Les Cahiers du GERAD ISSN: 0711–2440

A strongly polynomial Contraction-Expansion algorithm for network flow problems

J.B. Gauthier, J. Desrosiers,
M.E. Lübbecke

G–2016–18

March 2016
Revised: February 2017

CITATION ORIGINALE / ORIGINAL CITATION

Gauthier, Jean-Bertrand, Desrosiers, Jacques, Lübbecke, Marco E., A strongly polynomial Contraction-Expansion algorithm for network flow problems, Computers & Operations Research, 84, 16-32, August 2017, http://dx.doi.org/10.1016/j.cor.2017.02.019.

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

A strongly polynomial Contraction-Expansion algorithm for network flow problems

Jean Bertrand Gauthier [a]
Jacques Desrosiers [a]
Marco E. Lübbecke [b]

[a] *GERAD & HEC Montréal, Montréal (Québec) Canada, H3T 2A7*

[b] *RWTH Aachen University, D-52072 Aachen, Germany*

jean-bertrand.gauthier@hec.ca
jacques.desrosies@hec.ca
marco.luebbecke@rwth-aachen.de

March 2016
Revised: February 2017

Les Cahiers du GERAD
G–2016–18

Abstract: This paper addresses the solution of the capacitated minimum cost flow problem on a network containing n nodes and m arcs. Satisfying necessary and sufficient optimality conditions can be done on the residual network although it can be quite time consuming as testified by the minimum mean cycle-canceling algorithm (MMCC). We introduce a contracted network which exploits these conditions on a much smaller network. Since the construction of this contracted network is very flexible, we study its properties depending on the construction choice. A generic contraction algorithm is then produced around the contracted network. Interestingly enough, it turns out it encapsulates both the MMCC and primal network simplex algorithms as extreme cases. By guiding the solution using a particular expansion scheme, we are able to recuperate theoretical results from MMCC. As such, we obtain a strongly polynomial Contraction-Expansion algorithm which runs in $O(m^3n^2)$ time. There is thus no improvement of the runtime complexity, yet the expansion scheme sticks to very practical observations of MMCC's behavior, namely that of phases and jumps on the optimality parameter. The solution time is ultimately significantly reduced, even more so as the size of the instance increases.

Keywords: Network flow problem, residual network, contracted network, minimum mean cost cycle, complexity analysis, strongly polynomial algorithm

Acknowledgments: Jacques Desrosiers acknowledges the Natural Sciences and Engineering Research Council of Canada for its financial support.

1 Introduction

The primal network simplex algorithm performs surprisingly well considering that only a small fraction of iterations induce non-degenerate pivots (see Ahuja et al. 1993, Figure 18.7). In fact, the network simplex and the cost scaling methods are typically preferred over competing alternatives for the solution of capacitated minimum cost flow problems (CMCF). Kovács (2015) suggests the use of the former for smaller networks and the latter for larger instances.

The minimum mean cycle-canceling algorithm (MMCC) is a prominent such alternative. Introduced by Goldberg and Tarjan (1989), this algorithm copes with degeneracy at the expense of a more involved pricing problem able to identify improving directions only, that is, with strictly positive step sizes. Despite its strongly polynomial time complexity, the theoretical behavior of MMCC is in practice no match for other methods. Radzik and Goldberg (1994) even improve the complexity some five years later and introduce the concept of phases. Two decades further down the road brings another improvement due to Gauthier et al. (2015) which combines phases with Cancel-and-Tighten (CT) presented along side MMCC in the seminal paper as a self-standing algorithm.

This paper presents a *Contraction-Expansion* algorithm (CE) inspired by MMCC and its complexity proof. Since the visual aid granted by the network flow formulation gives a lot of perspective to the theoretical analysis, we opt to present algorithmic choices in a constructive fashion. We adopt the definitions and nomenclature of Ahuja et al. (1993).

The paper is organized as follows. Section 2 defines the network problem and exposes the building block of this paper, namely the so-called contracted network. The contracted network is a flexible construction which gives rise to a generic contraction algorithm whose properties are discussed in Section 3. A behavioral study of the so-called *optimality parameter* follows in Section 4. From these observations, an expansion scheme guiding the solution process is drafted in Section 5, where the ensuing complexity analysis expands upon theoretical results from the minimum mean cycle-canceling algorithm. Our final thoughts can be found in Section 6.

We should also mention that the goal of this research is not to beat leading algorithms on network problems. Our intent is a better understanding of algorithms with strictly positive step size at every iteration. We believe that a more involved pricing problem is worth investigating in methods where rather complex mathematical programs are already solved to guide the solution process, evoking in particular, the oracle of a column generation algorithm.

2 Network problem

Assume a capacitated directed network $G := (N, A)$, where N denotes the set of n nodes and A the set of m arcs. The arc parametrization is captured by the cost vector $\mathbf{c} := [c_{ij}]_{(i,j)\in A}$ and non-negative bounds $[\ell_{ij}, u_{ij}]$, $(i, j) \in A$. A supply or demand, respectively, defined by a positive or negative value b_i, $i \in N$, is associated with each node such that $\sum_{i\in N} b_i = 0$. Supported by G and the vector of bounded flow variables $\mathbf{x} := [x_{ij}]_{(i,j)\in A}$, a formulation of CMCF is given by:

$$
\begin{aligned}
z^* := \quad & \min \sum_{(i,j)\in A} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b_i, \quad [\pi_i], \quad \forall i \in N, \\
& \ell_{ij} \le x_{ij} \le u_{ij}, \qquad\qquad \forall (i,j) \in A,
\end{aligned}
\tag{1}
$$

where $\boldsymbol{\pi} := [\pi_i]_{i\in N}$ is the vector of dual variables, also known as node potentials. We further assume that G does not contain any multiarc, i.e., two or more arcs sharing both the same head and tail nodes.

When the right-hand side $\mathbf{b} := [b_i]_{i\in N}$ is the vector of all zeros, we obtain a *circulation* problem. The latter is also the support of the minimum mean cycle-canceling algorithm. Indeed, its core component, namely

the so-called residual network is a circulation problem for which upper bounds are eventually neglected while searching for an improved solution.

We recall the path and cycle definitions used by Ahuja et al. (1993). A *path* from node i_1 to node i_r (abbreviated $i_1 \rightsquigarrow i_r$) in a directed graph $G = (N, A)$ is a sequence without repetition of nodes and arcs $i_1 - a_1 - i_2 - a_2 - \ldots - i_{r-1} - a_{r-1} - i_r$, satisfying the property that either $a_k = (i_k, i_{k+1}) \in A$ (forward arc) or $a_k = (i_{k+1}, i_k) \in A$ (backward arc) for all $k,\ 1 \leq k \leq r-1$. The sequence is typically given using nodes only. A *directed path* is an *oriented* version of a path consisting of forward arcs only. A *cycle* is a path together with either the forward arc $(i_r, i_1) \in A$ or the backward arc $(i_1, i_r) \in A$. A *directed cycle* is a directed path together with the arc $(i_r, i_1) \in A$ (which indeed imposes the orientation of the directed cycle). The *cost* of a path or a cycle is the sum of the cost on the forward arcs minus that on the backward arcs. Note that there is no backward arc in a directed path nor a directed cycle.

With respect to any dual variable vector $\boldsymbol{\pi} := (\pi_i)_{i \in N}$, the reduced cost of x_{ij}, $(i,j) \in A$, is defined as $\bar{c}_{ij} := c_{ij} - \pi_i + \pi_j$. The *reduced cost* of a path or a cycle is then computed analogously to its cost. Therefore, although Ahuja et al. (1993) state the following fundamental property for a directed cycle, the same telescopic summation argument proves to be enough regardless of its directed nature.

Proposition 1 *The cost and reduced cost of a cycle, directed or not, are equal.*

In the following we define the traditional residual network using which we can determine whether the current solution $\mathbf{x}^0 := [x^0_{ij}]_{(i,j) \in A}$ can be improved or not. An optimality certificate is then provided which indeed amounts to the statement of *cycle-canceling* algorithms. We finally move on to a *contraction* manipulation which induces a so-called contracted network.

2.1 Residual network

The residual network allows a marginal construction of the flow that may traverse the network aside the current flow $\mathbf{x}^0$ (increasing flow on certain arcs, possibly decreasing it on others). The combination of $\mathbf{x}^0$ along with an optimal residual flow computed on the residual network is optimal in the original network. As depicted in Figure 1, each arc $(i,j) \in A$ can be replaced by two *residual* arcs representing possible flow increments or decrements that depend on the remaining capacities:

- a forward arc (i,j) with cost $d_{ij} := c_{ij}$ and residual flow $0 \leq y_{ij} \leq r^0_{ij} := u_{ij} - x^0_{ij}$;
- a backward arc (j,i) with cost $d_{ji} := -c_{ij}$ and residual flow $0 \leq y_{ji} \leq r^0_{ji} := x^0_{ij} - \ell_{ij}$.

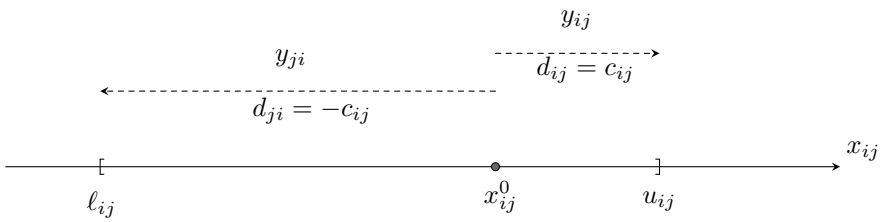


Figure 1: A change of variables

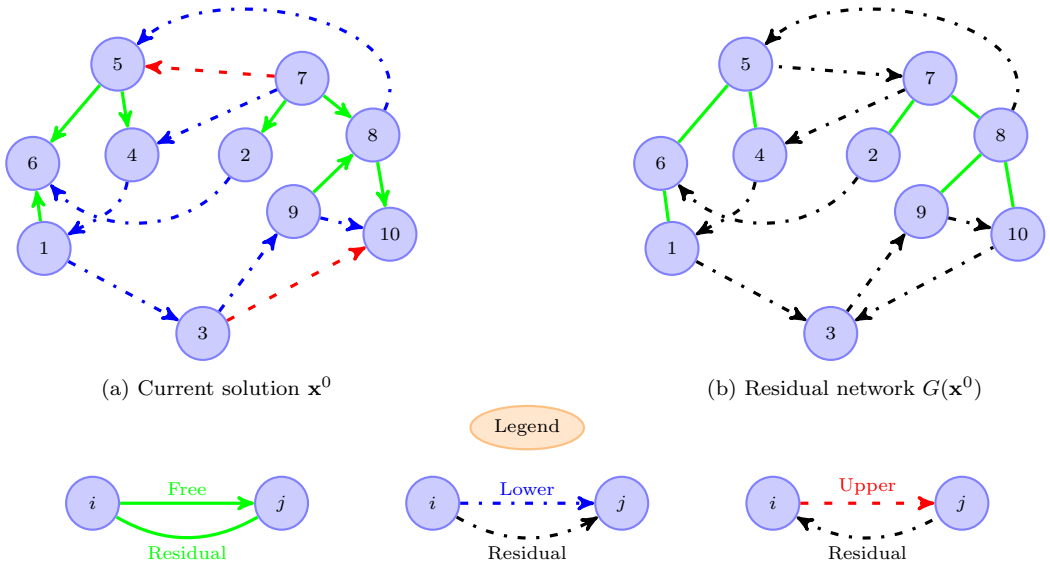
Denoted $G(\mathbf{x}^0) := (N, A(\mathbf{x}^0))$, the residual network with respect to $\mathbf{x}^0$ reflects the change of variables $y_{ij} - y_{ji} := x_{ij} - x^0_{ij}$, $(i,j) \in A$. It is based on the original nodes in N and the set of residual arcs $A(\mathbf{x}^0)$. Indeed, among the possible arc support $A' := \{(i,j) \cup (j,i) \mid (i,j) \in A\}$ only those arcs with strictly positive residual capacities are of interest, i.e., $A(\mathbf{x}^0) := \{(i,j) \in A' \mid r^0_{ij} > 0\}$.

