin (1983) Corollary 2.

Lemma 5.5 After at most \( n \) cut cancellations, the value of a \( \delta \)-MPC decreases by at least \( \delta \).
The time per iteration is dominated by computing a \( \delta \) takes at most \( \arctan \) contributes at most \( \delta \), \( \delta \) violates the bounds by at most \( \ell \). We first quote the “proximity lemma” (Ervolina and McCormick (1993a) Lemma 5.10, Radzik A Strongly Polynomial Bound

Let \( k \) be the earliest iteration where \( T^k \) shares a determining arc with an earlier cut \( T^h \). By Lemma 5.4 applied to \( \tilde{T} = T^h \) and \( S = T^k \), we have \( V_{\pi}^{h-1,\delta}(T^h) \geq V_{\pi}^{h,\delta}(T^k) + \delta \). Note that the \( \delta \)-MPC value at iteration \( i \) is \( V_{\pi}^{i-1,\delta}(T^i) \). If \( V_{\pi}^{h,\delta}(T^k) \geq V_{\pi}^{h-1,\delta}(T^k) \), Lemma 5.3 says that \( V_{\pi}^{h,\delta}(T^1) \geq V_{\pi}^{h-1,\delta}(T^h) \geq V_{\pi}^{h,\delta}(T^k) + \delta \geq V_{\pi}^{h-1,\delta}(T^k) + \delta \geq V_{\pi}^{h-1,\delta}(T^n) + \delta \).

If instead \( V_{\pi}^{h,\delta}(T^k) < V_{\pi}^{h-1,\delta}(T^k) \), then let \( p \) be the latest iteration between \( h \) and \( k \) with \( V_{\pi}^{p-1,\delta}(T^k) < V_{\pi}^{p,\delta}(T^k) \). The only way for the value of \( T^k \) to increase like this is if there is an arc \( a \) with \( c_{\pi}^{p-1}(a) = 0 \) that crosses \( T^k \) in the reverse orientation to its orientation in \( T^p \). If \( a \in \Delta^+T^p \cap \Delta^-T^k \) (the other case is similar) and \( x^p \) is a flow proving optimality of \( T^p \), then (13) implies that \( u^{x^p}_a = x^p_a + 2\delta \geq x^0_a + \delta \). Thus Case 2 of the proof of Lemma 5.4 applies, showing that \( V_{\pi}^{p-1,\delta}(T^p) \geq V_{\pi}^{p,\delta}(T^k) + \delta \). Then Lemma 5.3 and the choice of \( p \) says that \( V_{\pi}^{p,\delta}(T^1) \geq V_{\pi}^{p-1,\delta}(T^p) \geq V_{\pi}^{p,\delta}(T^k) + \delta \geq V_{\pi}^{p-1,\delta}(T^k) + \delta \geq V_{\pi}^{p-1,\delta}(T^n) + \delta \).

**Lemma 5.6** The cost of every \( \delta \)-MPC in a scaling phase is at most \( m\delta \).

**Proof:** Let \( x \) prove the \( 2\delta \)-optimality of the initial \( \pi \) in a phase, so that \( x \) violates the bounds \( l^n \) and \( u^n \) by at most \( 2\delta \) on every arc. To get an initial flow \( x^0 \) to begin the next phase that violates the bounds by at most \( \delta \), we need to change \( x \) by at most \( \delta \) on each arc. Since each arc contributes at most \( \delta \) to the positive excess of all nodes w.r.t. \( x^0 \), the positive excess of \( x^0 \) is at most \( m\delta \). Recall that the positive excess of \( x^0 \) is an upper bound on the value of the first \( \delta \)-MPC in this phase. By Lemma 5.3, the value of all \( \delta \)-MPCs in this phase are at most \( m\delta \).