An equivalent formulation of (1) using the new y-variables reads as

$$
\begin{aligned}
z^* = z^0 + \quad & \min \sum_{(i,j)\in A(\mathbf{x}^0)} d_{ij} y_{ij} \\
\text{s.t.} \quad & \sum_{j:(i,j)\in A(\mathbf{x}^0)} y_{ij} - \sum_{j:(j,i)\in A(\mathbf{x}^0)} y_{ji} = 0, \quad [\pi_i], \quad \forall i \in N, \\
& 0 \le y_{ij} \le r^0_{ij}, \qquad\qquad \forall (i,j) \in A(\mathbf{x}^0),
\end{aligned}
\tag{2}
$$

where $z^0 := \mathbf{c}^\intercal \mathbf{x}^0$ is the objective function value of the solution $\mathbf{x}^0$. Observe that using both forward and backward arcs between the same pair of nodes can be simplified to sending the net flow in a single direction only. This is why we assume $y_{ij}\, y_{ji} = 0$, $(i,j) \in A$.

Figure 2 exhibits the construction of the residual network $G(\mathbf{x}^0)$. Figure 2a contains arc flow variables x^0_{ij}, $(i,j) \in A$, at different values. Each of these can be attributed a status depending on its actual value: *lower* when $x^0_{ij} = \ell_{ij}$, *upper* when $x^0_{ij} = u_{ij}$, or *free* when $\ell_{ij} < x^0_{ij} < u_{ij}$. When a variable is free, the flow can be varied in either direction, implying the presence of two anti-parallel residual arcs. However, when a variable is lower (resp. upper), this induces only one arc oriented in the forward (resp. backward) direction. An arc at its lower or upper bound is also said to be *restricted*. Let the disjoint sets $L(\mathbf{x}^0)$, $U(\mathbf{x}^0)$, and $F(\mathbf{x}^0)$ be a partition that echoes the status of the original arcs at $\mathbf{x}^0$. In order to simplify the presentation, we denote this partition by $\{L^k, U^k, F^k\}$, where $k \ge 0$ refers to the solution $\mathbf{x}^k$ in iteration k.



(a) Current solution $\mathbf{x}^0$ (b) Residual network $G(\mathbf{x}^0)$

Figure 2: Residual network construction where undirected edges represent two anti-parallel arcs

2.2 Optimality conditions

It is known that a flow solution $\mathbf{x}^*$ is optimal if and only if the residual network $G(\mathbf{x}^*)$ contains no negative cost directed cycle, shortened to *negative cycle*, see Ahuja et al. (1993, Theorem 9.1). By iteratively sending as much flow as possible along negative cycles, i.e., *canceling* negative cycles, and updating the residual network accordingly, one obtains a generic cycle-canceling algorithm which terminates when there remains no negative cycle (Klein 1967).

We recall here three equivalent necessary and sufficient optimality conditions which can be interpreted in different perspectives with respect to network flows (see Ahuja et al. 1993, Theorems 9.1, 9.3, and 9.4). The first two refer to the residual network whereas the third is based on the original network. In regards to the

dual condition perspective, with respect to any dual variable vector $\boldsymbol{\pi}$, the reduced cost of x_{ij}, $(i,j) \in A$, is defined as $\bar{c}_{ij} := c_{ij} - \pi_i + \pi_j$. Let the reduced cost $\bar{d}_{ij} := d_{ij} - \pi_i + \pi_j$ of y_{ij}, $(i,j) \in A(\mathbf{x}^0)$, be computed in the same way.

Primal: $G(\mathbf{x}^0)$ contains no negative cycle.
Dual: $\exists \boldsymbol{\pi}$ such that $\bar{d}_{ij} \geq 0, \forall (i,j) \in A(\mathbf{x}^0)$.
Complementary slackness: $\exists \boldsymbol{\pi}$ such that, for every arc $(i,j) \in A$,

$$x^0_{ij} = \ell_{ij} \text{ if } \bar{c}_{ij} > 0; \qquad x^0_{ij} = u_{ij} \text{ if } \bar{c}_{ij} < 0; \qquad \bar{c}_{ij} = 0 \text{ if } \ell_{ij} < x^0_{ij} < u_{ij}. \tag{3}$$

Although the notion of cycle cancellation can be seen as an intuitive step size method, its foremost intention is the *elimination* of said cycle from the current residual network. Indeed, passing a strictly positive flow on any directed cycle W in $G(\mathbf{x}^0)$ obviously permits a transition between solutions, yet maintaining feasibility is ensured by limiting the flow to at most the smallest residual capacity of the arcs forming the cycle, say $\rho := \min_{(i,j)\in W} r^0_{ij}$. A directed cycle is *canceled* when the step size is equal to ρ such that at least one of the residual capacities r^0_{ij}, $(i,j) \in W$, is saturated. Note that *any* negative cycle qualifies for an improvement with respect to the objective function value since $\rho > 0$ by definition. As such, various *oracles* capable of coming up with negative cycles are conceivable, not all of these leading to strongly polynomial algorithms (see Ahuja et al. 1993, Section 9.6).

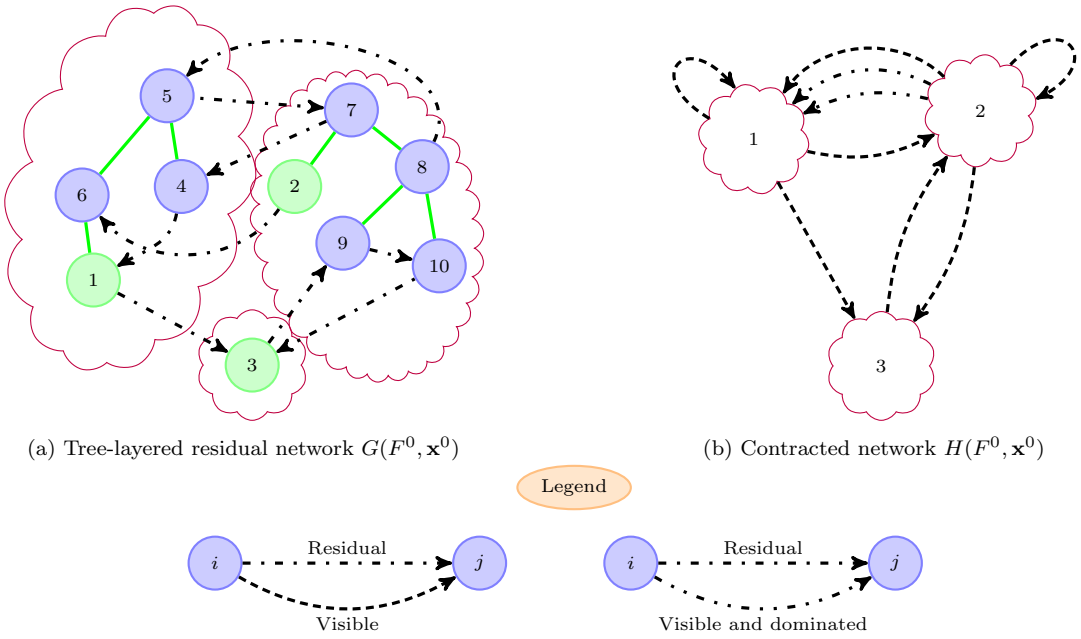
Since negative cycles are canceled sequentially, the oracle essentially defines a cycle-canceling algorithm. The minimum mean cycle-canceling is one such algorithm famous for its strongly polynomial runtime complexity, see Gauthier et al. (2015) for a recent survey. MMCC uses an oracle which dismisses residual capacities from the residual network and only identifies negative cycles of minimum mean cost. Technical aspects of this algorithm are presented in Section 4.2 where we build upon these results to show the runtime complexity of our Contraction-Expansion algorithm. The contracted network permits the exploration of alternative oracle constructions hoping to identify negative cycles faster. Whether these cycles are directed or not on the residual network remains to be seen.

2.3 Contracted network

A *cycle free* solution $\mathbf{x}^0$ does not contain any cycle of free arcs (Ahuja et al. 1993). In primal simplex terminology, such and only such solutions are basic. If one assumes a basic solution $\mathbf{x}^0$, the set of free arcs F^0 then defines a forest, that is, a collection of node-disjoint trees.

We arbitrarily identify each tree with one of its nodes which we call its *root*. For every node $i \in N$, let $\mathcal{R}(i)$ be its associated root (or tree identifier) such that any two nodes $i \neq j$ in N belonging to the same tree must have the same root node $\mathcal{R}(i) = \mathcal{R}(j)$. Moreover, if the node i is a root, then $\mathcal{R}(i) = i$. For the example in Figure 3a, root nodes 1, 2, and 3 have been chosen such that $\mathcal{R}(1) = \mathcal{R}(4) = \mathcal{R}(5) = \mathcal{R}(6) = 1$. Also, as a visual aid, we introduce the notion of a *tree-layer* to group the trees consisting of free arcs. Let $G(F^0, \mathbf{x}^0)$ denote the residual network on $\mathbf{x}^0$ where the tree-layer defined with respect to the set of free arcs F^0 is superposed. Note that $G(F^0, \mathbf{x}^0) \equiv G(\mathbf{x}^0)$. In fact, the arcs within the clouds in Figure 3a are bidirectional such that one must imagine the tree-layer with respect to the original arcs in A. Based on the tree-layer, we perform a *contraction* of every tree to its root node which effectively hides all free arcs and lets the remaining restricted arcs in $L^0 \cup U^0$ be visible. The tail and head for each of these visible arcs (i,j) is respectively redefined to $\mathcal{R}(i)$ and $\mathcal{R}(j)$. This contraction is likely to produce multiarcs. For instance, the arcs $(2,6)$, $(7,4)$, and $(8,5)$ have the same head and tail in the contracted network. Granted the actual cost computation has yet to be addressed, trivial cost dominance rules can be applied on such multiarcs. The end result appears in Figure 3b where the presence of the two dominated arcs between root nodes 1 and 2 concerns only efficiency matters.

Recall that a tree T is a connected graph that contains no cycle. In linear algebra terms, the coefficient columns in (1) associated with the arcs of T are linearly independent. By extension, we call a subset of arcs linearly independent if no cycle can be formed. Let us formalize the construction of the contracted network



(a) Tree-layered residual network $G(F^0, \mathbf{x}^0)$ (b) Contracted network $H(F^0, \mathbf{x}^0)$

Figure 3: Contracted network based on the set of free arcs, arcs within a cloud are contracted and become *hidden* while the others remain *visible*.

using a general tree-layer definition. While a great deal of attention is given to the set of free arcs F^0, the truth is that the tree-layer can be defined with respect to an arbitrary although *linearly independent* set of arcs, say $P^0 \subset A$, where the superscript is omitted unless iterate comparisons are required. The set P is then used to partition the arcs of $A(\mathbf{x}^0)$ in two categories. As such, one can think of the sets $H_P(\mathbf{x}^0)$ and $V_P(\mathbf{x}^0)$ as those arcs that are *hidden* and *visible*, respectively, in the contracted network. These sets are formed by

$$H_P(\mathbf{x}^0) := \bigcup_{(i,j)\in P} \begin{cases} (i,j),(j,i), & \text{if } (i,j) \in F^0 \\ (i,j), & \text{if } (i,j) \in L^0 \\ (j,i), & \text{if } (i,j) \in U^0 \end{cases} \tag{4}$$

$$V_P(\mathbf{x}^0) := A(\mathbf{x}^0) \setminus H_P(\mathbf{x}^0). \tag{5}$$

Definition 1 *With respect to the set P, the* tree-layer *identifies root nodes in the set*

$$N_P(\mathbf{x}^0) := \{i \in N \mid \mathcal{R}(i) = i\}, \tag{6}$$

as well as the arc partition $H_P(\mathbf{x}^0)$ and $V_P(\mathbf{x}^0)$ of the residual network $G(\mathbf{x}^0)$.