Putting Lemmas 5.5 and 5.6 together yields our first bound on the running time of \( \delta \)-MPC Canceling:

**Theorem 5.7** A scaling phase of \( \delta \)-MPC Canceling cancels at most \( mn \) \( \delta \)-MPCs. Thus the running time of \( \delta \)-MPC Canceling is \( O(mn \log(nU)) \).MF.

**Proof:** Lemma 5.6 shows that the \( \delta \)-MPC value of the first cut in a phase is at most \( m\delta \). It takes at most \( n \) iterations to reduce this by \( \delta \), so there are at most \( mn \) iterations per phase. The time per iteration is dominated by computing a \( \delta \)-MPC, which is \( O(MF) \). The number of phases is \( O(\log(nU)) \) by Lemma 5.1.
and Goldberg (1994) Lemma 10) we will need to fix the positive or negative part of a reduced cost to zero.

**Lemma 5.8** Suppose that \( x \) proves that \( \pi \) is \( \delta \)-optimal. If \( u_{ij} - x_{ij} > m\delta \), then every optimal \( \pi^* \) has \((c_{ij}^*)^- = 0\). If \( x_{ij} - l_{ij} > m\delta \), then every optimal \( \pi^* \) has \((c_{ij}^*)^+ = 0\).

We now group the scaling phases of \( \delta \)-MPC Canceling into groups, where each group consists of \( b = \lceil \log_2 m + 1 \rceil \) consecutive scaling phases. We then replace the first \( \delta \)-MPC cancel in each group by canceling a max mean cut \( T^* \).

Our choice of \( b \) ensures that the value of \( \delta \) at the end of a group is less than \( 1/m \) of the max mean cut value \( \mu \), which is the value of \( \delta \) at the beginning of the group. Let \( x' \) be the flow at the end of the group. There must be an arc \( i \to j \in \Delta T^* \) with \( |x'_i - l_{ij}| \geq \mu \) or \( |u_{ij} - x'_i| \geq \mu \), else the mean value of \( T^* \) would be less than \( \mu \). Since \( x'_i - u_{ij} \leq \delta < \mu \) and \( l_{ij} - x'_i \leq \delta < \mu \), we have an arc with \( x'_i - l_{ij} \geq \mu \) or \( u_{ij} - x'_i \geq \mu \). Thus there is at least one arc \( i \to j \) of \( T^* \) satisfying one of the hypotheses of Lemma 5.8, so that we can fix either the positive or negative part of \( c_{ij}^* \) to zero. It can be shown (Ervolina and McCormick (1993a), Radzik and Goldberg (1994)) that the positive or negative part of \( c_{ij}^* \) was changed by canceling \( T^* \), so this part of \( c_{ij}^* \) is newly fixed to zero. After \( O(m) \) groups, all \( c_{ij}^* \)'s are fixed, so we must be optimal. This yields

**Theorem 5.9** Canceling \( \delta \)-MPCs by groups with a max mean cut cancellation at the beginning of each group takes \( O(m^2n \log nMF) \) time overall.

**Proof:** Each group takes \( O(\log n) \) scaling phases, each of which costs \( O(mnMF) \) time, and we need \( O(m) \) groups to reach optimality. The cost of the max mean cuts is \((O(m) \text{ calls}) \times (O(nMF) \text{ time/call})\), or \( O(mnMF) \) time, which is not a bottleneck.

### 5.3 Efficient Implementation of \( \delta \)-MPC Canceling

Our improvement borrows an idea from Ervolina and McCormick’s (1993a) Dual Cancel and Tighten algorithm that allows us to cancel many \( \delta \)-MPCs at once. Suppose that we get optimal flow \( \tilde{x} \) in \( MFN(l^{\pi, \delta}, u^{\pi, \delta}) \) and cancel the \( \delta \)-MPC associated with its min cut to get \( \pi' \). Recall that \( \tilde{x} \) is feasible to \( MFN(l^{\pi', \delta}, u^{\pi', \delta}) \). It is possible that \( \tilde{x} \) is still a max flow in \( MFN(l^{\pi', \delta}, u^{\pi', \delta}) \), i.e., that there are other min cuts in the original network. Instead of re-solving \( MFN(l^{\pi', \delta}, u^{\pi', \delta}) \), it would be faster to just cancel all min cuts of \( MFN(l^{\pi, \delta}, u^{\pi, \delta}) \).

Here is a procedure that cancels all min cuts in \( MFN(l^{\pi, \delta}, u^{\pi, \delta}) \) until none remain. Let \( N_s \) and \( N_t \) be the sets of nodes reachable from \( s \) and to \( t \), respectively, in the residual graph of \( MFN(l^{\pi, \delta}, u^{\pi, \delta}) \) w.r.t. the max flow \( \tilde{x} \). Compute the strongly connected components of the induced subgraph of the residual graph by \( N \setminus (N_s \cup N_t) \), getting strongly connected components \( N_1, N_2, \ldots, N_k \). Contract each component \( N_s, N_1, N_2, \ldots, N_k, N_t \) to a single node, obtaining an acyclic network \( H \). The length of the arc of \( H \) from \( N_v \) to \( N_w \) is

\[
\min \left\{ \min\{c_{ij}^v \mid i \in N_v, j \in N_w, c_{ij}^v > 0, x_{ij} \leq u_{ij} \} \right\}, \\
\min\{ -c_{ij}^v \mid i \in N_w, j \in N_v, c_{ij}^v < 0, x_{ij} \geq l_{ij} \}, \infty \geq 0.
\]
Also add the reverse of each arc of \( H \), with length zero.

Now compute shortest path distance from \( N_t \) to every other node in \( H \) w.r.t. these non-negative lengths, and let \( \beta_v \) denote the shortest path distance to \( N_v \) in \( H \). Update the node potentials by \( \pi'_i := \pi_i + \beta_v \) when \( i \in N_v \). This update corresponds to simultaneously canceling \( O(n) \) min cuts (\( \delta \)-MPCs) w.r.t. \( x \) so that no min cuts remain. We call this procedure a Shortest Path Cut Cancellation. It costs \( O(SP) \) time.

**Lemma 5.10** A Shortest Path Cut Cancellation strictly decreases the \( \delta \)-MPC value.

**Proof:** After the cancellation, there will be an augmenting path in \( \text{MFN}(l^{\pi', \delta}, u^{\pi', \delta}) \) from \( s \) to \( t \) with positive capacity (corresponding to a shortest path from \( N_t \) to \( N_s \) in \( H \) with paths coming from the original network spanning across each strong component). Flow \( \tilde{x} \) can be augmented along this path to get new flow \( x' \) with a strictly smaller value of \( P(x) \).

Unfortunately, although doing a Shortest Path Cut Cancellation cancels \( O(n) \) \( \delta \)-MPCs and must cause \( P(x) \) to strictly decrease, we have no guarantee that we have canceled enough cuts to be able to apply Lemma 5.5, so that we cannot ensure that \( P(x) \) decreases by at least \( \delta \).

We propose two ways to deal with this problem. One is rounding the bounds to multiples of \( \delta \): Define \( \hat{l}^{\pi, \delta} = \delta \lceil l^{\pi, \delta} / \delta \rceil \) and \( \hat{u}^{\pi, \delta} = \delta \lceil u^{\pi, \delta} / \delta \rceil \), and use these in place of \( l^{\pi, \delta} \) and \( u^{\pi, \delta} \) in the algorithm. Since each \( \hat{l}^{\pi, \delta} \) and \( \hat{u}^{\pi, \delta} \) is an integer multiple of \( \delta \), as long as we start a \( \delta \) scaling phase with an \( x \) which is a multiple of \( \delta \), we will keep \( x \) as a multiple of \( \delta \) all the way through. Therefore, when \( P(x) \) strictly decreases after applying Shortest Path Cut Cancellation, it must decrease by at least \( \delta \). The rounding of the bounds means that the cost of the first \( \delta \)-MPC in the phase is now upper-bounded by \( 2m\delta \) instead of \( m\delta \), but we still need to apply Shortest Path Cut Cancellation only \( O(m) \) times before the phase ends.