The contracted network $H(P, \mathbf{x}^0)$ is then obtained from the sets of root nodes and visible arcs:

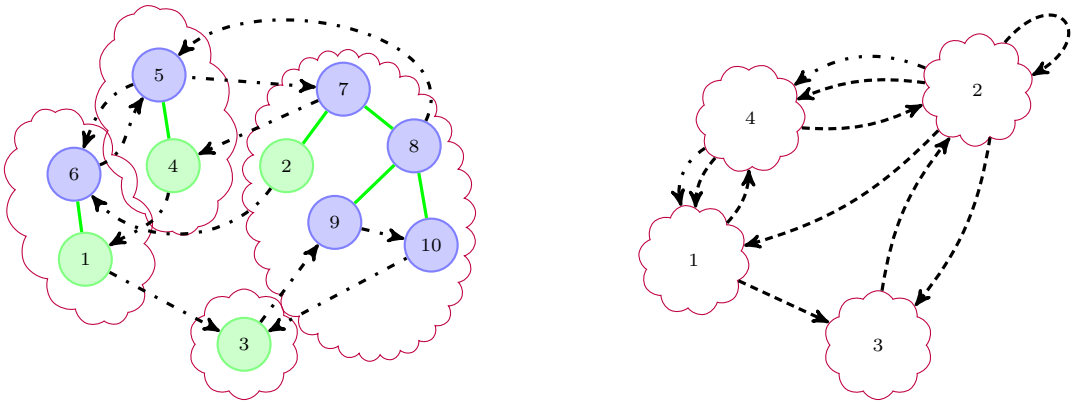
$$H(P, \mathbf{x}^0) := (N_P(\mathbf{x}^0), V_P(\mathbf{x}^0)). \tag{7}$$

where every visible arc (i, j) in the set $V_P(\mathbf{x}^0)$ is remapped to $(\mathcal{R}(i), \mathcal{R}(j))$ thus maintaining a bijection between arcs of the contracted network and the residual arcs.

We present two alternative contraction examples which collectively cover all possibilities. The first example uses a subset of free arcs $P \subset F^0$ while the second combines free arcs with some restricted arcs such that $P \supset F^0$. In the most general case, one can mix both possibilities by using some elements of F^0 and some of $A \setminus F^0$.

Figure 4 covers the first example where the subset of free arcs used is $P = F^0 \setminus \{(5, 6)\}$. In Figure 4a, the free arc $(5, 6)$ has been duplicated in both directions and consequently appears in the visible set $V_P(\mathbf{x}^0)$ with

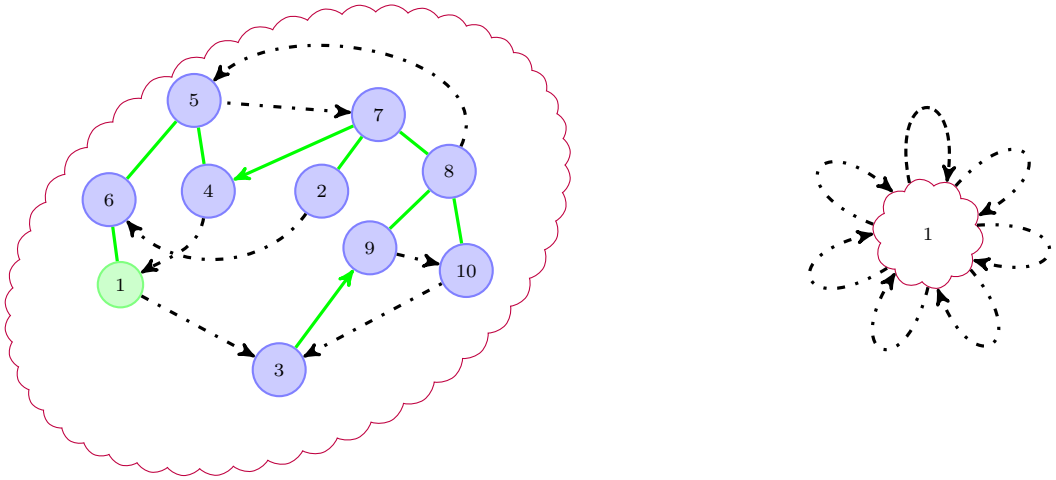
both arcs now being at a lower bound of zero in the residual network. We refer to this kind of manipulation on free arcs as *coerced degeneracy*. This yields a larger contracted network since there is now a forest with four trees to handle. Figure 4b portrays the consequent contracted network.



(a) Tree-layered residual network $G(P, \mathbf{x}^0)$ (b) Contracted network $H(P, \mathbf{x}^0)$

Figure 4: Contracted network with coerced degeneracy on the free arc $(5, 6)$

As additional free arcs are duplicated in both directions, one eventually reaches a point where the set $P \subseteq F^0$ is empty, such that no free arcs are hidden at all, hence yielding a contracted network whose arc set is the same as that of the residual network, i.e., $V_\emptyset(\mathbf{x}^0) = A(\mathbf{x}^0)$. The reader is now invited to consider the other extreme case where the tree-layer consists of a single spanning tree. Such is the content of Figure 5 where the set $P = F^0 \cup \{(7, 4), (3, 9)\}$ is still linearly independent and consists of the union of the free arcs along with two additional restricted arcs. Without loss of generality, assume these two arcs are basic degenerate in the primal simplex sense such that $P = B^0 \equiv B(\mathbf{x}^0)$ corresponds to a set of basic arcs at $\mathbf{x}^0$. The tree-layer $G(B^0, \mathbf{x}^0)$ seen in Figure 5a then splits the arcs in two subsets: The nine basic arcs of the spanning tree and the seven nonbasic arcs. The contracted network $H(B^0, \mathbf{x}^0)$ appears in Figure 5b, where the spanning tree is contracted to the sole root node hiding all basic arcs in the process and leaving visible the nonbasic ones. Each nonbasic arc becomes a loop, indeed, a directed cycle on $H(B^0, \mathbf{x}^0)$.



(a) Tree-layered residual network $G(B^0, \mathbf{x}^0)$ (b) Contracted network $H(B^0, \mathbf{x}^0)$

Figure 5: Contracted network with a set of basic arcs (primal network simplex algorithm)

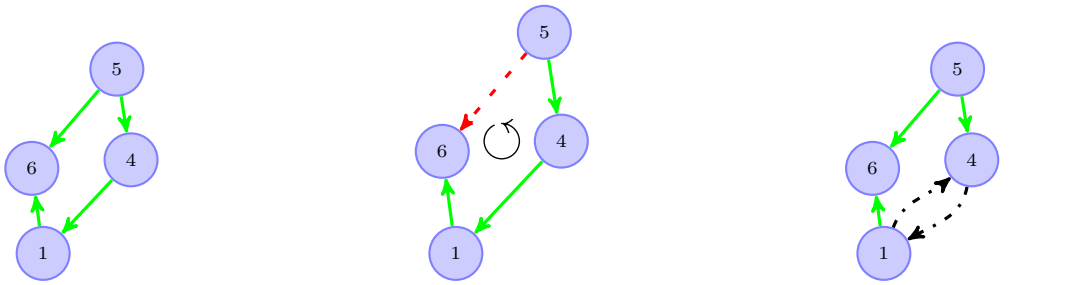
The arc cost computations of the contracted network are coming in the midst of the following section where features and properties of the contracted network are derived according to the choice of the set P, most notably the nature of the optimality conditions the oracle derived from the contracted network is able to fulfill. Since the discussion revolves around the content of the set P, it is worthwhile to underscore that only two cases are possible for an arc contained in the latter: It is either free or it is not.

3 Contracted network properties

Although the selection of the set P is fairly arbitrary, Section 3.1 addresses how easy it is to meet the linear independence requirement regardless of the current solution $\mathbf{x}^0$ being basic or not. Section 3.2 then states that any *contracted cycle*, a directed cycle identified on the contracted network, is *uniquely* extended on the residual network $G(\mathbf{x}^0)$. The analysis of the contracted network's arc cost is performed in Section 3.3 which is followed with the so-called pricing problem and the algorithm in Section 3.4. We then derive in Section 3.5 the optimality certificate whose nature depends on the selected set P. As the set P influences the content of the contracted network, different known algorithms are referenced in Section 3.6 by examining the possible outcomes of the pricing problem. Finally, Section 3.7 shows that once the set P is selected, the remaining arbitrary decisions that must be made have no impact on the algorithm.

3.1 Nonbasic solution

The set F^0 trivially fulfills the linear independence assumption when the current solution $\mathbf{x}^0$ is basic. We show that the basic status is not restrictive for the selection of the set P. Figure 6a presents a nonbasic solution where a cycle of free arcs is contained in a given feasible solution. Figures 6b and 6c handle the issue using two alternative mechanisms. With the first mechanism, the cycle of free arcs is canceled, in either direction, yielding at least one restricted arc within the cycle. Since the cancellation ultimately modifies the current solution, one may altogether prefer the improving direction, say saturating the arc $(5, 6)$ to its upper bound. With the second mechanism, the arc $(4, 1)$ is coerced degenerate in G. This duplicating manipulation does not change the current solution yet provides a fast way to eliminate the cycles of free arcs thus allowing one to define the set P using only independent arcs of F^0.



(a) Cycle of free arcs (b) Cycle canceled (c) Arc $(4, 1)$ coerced degenerate

Figure 6: Working with a nonbasic feasible solution on network G

Letting $f := |F^0| \leq m$, Proposition 2 asserts that, regardless of whether $\mathbf{x}^0$ is basic or not, it is easy to maintain a tree-layer induced by a linear independent set P.

Proposition 2 *A linear independent set $P \subset A(\mathbf{x}^0)$ can be derived from F^0 either by removing any cycle of free arcs using at most f cycle cancellations or by applying coerced degeneracy to at most f arcs.*

Proof. It is trivial to verify that rendering at most f arcs coerced degenerate means that there remains a suitable linearly independent subset of free variables capable of forming a tree-layer. In fact, one may think of these coerced degenerate variables as super-basic such that canceling a cycle of free arcs amounts to the *recovery of an optimal [or not] basis* which can be done in a straightforward manner according to Marsten et al. (1989). □

3.2 Uniqueness of the extended cycle

Observe that in the tree-layered residual network, any visible arc $(i, j) \in V_P(\mathbf{x})$ connects two root nodes using first a path from $\mathcal{R}(i)$ to i followed by arc (i, j) and second a path from j to $\mathcal{R}(j)$. Figure 7 portrays such an alternated path-arc-path sequence, from now on called a *rooted path*. When performing the contraction, it is convenient to let $\mathcal{P}(i)$, $i \in N$, denote the path from node i to root node $\mathcal{R}(i)$. The rooted path associated with the visible arc $(i, j) \in V_P(\mathbf{x})$ can therefore be decomposed as $\{-\mathcal{P}(i), (i, j), \mathcal{P}(j)\}$, where $-\mathcal{P}(i)$ is the reversed path $\mathcal{P}(i)$ and the orientation of a free arc is the one given by (i, j).

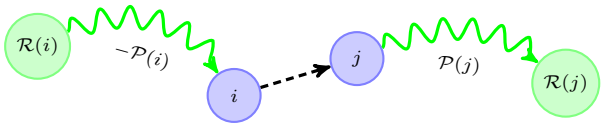


Figure 7: Rooted path $\mathcal{R}(i) \rightsquigarrow \mathcal{R}(j)$ associated with visible arc (i, j)

Several remarks about the rooted path are noteworthy. The two root nodes can possibly be the same, see the arc compatibility Definition 2. The two distinct paths are formed using only arcs in $H_P(\mathbf{x}^0)$, that is, hidden arcs in tree structures. Either or both of these paths could be of length zero such that the defining arc $(i, j) \in V_P(\mathbf{x}^0)$ could support the rooted path by itself. Finally, by definition of a tree, any visible arc $(i, j) \in V_P(\mathbf{x}^0)$ induces a *unique* rooted path.