The other way is canceling approximate \( \delta \)-MPCs. Instead of removing only arcs with zero residual capacity from \( \text{MFN}(l^{\pi, \delta}, u^{\pi, \delta}) \), we now remove all arcs with residual capacity strictly less than \( \delta \) (the resulting cuts might no longer be \( \delta \)-MPCs, but only approximations to them). This implies that there will be an augmenting path of capacity at least \( \delta \) after the Shortest Path Cancellation, so that \( P(x) \) decreases by at least \( \delta \). The lack of rounding makes it easier to adapt this method to get a strongly polynomial bound.

The bottleneck operation in a phase is now solving the max flow problems. However, recall that we can re-optimize a max flow problem by finding augmenting paths. Each augmenting path costs only \( O(m) \) time, and we need only \( O(m) \) augmenting paths in a phase.

**Theorem 5.11** When we implement \( \delta \)-MPC Canceling using the Shortest Path Cut Cancellation to cancel cuts together with augmenting path updates, the running time is \( O(mSP \log(nU)) \). If we use approximate \( \delta \)-MPCs and the strongly polynomial time technique, the algorithm is \( O(m^2SP \log n) \).

**Proof:** We again have \( O(\log(nU)) \) scaling phases. Each phase now involves solving only one initial max flow problem, \( O(m) \) calls to Shortest Path Cut Cancellation, and \( O(m) \) augmenting path updates. The bottleneck operation is the calls to Shortest Path Cut Cancellation, which cost \( O(SP) \) each. In the strongly polynomial time version, each of the \( O(m) \) groups involves
one Semi-Exact Tighten (Ervolina and McCormick’s (1993a)) and runs $O(\log n)$ scaling phases. The scaling phases dominate the running time as before.

In contrast to the primal case, we are now able to reduce the number of scaling phases $O(\log(nU))$ to $O(\log U)$. Replace $l^{\pi,\delta}$ and $u^{\pi,\delta}$ respectively by $\tilde{l}^{\pi,\delta} = \delta \lfloor l^{\pi}/\delta \rfloor$ and $\tilde{u}^{\pi,\delta} = \delta \lceil u^{\pi}/\delta \rceil$, which coincide with original modified bounds $l^{\pi}, u^{\pi}$ when $\delta < 1$. Then Lemma 5.10 continues to hold. By starting with $\delta = 2^{\lceil \log U \rceil}$ in place of $U$, all intermediate bounds and residual capacities are integral multiples of powers of 2. Hence $P(x)$ is not less than 1, which implies that we obtain an optimal solution when $\delta = 1$. We call this implementation the pure scaling version of $\delta$-MPC Cancellation.

**Theorem 5.12** The pure scaling version of $\delta$-MPC Cancellation runs in $O(m \log U)$ time.

### 6 Practical Potential and Future Directions

We have presented the primal $\epsilon$-MNDC Canceling algorithm, and the dual $\delta$-MPC Canceling algorithm. Both are based on scaling a relaxed optimality parameter. These algorithms could be viewed as explicit scaling counterparts to the Min Mean Cycle Canceling and (Primal) Cancel and Tighten algorithms of Goldberg and Tarjan (1989) on the primal side, and the Max Mean Cut and Dual Cancel and Tighten algorithms of Ervolina and McCormick (1993a) on the dual side.

Here are some tricks that do not improve the worst-case running time of the algorithms, but they should improve the performance of the algorithms in practice. The proof of Lemma 4.6 gives a good heuristic for transforming the final optimal solution to $\text{APN}(x, \tilde{c}^{2\epsilon})$ into an optimal solution to the first $\text{APN}(x, \tilde{c}^\epsilon)$: Change $\sigma$ to $\sigma'$ as suggested there, which yields a dual solution only $n\epsilon$ from optimality, and then use $n$ calls to the shortest path algorithm to re-optimize. In fact, we need to decrease $\sigma_i'$ by $\epsilon$ only if there is a negative reduced cost arc with tail $i'$ in $\text{APN}(x, \tilde{c}^{\epsilon})$, which could save many shortest path calls. If we start with $\epsilon = 2^{\lceil \log C \rceil}$ in place of $C$ and pre-multiply all costs by $(n + 1)$, then all intermediate costs are integral.

The proof of Lemma 5.6 shows how to transform the final max flow solution at the end of one phase into a good initial flow solution for the next phase at the cost of creating a positive excess of only $O(m\delta)$, which can be minimized using Dinic’s algorithm in only $O(mn)$ time. If we start with $\delta = 2^{\lceil \log U \rceil}$ in place of $U$ and pre-multiply all bounds by $(m + 1)$, then all intermediate bounds are integral.

We showed how to efficiently implement our algorithms by re-using some of the same technology that speeded up the Cancel and Tighten algorithms. We end up with the same weakly polynomial time bounds as the Scaling Network Simplex and Cancel and Tighten algorithms, which are the fastest known for any canceling algorithms, and which are within log factors of the best weakly polynomial time bounds of any min-cost flow algorithms. It is difficult to guess whether our algorithms would be better or worse in practice than the Scaling Network Simplex algorithms.

Our new algorithms potentially have better practical performance than the Cancel and Tighten algorithms due to two factors. First, our algorithms can be implemented to use all-
integer arithmetic when the data are integral, whereas the Cancel and Tighten algorithms must deal with fractions. Second, our algorithms produce decreases of $1/2$ in the scaled optimality parameter in each phase, whereas the Cancel and Tighten algorithms produce a decrease of only $(1 - 1/n)$ or $(1 - 1/m)$ (albeit with faster phases, producing the same overall running time), which can get lost in numerical imprecision. However, empirical evidence suggests that the Cancel and Tighten algorithms do not perform as well in practice as other algorithms (see, e.g., Goldberg (1997), McCormick and Liu (1993)), so our algorithms would have to perform substantially better than Cancel and Tighten algorithms to be practically useful.

The fastest known bound for cycle canceling algorithms is $O(nm \log n \log(nC))$, which is attained by our Blocking Cancel Approximate $\varepsilon$-MNDC algorithm, by Cost Scaling Primal Network Simplex (Orlin (1997)), and by Primal Cancel and Tighten (Goldberg and Tarjan (1989)). It is frustrating that we cannot reduce the $O(\log(nC))$ factor to $O(\log(C))$ in the same way that the $O(\log(nU))$ factor for $\delta$-MPC Canceling with Capacity Rounding was reduced to $O(\log(U))$ for Pure Capacity Scaling. This asymmetry in our framework comes from the fact that Lemma 4.10 does not hold for purely scaled costs, whereas Lemma 5.10 does hold for purely scaled capacities.

Our Blocking Approximate MNDC canceling algorithm using Sleator-Tarjan is similar to algorithms proposed by Goldberg (1995). The differences are that Goldberg’s algorithms do the Sleator-Tarjan cancellation directly on the original graph, and we do it w.r.t. a graph we derive from the assignment problem. Goldberg updates the dual variables using his \textsc{Cut-Relabel} operation, and we do it by updating the solution of our assignment problem. In the analysis of the number of Sleator-Tarjan cancellations, Goldberg shows that a \textsc{Cut-Relabel} operation removes at least one improvable node, whereas we show that the objective value of the assignment problem strictly increases. Goldberg’s algorithms are simpler to describe since they do not involve an assignment subproblem. It is again difficult to guess at the relative practical performance of our algorithm versus Goldberg’s algorithms without doing empirical testing.