Proposition 3 *Any contracted cycle W_H obtained on the network $H(P, \mathbf{x}^0)$ yields a unique extended cycle $W_{H:G}$ on the residual network $G(\mathbf{x}^0)$.*

Proof. Since only arcs that are part of the bijection are used to produce contracted cycles, the uniqueness of rooted paths guarantees that the extended cycle uniquely exists in the tree-layered residual network. Once again, the orientation of the free arcs in the extended cycle is given by that of the contracted cycle. □

Implementation. The remainder of this section is dedicated to describing a contracted cycle obtained on the contracted network $H(P, \mathbf{x}^0)$ in more details. Let

$$W_H := \{(\mathcal{R}(i_1), \mathcal{R}(j_1)), (\mathcal{R}(i_2), \mathcal{R}(j_2)), \ldots, (\mathcal{R}(i_{|W_H|}), \mathcal{R}(j_{|W_H|}))\}$$

be one such directed cycle on the contracted network. Observe that it produces a sequence of *entry* and *exit* nodes in the tree-layer which ultimately cycle through the same root nodes, i.e., $\mathcal{R}_s := \mathcal{R}(j_s) = \mathcal{R}(i_{s+1})$, $s \in \{1, \ldots, |W_H|\}$, where $i_{|W_H|+1}$ abusively equals to i_1. The extended cycle, seen on the residual network $G(\mathbf{x}^0)$, can then be expressed as a concatenation of rooted paths. This may not immediately lead to an elementary cycle.

One may take a look at Figure 8 should the following explanation require visual support. Think of the path contained in a given tree, say between the arcs (i_s, j_s) and (i_{s+1}, j_{s+1}), $s \in \{1, \ldots, |W_H|\}$. Notice that both paths $\mathcal{P}(j_s)$ and $-\mathcal{P}(i_{s+1})$ have at least one node in common, namely the root node $\mathcal{R}_s$. A cycle is formed during the concatenation if and only if there exists some other common node $u_s \neq \mathcal{R}_s$. Consider for instance that $\mathcal{P}(j_s)$ follows the path $j_s \rightsquigarrow u_s$ and then $u_s \rightsquigarrow \mathcal{R}_s$ while $-\mathcal{P}(i_{s+1})$ travels on $\mathcal{R}_s \rightsquigarrow u_s$ followed by $u_s \rightsquigarrow i_{s+1}$. Since all nodes connecting u_s to $\mathcal{R}_s$ and vice versa are common to both paths, an elementary

path is found by detecting the node $u_s \in \mathcal{P}(j_s)$ closest to j_s thus eliminating any back and forth play across opposing arcs. At last, the extended cycle $W_{H:G}$ extracted from the contracted cycle W_H is given by

$$W_{H:G} := \bigcup_{s=1}^{|W_H|} \{(i_s, j_s), \mathcal{P}(j_s) \setminus \mathcal{P}(u_s), -\mathcal{P}(i_{s+1}) \setminus -\mathcal{P}(u_s)\}, \tag{8}$$

where some of the composing paths $\mathcal{P}(j_s) \cup -\mathcal{P}(i_{s+1}),\ \forall s \in \{1, \ldots, |W_H|\}$, may be truncated to obtain an elementary cycle.

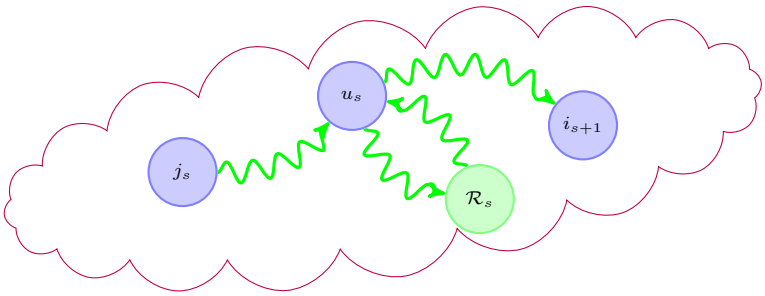


Figure 8: Elementary tree path detection between j_s and i_{s+1}

3.3 Arc cost transfer policy

So far, it has been established that any contracted cycle W_H exists uniquely as $W_{H:G}$ on the residual network. In order to extend optimality conditions to the contracted network, we require a somewhat opposite feature: If the residual network contains a negative cycle then so must the contracted network.

Since we assume that P contains only free arcs, optimality conditions imply that the reduced cost on all these arcs must be zero by the complementary slackness conditions (3). This is in line with the way dual variables are determined in the primal network simplex algorithm, that is, Ahuja et al. (1993, Chapter 11.4) arbitrarily set to zero the dual variable of the root node allowing a unique establishment for the remaining dual variables. It is a consequence of this procedure that π_i, $i \in N$, gives the cost of the path from i to the root node (with the cost from the root to itself of zero). This in turn produces a reduced cost of zero for every arc in the spanning tree. We reproduce the same scheme using the more general arbitrary constant initialization of the root node in Procedure 1.

Procedure 1 (Compute dual variable values) *The dual variable of a root node π_i, $i \in N_P(\mathbf{x}^0)$, is fixed to an arbitrary value. The remaining dual variable values π_i, $i \in N \setminus N_P(\mathbf{x}^0)$, are then the cost of the path $\mathcal{P}(i)$ plus $\pi_{\mathcal{R}(i)}$.*

By construction, the reduced cost of every arc in the forest induced by P is zero, i.e., $\bar{d}_{ij} = 0$, $(i,j) \in H_P(\mathbf{x}^0)$, indeed for all those arcs that are hidden in the contracted network.

Proposition 4 *Given $\boldsymbol{\pi}$ satisfying Procedure 1, the reduced cost of the contracted cycle W_H is equal to the cost of its extended counterpart $W_{H:G}$.*

Proof. Hiding the zero-cost arcs of $W_{H:G}$ makes the reduced cost of the contracted cycle W_H equal to the reduced cost of its extended version. By Proposition 1, as the reduced cost and the cost of a cycle (directed or not) are the same, the result follows. □

Remark Since the cost and reduced cost of a cycle are equal, the way dual variables are established in order to satisfy optimality conditions effectively transfers cost information of hidden arcs in $H_P(\mathbf{x}^0)$ to those that remain visible in the pricing problem $V_P(\mathbf{x}^0)$. Indeed, as the reduced cost of the former set is zero, the

reduced cost of the latter set must essentially capture the original cost information. Furthermore, in case of multiarcs in the contracted network, it is possible to keep only a single arc with the smallest reduced cost. Such an arc is *non-dominated* with respect to its reduced cost.

3.4 Contraction algorithm

We are now ready to present a contraction algorithm. We aim to detect a negative cycle using the contracted network $H(P, \mathbf{x}^0) = (N_P(\mathbf{x}^0), V_P(\mathbf{x}^0))$. Like in the pricing oracle of the minimum mean cycle-canceling algorithm, residual capacities are omitted and we bound the flow values by a *normalizing constraint* (Dantzig and Thapa 2003, Section 10.2). Proposition 5 states the kind of cycles that are identified using the following pricing problem:

$$\begin{aligned}
\mu_H \quad &:= \quad \min \sum_{(i,j)\in V_P(\mathbf{x}^0)} \bar{d}_{ij} y_{ij} \\
\text{s.t.} \quad \sum_{(i,j)\in V_P(\mathbf{x}^0)|\mathcal{R}(i)=\ell} y_{ij} - \sum_{(i,j)\in V_P(\mathbf{x}^0)|\mathcal{R}(j)=\ell} y_{ij} &= 0, \quad [\pi_\ell], \quad \forall \ell \in N_P(\mathbf{x}^0), \\
\sum_{(i,j)\in V_P(\mathbf{x}^0)} y_{ij} &= 1, \quad [\mu], \\
y_{ij} &\geq 0, \qquad \forall (i,j) \in V_P(\mathbf{x}^0),
\end{aligned} \tag{9}$$

where dual variables appear on the right between brackets. Flow conservation constraints are defined for each root node where the summation indices simply specify all arcs from $V_P(\mathbf{x}^0)$ that are related to the specified root nodes.

Proposition 5 *An extreme point solution to the pricing problem* (9) *corresponds to a minimum mean cost directed cycle on* $H(P, \mathbf{x}^0)$*, averaged over the number of arcs it contains.*

Proof. Define the strictly positive arc support set $W := \{(i,j) \in V_P(\mathbf{x}^0) \mid y^*_{ij} > 0\}$ of an optimal solution. By flow conservation, the arcs in W form either a cycle or a collection of cycles. Solutions of the latter category cannot be expressed as extreme points of (9). In the former case, the normalizing constraint forces the flow on each arc of the single cycle to be equal to $1/|W|$. Finally, cost coefficients of the pricing problem (9) are $\bar{\mathbf{d}}^\intercal = [d_{ij} - \pi_i + \pi_j]_{(i,j)\in V_P(\mathbf{x}^0)}$. Factoring out $1/|W|$ and using Proposition 1, we arrive at $\mu_H = \frac{1}{|W|} \sum_{(i,j)\in W} \bar{d}_{ij} = \frac{1}{|W|} \sum_{(i,j)\in W} d_{ij}$. □

By Proposition 4, if the residual network contains a negative cycle, we identify one by solving the pricing problem (9). By construction, the current solution $\mathbf{x}^k$, $k \geq 0$, is optimal when $\mu^k_H \geq 0$, that is, when the contracted network does not contain any negative cycle. Otherwise, given the identified contracted cycle W^k_H, one computes the non-negative step size

$$\rho^k := \min_{(i,j)\in W^k_{H:G}} r^k_{ij} \geq 0, \tag{10}$$

which is zero only if an arc of the extended cycle $W^k_{H:G}$ has a residual capacity of zero. We then obtain the solution $\mathbf{x}^{k+1}$ where the flow update, if any, is only performed on the arcs of $W^k_{H:G}$, that is,

$$\begin{aligned}
x^{k+1}_{ij} &:= \begin{cases} x^k_{ij} + \rho^k, & \forall (i,j) \in A \mid (i,j) \in W^k_{H:G} \\ x^k_{ij} - \rho^k, & \forall (i,j) \in A \mid (j,i) \in W^k_{H:G} \\ x^k_{ij}, & \text{otherwise} \end{cases} \\
z^{k+1} &:= z^k + \rho^k \, |W^k_H| \, \mu^k_H.
\end{aligned} \tag{11}$$

Observe that when $\rho^k = 0$, the set P^{k+1} must be different from P^k for otherwise the algorithm would not terminate. Degeneracy and cycling phenomena from the primal simplex algorithm come to mind. Indeed, unless rules for zero step size cycles are included, the same minimum mean cycle is identified in the next iteration. A pseudo-code is elaborated in Figure 9 where this process is repeated until optimality is reached.

Initialization: Iteration $k := 0$;
Feasible solution $\mathbf{x}^0$;
Define the set $P^k \subset A$;
Derive the contracted network $H(P^k, \mathbf{x}^k)$;
Solve the pricing problem (9) for μ_H^k, W_H^k;
If $\mu_H^k \geq 0$, **terminate** with an optimality certificate for $\mathbf{x}^k$;
Extract the extended cycle $W_{H:G}^k$;
Compute the step size ρ^k as in (10);
Update to solution $\mathbf{x}^{k+1}$ using the system (11);
Update the set $P^{k+1} \subset A$;
Iterate $k := k + 1$;

Figure 9: Generic contraction algorithm for network flow problems

The nature of the optimality certificate in the generic contraction algorithm goes hand in hand with the non-degeneracy guarantee provided (or not) by the selected set P and the induced contracted network. Let us analyze this property.

3.5 Nature of the optimality conditions

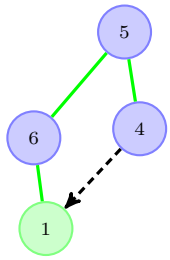
Given that any contracted cycle is uniquely extended, the underlying expectation is that it is possible to travel on the hidden arcs, i.e., the extended cycle is directed on $G(\mathbf{x}^0)$. Depending on the actual status of the arcs in set P, this expectation could be challenged. Verifying that any set P which consists of only free arcs always provides a contracted cycle W_H on which at least one flow unit can always be sent on the extended cycle $W_{H:G}$ is trivial: All arcs unaccounted for can be used in either direction. As supporting evidence, the first three extended cycle extractions illustrated in Figure 10 are directed in $G(\mathbf{x}^0)$. They come from contracted cycles obtained on a contracted network using only free arcs (Figure 3b or 4b). The fourth extended cycle extraction comes from Figure 5b and yields an undirected cycle inducing a zero step size.