The fastest known weakly polynomial time bound for cut canceling algorithms is $O(mSP \log U)$, which is attained by four algorithms: The pure scaling version of our $\delta$-MPC Canceling algorithm, the original Edmonds-Karp (1972) capacity scaling algorithm (as pointed out in Goldberg and Tarjan (1990)), the excess scaling algorithm of Orlin (1993), and the excess scaling dual network simplex algorithm of Armstrong and Jin (1997). It is natural to wonder if our algorithm is really new, or is just a variant of one of the three older algorithms.

Since Orlin’s and Armstrong and Jin’s algorithms are based on scaling excesses rather than capacities, they are clearly different. However, our algorithm does resemble the Edmonds-Karp algorithm in some ways, in that both scale the capacities in the same way, and both must remove $O(m)$ units of infeasibility in each scaling phase (measured at the current scale). The Edmonds-Karp algorithms removes this infeasibility arc-by-arc by using a shortest path computation between the ends of each infeasible arc to establish an augmenting path on which to remove this arc’s infeasibility. By contrast, our algorithm collects all the infeasibilities into node excesses, then uses augmenting paths to remove as much as possible. When no further augmenting paths exist, a shortest path computation is used to re-establish augmenting paths. Our approach seems more promising in practice since some infeasibility could be canceled just
by aggregating it at nodes, so we could end up having to do fewer shortest path and augmenting path computations than Edmonds-Karp.

The strongly polynomial bound for Armstrong and Jin’s Excess Scaling Dual Network Simplex algorithm is a factor of $O(m/n)$ faster than the bound for our Blocking Approximate Most Positive Cut Canceling algorithm. This leads to the open question of whether a more careful analysis of our algorithm (and/or (Primal) Cancel and Tighten) would lead to improving the strongly polynomial time bound to match the Armstrong and Jin bound.

Our Relaxed Min/Max Canceling framework also looks promising because it can be extended to more general problems. For example, in Iwata, McCormick, and Shigeno (1999c) we show that it can be extended to minimize separable convex objective functions over totally unimodular constraints as in Hochbaum and Shanthikumar (1990) and Karzanov and McCormick (1997). That paper also discusses the minor asymmetry between Lemmas 4.4 and 4.5 and Lemmas 5.4 and 5.5 arising from the lack of symmetry between primal and dual minimum cost flow from the viewpoint of separable convex costs. We also show (Iwata, McCormick, and Shigeno (1999a, 1999b)) that the Blocking Approximate MNDC canceling algorithm and the $\delta$-MPC canceling algorithm both extend to the submodular flow problem (see, e.g., Fujishige (1991)). The primal algorithm is faster than any previously known weakly polynomial algorithm for submodular flow (though Iwata, McCormick, and Shigeno (1999d) describes a different, slightly faster algorithm), and it matches the best known strongly polynomial bound. (On the suggestion of one referee of this paper, we were also able to show that one of Goldberg’s (1995) scaling shortest path min cost flow algorithms extends to submodular flow using our techniques Iwata, McCormick, and Shigeno (1998).) The dual algorithm gives the fastest known weakly polynomial dual algorithm for submodular flow, and the first known strongly polynomial dual algorithm for submodular flow. This contrasts with Mean Canceling algorithms, which seem to be difficult to extend to submodular flow (Cui and Fujishige (1988), McCormick, Ervolina, and Zhou (1994), Iwata, McCormick, and Shigeno (1999b)).

Acknowledgments

We are grateful to two anonymous referees for comments which significantly strengthened the content and presentation of this paper.

References


196–208.
minimum cost network flow. Discrete Applied Mathematics 46 133–165.
algorithms. Journal of ACM 34 596–615.
Vol. 47 (North-Holland, Amsterdam).
on Applied Mathematics 9 18–27.
on Computing 18 1013–1036.
11 43–58.
puting 24 494–504.
ACM 35 921–940.
Networks 20 79–91.
les cocycles de coûts moyens minimums. Technical Report, Groupe Intelligence Artificielle, Faculté
des Sciences de Luminy (Marseille, France).


31