Figure 10a is obtained from a contracted cycle defined by the arc loop on root node 1 in the contracted network of Figure 3b. Figure 10b comes from the same contracted network but the contracted cycle uses more arcs: $\{(1,3),(3,9),(8,5)\}$. Figure 10c identifies part of the previous cycle although it is based on the contracted network of Figure 4b where more visible arcs are available. Finally, Figure 10d is based on Figure 5b used with the primal network simplex algorithm. The arc $(7,4)$ is basic degenerate and one of the nonbasic arcs corresponds to the visible arc $(2,6)$, a loop on root node 1. The corresponding extended cycle is defined by $\{(\mathbf{2},\mathbf{6}),(6,5),(5,4),(4,7),(7,2)\}$, or in accordance with the threefold decomposition of $W_{H:G}$ given in (8) as

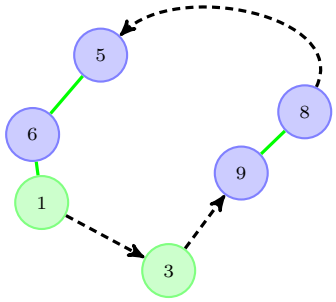
$$\{(\mathbf{2},\mathbf{6}),\ \{\{(6,1)\} \setminus \{(6,1)\}\},\ \{\{(1,6),(6,5),(5,4),(4,7),(7,2)\} \setminus \{(1,6)\}\}\}.$$

Unfortunately, the arc $(4,7)$ is not a residual arc and reduces the possible flow on the identified extended cycle to zero. In the primal simplex algorithm, pivoting on arc variable x_{26} induces a degenerate pivot.

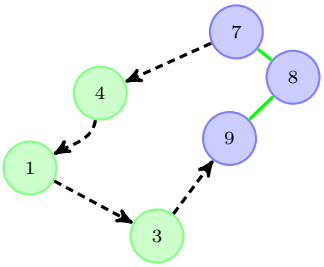
When $P \subseteq F^0$ and $\mu_H^0 < 0$, the contracted network $H(P, \mathbf{x}^0)$ is guaranteed to identify a directed extended cycle, that is, when the hidden arcs are free, any contracted negative cycle has a strictly positive step size on $G(\mathbf{x}^0)$. This refers to the primal optimality conditions of Section 2.2. On the other hand, when $P \nsubseteq F^0$, there is no such guarantee since the extended cycle could contain an arc with a residual capacity of zero such that the associated step size is non-negative. In that case, $\mu_H^0 \geq 0$ is only a sufficient optimality condition. This proves the following proposition.



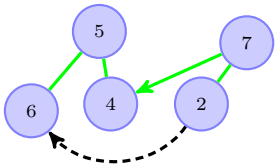
(a) Extended cycle from the contracted cycle $\{(4,1)\}$ on Figure 3



(b) Extended cycle from the contracted cycle $\{(1,3),\ (3,9),\ (8,5)\}$ on Figure 3



(c) Extended cycle from the contracted cycle $\{(1,3),\ (3,9),\ (7,4),\ (4,1)\}$ on Figure 4



(d) Extended cycle from the contracted cycle $\{(2,6)\}$ on Figure 5

Figure 10: Extended cycle extractions

Proposition 6 *Given the set $P \subset A$ of linearly independent arcs, if*

- *$P \subseteq F^0$, the oracle (9) provides necessary and sufficient optimality conditions for (1), i.e., $\mathbf{x}^0$ is optimal if and only if $\mu^0_H \geq 0$.*
- *$P \nsubseteq F^0$, the oracle (9) provides sufficient optimality conditions for (1), i.e., $\mathbf{x}^0$ is optimal if $\mu^0_H \geq 0$.*

Regardless of the choice of the set P, arcs $(i,j) \in A$ can be categorized in two classes: Those arcs that link nodes of the same tree, i.e., the head and tail of the residual arc have the same root node $\mathcal{R}(i) = \mathcal{R}(j)$, and those that link different trees, i.e., $\mathcal{R}(i) \neq \mathcal{R}(j)$.

Definition 2 *An arc $(i,j) \in A$ is* compatible *with P if and only if the head and tail refer to the same root node in the contracted network, i.e., $\mathcal{R}(i) = \mathcal{R}(j)$. Otherwise, it is* incompatible.

Compatible arcs are easy to identify as loops on the contracted network thus forming directed cycles on their own. Examples can be seen in Figure 3a as arcs $(4,1)$ and $(9,10)$ or in Figure 3b as the two loops on root nodes 1 and 2. This also implies that the extended cycle associated with a compatible arc $(i,j) \in V_P(\mathbf{x}^0)$ is not influenced by the content of other trees. The reduced cost of a loop is therefore the reduced cost of the cycle. For incompatible arcs, rooted paths have to be combined to form cycles in $H(P,\mathbf{x}^0)$. In this respect, we like to think of the reduced cost $\bar{d}_{ij}$, $(i,j) \in V_P(\mathbf{x}^0)$, as a *rooted cost*.

3.6 Extremal point solution space

It is straightforward to verify that the extreme cases, $P = \emptyset$ and $P = B^0$, respectively, correspond to the minimum mean cycle-canceling algorithm and the primal network simplex algorithm. In the former case, the contracted network $H(\emptyset,\mathbf{x}^0)$ has exactly the same structure as the residual network $G(\mathbf{x}^0)$, thus yielding a pricing problem equivalent to MMCC's. In the latter case, consider the contracted network of Figure 5 where the arcs of P are basic arcs (indeed forming the primal network simplex spanning tree) and all visible arcs are nonbasic. Also recall that the arc cost transfer policy seen in Procedure 1 is a generalized version

of the one used in the primal network simplex algorithm where only one root node is used. As each visible arc is compatible (appearing as a loop in $H(B^0, \mathbf{x}^0)$), applying cost dominance trivially results in a single contender for the pricing problem to identify which incidentally reconciles us with the entering variable of Dantzig's pivot rule.

Let $\mathcal{C}_P$ denote the set of all directed cycles obtainable from extreme points of the pricing problem (9). The cardinality of this set can be measured in two parts, that is, the compatible and incompatible portions. The first portion is insignificant and reduces to a single element regardless of the set P while the second grows exponentially as the set P reduces in cardinality.

Proposition 7 *The cardinality of the set of extreme point solutions $|\mathcal{C}_P|$ of the pricing problem (9) on the contracted network $H(P, \mathbf{x}^0)$ is $O(2^{(n-|P|+1)^2})$.*

Proof. For the sake of simplicity, the argument is carried with a dominance rule applied on the visible arcs. Furthermore, we use a worst case type analysis on the density of these sets. As an extreme point of (9) can only identify a single compatible arc, dominance can be applied across all compatible arcs regardless of their root node association.

The more interesting portion thus concerns incompatible combinations which of course require at least two incompatible arcs. Using combinatorial enumeration, let us count the cycles in rough numbers by randomly selecting arcs within the available possibilities $V_P(\mathbf{x}^0)$ thus forming directed cycles of different sizes ranging from 2 to $n - |P| + 1$, the number of nodes in the contracted network, i.e., $|\mathcal{C}_P| = \sum_{k=2}^{n-|P|+1} \binom{|V_P(\mathbf{x}^0)|}{k}$. Then again, once dominance is applied, the set of visible arcs vastly overestimates the actual number of arcs remaining in the contracted network which for a complete graph amounts to $(n - |P| + 1)^2$. Since basic calculus reduces $\sum_{k=1}^{n} \binom{n^2}{k}$ to the dominant term 2^{n^2}, we obtain $|\mathcal{C}_P| \in O\left(2^{(n-|P|+1)^2}\right)$. □

Granted it is possible to order the cardinalities, $|\mathcal{C}_\emptyset| \gg |\mathcal{C}_F| \gg |\mathcal{C}_B|$, the same cannot be done for the actual sets. Indeed, hidden arcs and dominance rules means that directed cycles present in a smaller set are not necessarily present in a larger one. Consider for instance Figures 10b and 10c where the cycle uses the non-dominated arc $(8, 5)$ for the former and $(7, 4)$ for the latter.

While this computation tremendously dramatizes the size of the extreme point solution set of the pricing problem (9), it gives the general intuition that some balance can be achieved between the workload offset transferred to the oracle and the simplicity of obtaining undirected cycles that may induce a step size of zero. Let us put this in perspective of the contracted networks seen in Figures 3b, 4b, and 5b. The formulation of the pricing problem (9) is always the same, yet the contracted network reminiscing of the primal network simplex algorithm (5b) has $|\mathcal{C}_B| = 1$ whereas many more cycles are present in the contracted network using all free variables (3b). While the density of the resulting contracted network converges towards the density of the original network such that the exponential bound becomes increasingly approximative as $|P|$ get smaller, it is nevertheless expected that the difficulty of solving the pricing problem (9) follows the burden of the extreme point solution set.

Furthermore, for the exponential cardinality growth to be in line with Proposition 6, one must realize that additional cycles available through a smaller set P must come from the combination of cycles available in a larger set P. Consequently, since cycles available using the set $P = F$ are sufficient to ascertain optimality, some of the cycle combinations available in the pricing problem using set $P \subset F$ must lead to nonbasic solutions. In this matter, the purpose of Section 3.1 which handles nonbasic solutions is twofold. Sure enough, it manages the initial starting solution but more importantly solutions encountered during the solution process. That is, in particular there is no guarantee that MMCC (with $P = \emptyset$) travels through extreme points and the same is true for other cases where actual free arcs are rendered visible in V_P, that is, when $P \subset F$.

3.7 Root selection

Since the selection of root nodes is arbitrary, the reader might wonder what impact, if any, a different set of root nodes would have on the oracle and thus the algorithm's course. As shown in the proof of Proposition 8, it turns out very little. Indeed, the oracle's content is modified on a per component basis yet as a whole it is completely unaffected.

Proposition 8 *The root selection has no influence on the pricing problem (9). The compatible set, the extreme point solution set $\mathcal{C}_P$, the mean cost evaluation μ_H and the nature of the optimality certificate in Proposition 6 are unaltered.*

Proof. On the one hand, it is no surprise that, for all nodes $i \in N$, the paths $\mathcal{P}(i)$ and their costs π_i are modified when an alternative set of root nodes is used. This means that while each visible arc (i, j) of $V_P(\mathbf{x}^0)$ continues to exist on $H(P, \mathbf{x}^0)$, it is now associated with a different reduced or rooted cost $\bar{d}_{ij} = d_{ij} - \pi_i + \pi_j$. The first observation is that the set of extreme point solutions $\mathcal{C}_P$ corresponding to directed cycles obtained by linear combination of visible arcs is unaltered. Furthermore, by Proposition 4 the original cost on each cycle is maintained regardless of the root selection. As such, the mean cost of every extended cycle is also maintained which obviously means as much for the nature of the optimality certificate. Finally, observe that not only is the compatibility status of any arc (i, j) persistent, the reduced or rooted cost of a compatible arc is also immune to change. □

This is a testament to the fact that a degenerate solution grants degrees of freedom for dual variables, in fact one for each degenerate constraint in accordance with Procedure 1. Intuitively speaking, the pricing problem (9) can be interpreted to optimize over these degrees of freedom. Indeed, solving the former not only provides an optimal cycle W_H^0 of minimum mean reduced cost μ_H^0 on the contracted network but also a new set of dual variable values for the root nodes. The following is an adaptation of Goldberg and Tarjan (1989, Theorem 3.3) or, more directly in terms of notation, of Gauthier et al. (2015, Proposition 3) which also bases its argument on linear programming theorems.

Proposition 9 *When solving the pricing problem (9) at iteration $k \geq 0$, there exists some dual variable values π_ℓ^k, $\ell \in N_P(\mathbf{x}^k)$, such that $\bar{d}_{ij} - \pi^k_{\mathcal{R}(i)} + \pi^k_{\mathcal{R}(j)} \geq \mu_H^k$, $(i, j) \in V_P(\mathbf{x}^k)$. By complementary slackness, the latter in fact hold at equality for all strictly positive variables, i.e., $(i, j) \in W_H^k$.*

In other words, the initial dual variable values on root nodes are irrelevant as much as the roots selection. Either way, the pricing problem (9) optimizes dual variables aiming to satisfy optimality conditions.

4 Behavioral study

Supported by numerical results from a minimum cost flow problem instance containing 1 025 nodes and 92 550 arcs (referred to as Instance 1), in Section 4.1 we analyze the behavior of a specific variant of the contraction algorithm opposite MMCC's established behavior. This study not only serves to grasp some mechanical aspects of these algorithms, but more importantly draws attention on key points of the theoretical minimum mean cycle-canceling complexity. Section 4.2 connects the dots with Cancel-and-Tighten and digs into more advanced aspects.

4.1 A lower bound on the minimum mean cost

Our analysis opposes the mean cost of negative cycles obtained on the contracted network to those that would be obtained in the framework of MMCC. At a given iteration $k \geq 0$, let μ_G^k be the minimum mean cost of the directed cycle W_G^k on the residual network as obtained by MMCC. We then denote by μ_H^k the

minimum mean cost of the directed cycle W_H^k fetched on the contracted network and

$$\mu_{H:G}^k = \mu_H^k \frac{|W_H^k|}{|W_{H:G}^k|} \tag{12}$$

the mean cost of the extended cycle where hidden arcs are accounted for. The following proposition is verified by construction.

Proposition 10 *Let $\mathbf{x}^k$, $k \geq 0$, be a non-optimal solution. Given the set $P \subseteq F^k$, the following ordering of the minimum mean costs always holds:*

$$\mu_H^k \leq \mu_G^k \leq \mu_{H:G}^k. \tag{13}$$

Proof. The cycle $W_{H:G}^k$ is visible as is on the residual network $G(\mathbf{x}^k)$ which means $\mu_G^k \leq \mu_{H:G}^k$ trivially holds by the nature of minimizing the mean cost in $G(\mathbf{x}^k)$. Furthermore, although contracted, any cycle W_G^k that also appears in $H(P, \mathbf{x}^k)$ is evaluated with a different mean cycle cost which eventually uses less arcs such that $\mu_H^k \leq \mu_G^k$. □

Since we aim to establish complexity results in light of MMCC's analysis, it is quite natural to think of W_G^k as the reference minimum mean cost cycle. The inequalities literally state that the optimal value of the minimum mean cost μ_G^k may be underestimated by μ_H^k and overestimated by $\mu_{H:G}^k$. The notion of estimation might be better understood by observing the evolution of the various μ during the solution process. For illustrative purposes, we systematically use the contracted network $H(F^k, \mathbf{x}^k)$ at every iteration, where all and only the free arcs are hidden. The value of μ_H^k and its counterpart $\mu_{H:G}^k$ are naturally obtained on top of which we also poll for the value μ_G^k as if to look for the minimum mean cycle. Figure 11 shows the evolution of these measures for Instance 1.

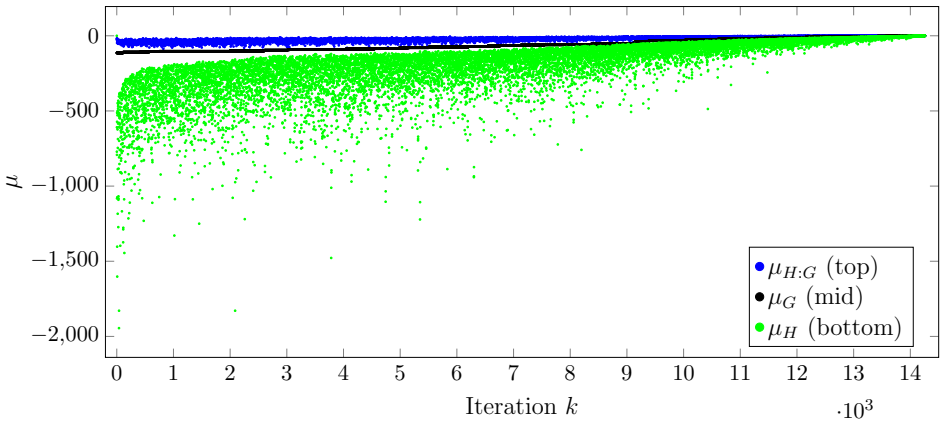


Figure 11: Comparison of μ_H^k, μ_G^k and $\mu_{H:G}^k$

While all three plots show a general increasing tendency, what is striking is how different the inequality $\mu_H^k \leq \mu_G^k$ is from $\mu_G^k \leq \mu_{H:G}^k$. The first gap is fairly intuitive and goes back to basic mathematics: The density of the contracted network produces contracted cycles which use relatively few arcs compared to their extended counterparts. As the cycle costs remain the same, the denominator strongly influences the mean evaluation. The ordering (13) can then be interpreted as a deceptively small minimum mean cost μ_H^k whose value is corrected when accounting for the hidden arcs. Observe that the ordering (13) is equal throughout if there are no hidden arcs in the contracted cycle, i.e., $|W_H^k| = |W_{H:G}^k|$.

The second gap is much more tight and deserves more attention. In this matter, Figure 12 zooms on the evolution of the extended mean cycle cost $\mu_{H:G}^k$ and minimum mean cycle cost μ_G^k. For the record, MMCC both searches for and applies a minimum mean cycle at each iteration, yielding an algorithm which features

a non-decreasing property on μ_G^k, see Goldberg and Tarjan (1989, Lemma 3.5) or Gauthier et al. (2015, Proposition 4). Take a close look around the $10\,000^{\text{th}}$ iteration. This is not a graphical aberration showing that this property is indeed lost when negative cycles are not canceled in the order suggested by MMCC. What we think is surprising for this particular instance is how little this phenomenon occurs, only 21 times within 14 258 iterations to be exact.

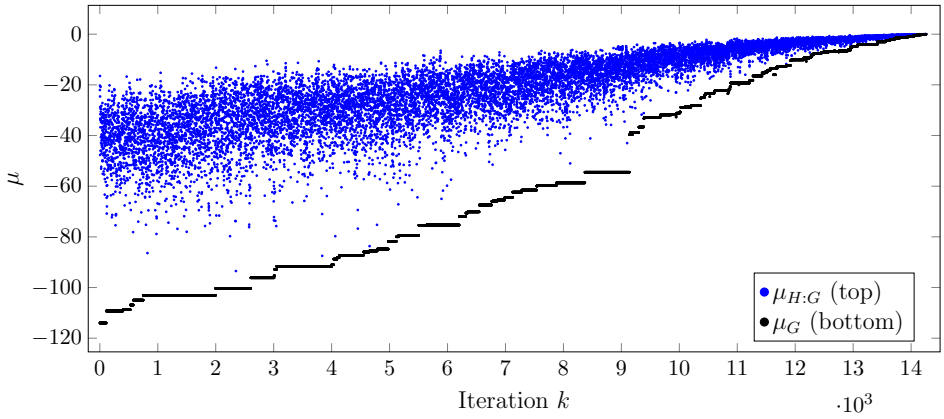


Figure 12: Zoom on comparison of μ_G^k and $\mu_{H:G}^k$

Convergence. As the contraction algorithm identifies a negative cycle at every iteration, it is evident that convergence of the objective function to optimality is guaranteed. By default, there are then at most $O(mCU)$ such negative cycles, a weakly polynomial result referring to the largest absolute cost ($C := \max_{(i,j)\in A} |c_{ij}|$) and interval bound values ($U := \max_{(i,j)\in A} u_{ij} - \ell_{ij}$).

Nevertheless, comparing the canceled cycle mean cost to that of the minimum mean cycle cost is an enlightening exercise. Indeed, the erratic behavior also appears in a strongly polynomial algorithm, namely Cancel-and-Tighten. Section 4.2 recalls some concepts from the minimum mean cycle-canceling algorithm (Gauthier et al. 2015) for which strongly polynomial runtime is established through the analysis of the so-called *optimality parameter*.

4.2 Optimality parameter analysis

Strongly polynomial runtime complexity certifies that an algorithm performs a polynomial number of operations measured solely by the size of its input (see Cormen et al. 2009). For network flow problems, this number should be a function of m and n only. In introducing the minimum mean cycle-canceling algorithm, Goldberg and Tarjan (1989) also provide such a complexity proof which holds in two parts. First, an oracle capable of producing a *minimum mean cost directed cycle* in $O(mn)$ time. Second, an optimal solution is reached by canceling at most $O(m^2 n \log n)$ such negative cycles. The first part can be seen as the inner loop and the second as the outer loop. Their product then yields a strongly polynomial global complexity. Radzik and Goldberg (1994) refine the complexity analysis, reducing the number of cycle cancellations to $O(m^2 n)$. Although they also introduce the concept of *phases* to analyze the behavior of the algorithm, Gauthier et al. (2015) strongly insist on the latter to further improve the complexity result by combining the Cancel-and-Tighten strategy introduced in Goldberg and Tarjan (1989) with the original algorithm.

In MMCC, the value μ_G^k is coined as the *optimality parameter* because it converges without oscillations to 0 from below in strongly polynomial time. It might however be more appropriate to reserve this name for another value as illustrated by the upcoming analysis.

Type 2 (negative) cycles. Even though the non-decreasing property of μ_G^k across iterations in the minimum mean cycle-canceling algorithm is interesting, it has been argued by Gauthier et al. (2015) that the strictly increasing behavior observed across *phases* is more enlightening. The phase definition goes hand in hand with

the proof of Gauthier et al. (2015, Proposition 5) which distinguishes between Type 1 and Type 2 cycles. It shows that a Type 2 cycle is attained within m cancellations or optimality is achieved, where a Type 2 cycle is obtained when there exists at least one variable in the cycle with a non-negative reduced cost computed with respect to a set of dual variables established at the beginning of a phase. Given μ_G^h at the beginning of phase $h \geq 0$, such a Type 2 cycle W^ℓ then serves to imply a strict increase on μ_G^h as

$$\mu_G^\ell \geq (1 - 1/|W^\ell|)\, \mu_G^h \geq (1 - 1/n)\, \mu_G^h \tag{14}$$

thus marking the end of the phase, i.e., the sequence of iterations leading to this *measurable jump factor*.

Since each phase $h \geq 0$ contains at most m cancellations, the number of cancellations can be interpreted as at most $O(mn \log n)$ or $O(mn)$ phases depending on the complexity point of view (Gauthier et al. 2015). These points of view respectively refer to Theorem 2 which uses the minimal increasing factor $(1 - 1/n)$ and Theorem 3 rather exploiting the stronger factor $(1 - 1/|W^\ell|)$. While the concept of phases is useful for the complexity analysis of MMCC, it is not transparent at all in the implementation. Indeed, in MMCC, the phase is purely a question of theoretical existence; dual variables are never required to advance such that the solution process cares not about these cycle Types. The enlightenment comes from the inclusion of Cancel-and-Tighten in the analysis where phases are observed *in actu*. The latter fixes dual variables and depletes the residual network of Type 1 cycles (those formed with negative reduced cost arcs only). Once this Cancel-step is terminated, one must conclude that a Type 2 cycle comes next thus implying a jump with respect to some lower bound $\hat{\mu}_{CT}^h$ on μ_G^h at the beginning of phase h.

Figure 13 opposes the mean value μ_{CT}^k at iteration k of the Type 1 cycles canceled in the Cancel-and-Tighten implementation with the lower bound value $\hat{\mu}_{CT}^k \leq \mu_G^k$ proposed by Gauthier et al. (2015, Proposition 14). While the erratic behavior of μ_{CT}^k can clearly be observed throughout the solution, a general pattern of increase can be noted across the phases. The minimum (i.e., optimal) mean cycle value μ_G^k is once again fetched as background information. The $4\,900^{\text{th}}$ iteration is worth a look: Again, a small decrease for μ_G^k. In total, four such occurrences within 7 294 cancellations contained in 357 phases. Furthermore, Cancel-and-Tighten maintains strongly polynomial runtime despite the usage of Type 1 cycles going against the non-decreasing property of μ_G^k. The *strictly increasing lower bounds* $\hat{\mu}_{CT}^h$ *between phases* obtained with the existence of a Type 2 cycle marking the end of phase h is indeed where the properties are established (Gauthier et al. 2015, Theorem 6).

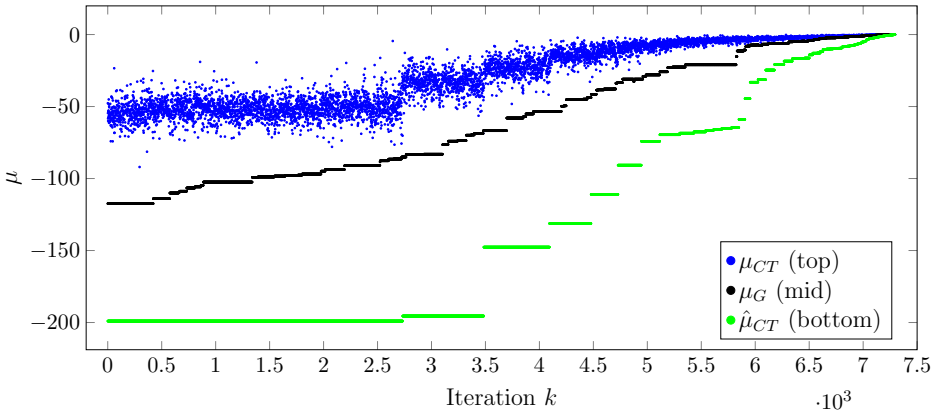


Figure 13: Comparison of μ_{CT}^k, μ_G^k and $\hat{\mu}_{CT}^k$

Arc fixing. Strongly polynomial time complexity is achieved by keeping track of the number of phases through the concept of *arc fixing* as seen in the minimum mean cycle-canceling algorithm (Gauthier et al. 2015, Propositions 9 and 11). A relaxation of the arc fixing condition is used in Proposition 11, where the two different arc fixing conditions are in line with the complexity point of views, that is, $O(mn \log n)$ or $O(mn)$ phases. The proof is straightforward and recuperates the Cancel-and-Tighten proof adaptation where the

lower bound values $\hat{\mu}^h_{CT}$ are shown to mimic the behavior of the true value μ^h_G. Indeed, as mentioned in Gauthier et al. (2015, p. 131), it is conceivable to rewrite the arc fixing conditions using a lower bound $\hat{\mu}^h$ on μ^h_G instead.

Proposition 11 *Assuming $\hat{\mu}^h$ is a lower bound for μ^h_G at the beginning of phase $h \geq 0$.*

- *Arc fixing for arc $(i,j) \in A$ occurs or has already occurred if $|\bar{c}^h_{ij}| \geq -2n\hat{\mu}^h$.*
- *Implicit arc fixing for arc $(i,j) \in A$ occurs or has already occurred if $|\bar{c}^*_{ij}| > -n\hat{\mu}^h$, where $\bar{c}^*_{ij}$ is the reduced cost of arc (i,j) computed with an optimal set of dual variables.*

It is at this point important to distinguish between μ^k_G at iteration k and strongly polynomial runtime. Indeed, whether one looks at μ^k_H or μ^k_{CT} or any other cycle-canceling method, the minimum mean cycle value μ^k_G can always be fetched as secondary information (recall that $\mu^k_H \leq \mu^k_G \leq \mu^k_{H:G}$ and $\hat{\mu}^k_{CT} \leq \mu^k_G \leq \mu^k_{CT}$). If strongly polynomial properties are to be established on oscillating values from one iteration to the next, it appears mandatory to rather divert the analysis to a lower bound $\hat{\mu}^h$ over the phases instead. In other words, while the portion above μ^k_G (blue) may be unpredictable, it is the portion below μ^k_G (green) that should be well-behaved. We are ready to propose an adaptation of the contraction algorithm. Whether that adaptation is necessary to achieve a strongly polynomial time complexity is an open question although, since the proof relies on a controllable behavior of μ_H, we conjecture that it is.

5 Contraction-Expansion algorithm

Section 5.1 introduces a flexible phase definition based on so-called Type 3 cycles that serve as end-phase markers. This is followed in Section 5.2 by a discussion of an expansion scheme which controls the visible and hidden arc sets, that is the content of the set P, for the proposed algorithm. Section 5.3 argues that applying the contraction on the residual network and expanding the contracted network using that specific expansion scheme as the algorithm progresses produces a strongly polynomial algorithm. In Section 5.4, we show that the expansion scheme is not unique such that different strategies can be used to improve the algorithm. Finally, computational results are presented in Section 5.5.

5.1 End-phase markers

Our contribution is to revisit the phase definition in order to extract the true pertinent information which allows convergence in strongly polynomial time. In this respect, the current phase definition is built upon a weak jump condition which waits for the identification of a Type 2 cycle as the minimum mean cycle to confirm a jump on μ^h_G at the end of phase h. Let us propose a more flexible definition.

Definition 3 *A* phase $h \geq 0$ *is a sequence of cycle cancellations terminated whenever a* measurable jump *is observed in strongly polynomial metrics, that is, both the factor and the time required to obtain it are expressed in strongly polynomial time. A phase solution $\mathbf{x}^h$ is understood as the solution at the beginning of phase h.*

The factor proposed in the following cycle Type obviously satisfies the strongly polynomial requirement whereas the time requirement is shown in Proposition 12.

Definition 4 *Let $\mathbf{x}^h$, $h \geq 0$, be a non-optimal phase solution. At an iteration t within the phase h, let us call a cycle W^t on $G(\mathbf{x}^t)$ a* Type 3 (negative) cycle *if its underestimated mean cost $\hat{\mu}^t$ (with $\hat{\mu}^t \leq \mu^t_G \leq \mu^t$) produces the measurable jump*

$$\hat{\mu}^t \geq \left(1 - \frac{1}{|W^t|}\right) \mu^h_G. \tag{15}$$

Proposition 12 *In MMCC, a Type 3 cycle is identified within at most m cancellations.*

Proof. Consider the phase h. The proof is trivial and connects with the existence of a Type 2 cycle, say W^ℓ, which must have $\mu_G^\ell = \hat{\mu}^\ell \geq (1 - 1/|W^\ell|)\,\mu_G^h$. Therefore, the iteration t either happens simultaneously to ℓ or earlier. □

Observe that whether the jump factor is obtained using an actual Type 2 cycle or not is irrelevant: A phase is completed in accordance with Definition 3. In other words, the only important aspect of the Type 2 cycle is the measurable jump it procures on μ_G^h, a tactic which can incidentally also be verified against a lower bound according to Proposition 11, that is,

$$\hat{\mu}^t \geq \left(1 - \frac{1}{|W^t|}\right) \hat{\mu}^h. \tag{16}$$

This is true for $\hat{\mu}^h = \hat{\mu}_{CT}^h$ in Cancel-and-Tighten (Gauthier et al. 2015, Theorem 3) and is also used with $\hat{\mu}^h = \mu_H^h$ in the proposed Contraction-Expansion algorithm.

5.2 Expansion scheme

The focus on phases should by now be realized by the reader. Figure 14 presents a pseudo-code for the Contraction-Expansion algorithm whereas the following paragraphs explain how the proposed expansion scheme constrains the latter into producing such phases.

Initialization: Iteration $k := 0$;
Phase $h := 0$, $\hat{\mu}^0 := -\max_{(i,j)\in A} |c_{ij}|$, *new phase* := true;
Feasible solution $\mathbf{x}^0$;
if *new phase* **then**
  new phase := false;
  counter := 0;
  Eliminate cycles of free arcs from $\mathbf{x}^k$;
  Build partition F^k, L^k, U^k;
  Define the set $P^k := F^k$;
Derive the contracted network $H(P^k, \mathbf{x}^k)$;
Solve the pricing problem (9) for μ_H^k, W_H^k;
If $\mu_H^k \geq 0$, **terminate** with an optimality certificate for $\mathbf{x}^k$;
Extract the extended cycle $W_{H:G}^k$;
Compute the strictly positive step size ρ^k from (10);
Update to solution $\mathbf{x}^{k+1}$ using the system (11);
if $\mu_H^k \geq (1 - 1/|W_{H:G}^k|)\,\hat{\mu}^h$ **then**
  $\hat{\mu}^{h+1} := \mu_H^k$;
  $h := h + 1$;
  new phase := *true*;
else
  counter := *counter* + 1;
  if *counter* < n **then**
    Expand the contracted network with $P^{k+1} := F^{k+1} \cap F^k$;
  else
    $P^{k+1} := \emptyset$;
Iterate $k := k + 1$;

Figure 14: Contraction-Expansion algorithm

Let $\mathbf{x}^0$ be a non-optimal solution at iteration $k = 0$. Note that $\mathbf{x}^0$ is also a non-optimal phase solution ensuring a lower bound $\hat{\mu}^0 \leq \mu_G^0$. Let the set $P^0 = F^0$. By Proposition 6, the extended cycle $W_{H:G}^0$ is directed on the residual network $G(\mathbf{x}^0)$ and the step size is strictly positive.

When $W_{H:G}^0$ is canceled, the aftermath is hard to predict but one thing is for certain: Only the arcs part of $W_{H:G}^0$ are affected. Some arcs that were free in $\mathbf{x}^0$ remain free in $\mathbf{x}^1$ in the next iteration while

all the other arcs have changed status either from restricted lower to restricted upper and vice versa, from free to restricted or from restricted to free for a total of six possibilities. Four of these end up with a new status restricted whereas two end up with a free status. Intuitively speaking, since the contraction happens around the free variables and the non-decreasing μ_G^k property is lost when the contraction is systematically applied, let us concentrate on controlling these newly freed variables in $\mathbf{x}^1$, i.e., $F^1 \setminus F^0$. By applying coerced degeneracy on these specific variables, the hidden set then contains only those variables that were already free in $\mathbf{x}^0$ and are still free in $\mathbf{x}^1$. This amounts to selecting the set $P^1 = F^1 \cap F^0$ at iteration $k = 1$.

From there, the idea is simple, repeatedly apply cycle cancellation with the extended cycle $W_{H:G}^k$ at iteration k and expand the contracted network using the intersection of variables free in both the previous and new solution until the algorithm reaches a Type 3 cycle, that is, a negative cycle producing a jump factor of at least $(1 - 1/|W_{H:G}^k|)$. The phase $h = 0$ is then terminated and a new phase begins. Since we may at this point re-apply the full contraction, it is worthwhile to eliminate any cycle of free arcs from the phase solution to maximize the contraction benefit.

5.3 Complexity analysis

The complexity analysis of the Contraction-Expansion algorithm revolves around bringing the solution process of the latter to terms with MMCC's behavior. As testified by Cancel-and-Tighten, these terms refer to the strictly increasing phase markers. Figure 15 might help to get a feel for this endeavor. It displays the value of μ_H^k using the minimal expansion scheme $P^{k+1} = F^{k+1} \cap F^k$ where each trail corresponds to a phase during the solution of Instance 1.

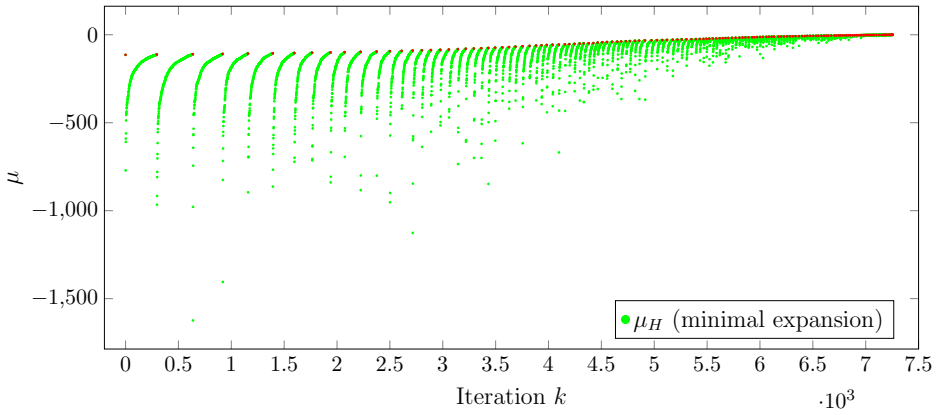


Figure 15: Evolution of μ_H^k with the Contraction-Expansion algorithm

The upcoming analysis focuses on the outer loop, that is, we examine the time required to reach the end of a phase and the total number of such phases.

Proposition 13 *The Contraction-Expansion algorithm completes a phase in $O(m^2 n)$ time.*

Proof. The proof is threefold with respect to the expansion scheme portion of the algorithm. We prove that 1) the non-decreasing property of μ_H^k during the phase is maintained, 2) at most $O(n)$ contracted cycle cancellations are required to reach a contracted network equivalent to the residual network, and 3) finding the minimum mean reduced cost cycle on the contracted network requires $O(mn)$ time.

Gauthier et al. (2015, Proposition 4) state that a cycle cancellation on the residual network $G(\mathbf{x}^k)$ cannot introduce a minimum mean cycle in $G(\mathbf{x}^{k+1})$ yielding $\mu_G^{k+1} < \mu_G^k$ in the following iteration. Here is an adaptation for the expansion scheme. Consider the contracted network $H(P^k, \mathbf{x}^k)$ and an optimal contracted cycle W_H^k of mean reduced cost μ_H^k along with an optimal set of dual variables π_ℓ^k, $\ell \in N_P(\mathbf{x}^k)$, on the root nodes (Proposition 9). From there, fix the root nodes' dual variables to these new values and update

the remaining dual variables accordingly (Procedure 1). The residual network $G(\mathbf{x}^k)$ hence satisfies $\bar{d}_{ij} - \pi^k_{\mathcal{R}(i)} + \pi^k_{\mathcal{R}(j)} \geq \mu^k_H$, $(i,j) \in A(\mathbf{x}^k)$, and in particular $\bar{d}_{ij} = 0$, $(i,j) \in H_P(\mathbf{x}^k)$. Upon canceling the expanded cycle $W^k_{H:G}$ and obtaining $\mathbf{x}^{k+1}$, every new residual arc in $G(\mathbf{x}^{k+1})$ either has a reduced cost of 0 or $-\mu^k_H$, i.e., $\bar{d}_{ij} - \pi^k_{\mathcal{R}(i)} + \pi^k_{\mathcal{R}(j)} \geq \mu^k_H$, $(i,j) \in A(\mathbf{x}^{k+1})$. Observe that every arc in $F^k \cap F^{k+1}$ already has a reduced cost of 0 such that the arc cost transfer policy already holds. The contraction is readily available with every remaining visible arc evaluated at a reduced cost greater than or equal to μ^k_H. The mean cost of the next contracted cycle is then at least μ^k_H.

Recall that the pricing problem in MMCC can be derived from the pricing problem (9) by making visible all residual arcs, that is, by setting $P = \emptyset$. Then, observe that the set P is updated by intersecting the sets of free variables of the previous solution with the current one such that its size either stays the same or decreases at each cancellation. Assuming the initial phase solution $\mathbf{x}^0$ is a basic solution, at most $|F^0| \in O(n)$ cancellations yielding a decrease are then obviously required to reach $P = \emptyset$. Furthermore, when a cycle is canceled without modifying the set of free arc variables, it means that one or several restricted variables change from one bound to another. Unfortunately, this kind of phenomenon gives meaning to the worst case complexity bound observed in pathological instances. An iteration counter limiting the number of cancellations prior to reaching the residual network to at most n is put in place to make the move directly should it be necessary. Trivially, at most n cycle cancellations allow the expansion scheme to reach a contracted network equivalent to the residual network. From there, a Type 3 cycle is ensured within an additional m cycle cancellations (Proposition 12). To sum up, at most $n + m \in O(m)$ cycle cancellations are required to meet the required jump.

Finally, solving the pricing problem (9), which can still be accomplished in $O(mn)$ time using dynamic programming (Karp 1978), dominates all the other operations performed at every iteration. Indeed, the data structure is maintained in $O(m)$ time while the extended cycle extraction, step size computation and solution update is done in $O(n)$ time. A phase is ultimately completed in at most $O(m)$ iterations each one requiring at most $O(mn)$ for a total phase runtime of $O(m^2 n)$. □

Theorem 1 *The Contraction-Expansion algorithm runs in $O(m^3 n^2)$ strongly polynomial time.*

Proof. We show that at most $O(mn)$ phases can occur in accordance with Gauthier et al. (2015, Theorem 3). Since $\mu^k_H \leq \mu^k_G$ is indeed a lower bound for the true minimum mean cycle cost value by Proposition 10, this is also true for any phase solution. As soon as $\mu^k_H \geq \hat{\mu}^h \left(1 - 1/|W^k_{H:G}|\right)$, the phase h ends and the lower bound is increased to $\hat{\mu}^{h+1} := \mu^k_H$. By Proposition 11, arc fixing occurs on the lower bound value $\hat{\mu}^h$ as well such that the phase time complexities are valid. By Proposition 13, since each phase runs in $O(m^2 n)$, the compound time is obtained. An initial valid lower bound for μ^0_G can be trivially obtained with $\hat{\mu}^0 := -\max_{(i,j)\in A} |c_{ij}|$. □

The proposed phase definition along with the Type 3 cycle not only satisfy theoretical properties of the strongly polynomial time complexity of MMCC, they also express very practical observations. The hope is that not only the Type 3 cycle occurs much earlier than after m cancellations, it also happens while the phase still exploits the contracted network.

5.4 Alternative end-phase markers and expansion schemes

So long as strongly polynomial phase time is maintained, alternative expansion schemes may be used. Figures 16–17 show the evolution of μ^k_H for two simple variations. The first variation (cycle expansion) updates the set P with a more aggressive reduction, i.e., a *faster* expansion. The update writes as

$$P^{k+1} = \left(F^{k+1} \cap F^k\right) \setminus \{(i,j) \in A \mid (i,j) \in W^k_{H:G} \text{ or } (j,i) \in W^k_{H:G}\} \tag{17}$$

such that all arcs of the expanded cycle $W^k_{H:G}$ that are still free (thus common to F^k and F^{k+1}) are also removed from P^{k+1}. The second variation (cycle expansion & loops) uses the same update and also uses

a post cycle-cancellation heuristic which cancels loops derived from all compatible variables with negative rooted costs as well.

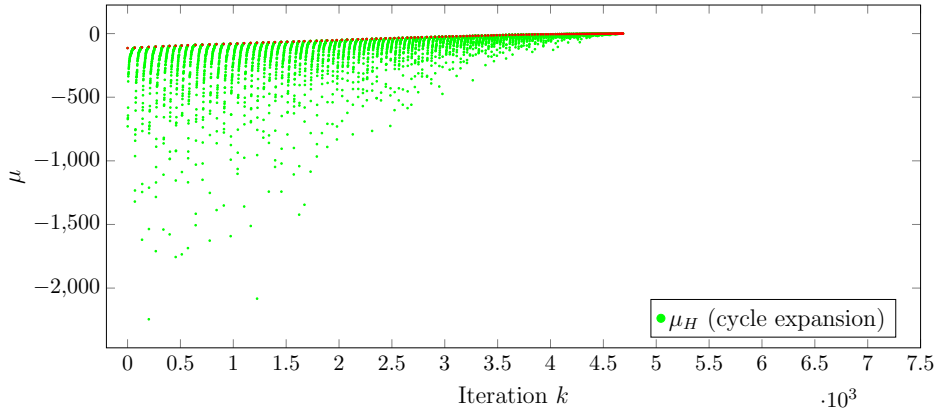


Figure 16: Evolution of μ_H^k with cycle expansion

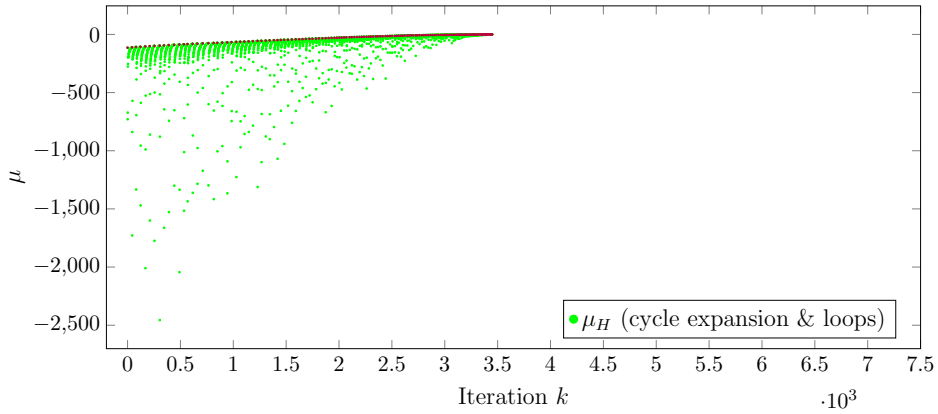


Figure 17: Evolution of μ_H^k with cycle expansion and cancellation of negative loops

For anyone familiar with successful divide-and-conquer methods, the solution speed typically benefits from the decomposition at a higher rate than the cost of the latter. While postponing the end of a phase with a less aggressive expansion scheme appears to agree with this statement, expanding the contracted network faster also means that end-phase markers are reached faster at which point the full contraction is re-applied. It seems that reaching said end-phase markers as fast as possible is of particular interest.

Speaking of end-phase markers, a measurable jump could be established using alternative Type 3 cycle definitions. Consider for instance the modification of (16) as follows:

$$\hat{\mu}^t \geq \left(1 - \frac{1}{|W^t|}\right)\hat{\mu}^h \geq \left(1 - \frac{1}{\max_{k\in h}|W^k|}\right)\hat{\mu}^h \geq \left(1 - \frac{1}{n}\right)\hat{\mu}^h. \tag{18}$$

where $k \in h$ is understood as an iteration k within the phase h. While the last criterion is reminiscing of Theorem 2 which contents itself with the same jump every time thus obtaining the $O(mn \log n)$ phases, the second criterion also tracks a cycle length and compromises on the desired jump. The latter is in fact the criterion used in all plots of the Contraction-Expansion algorithm.

5.5 Computational experiments

This section is separated in two parts. The first details computational results with respect to three specific instances and serves to show CE indeed behaves as expected when identifying smaller cycles on the contracted network. The second complements intuitive assertions made in the first part with a computational profile on benchmark instances from DIMACS (1990–1991). The latter contains several network problems of different nature codenamed by family `GOTO`, `GRIDGEN`, `GRIDGRAPH`, `NETGEN`, `ROAD`, and `VISION`. Each family is further divided into subcategories of different sizes each one containing five versions `A-E` which vary only in the data set. Algorithms are implemented in `C++`. All results are obtained using an `Intel i7-4770K@3.50GHz` with 16GB of RAM, running `Windows 10`.

Table 1 resumes the content of plots displayed in Figures 15–17 along with their computational times as well as averages of cycle sizes and induced step sizes. The performances of MMCC and CT are added for order comparison purposes whereas results regarding CE also incorporate an average contraction level given by $\overline{\alpha}$ where $\alpha^k = |N_P(\mathbf{x}^k)|/n$, $k \geq 0$.

Table 1: Computational results for variations of CE vs. MMCC and CT

Solution	CPU (sec)	k / cycles	h	$\overline{\lvert W \rvert}$	$\overline{\rho}$	$\overline{\alpha}$
Instance 1						
MMCC	91.52	4 296	90	27.11	1.30	–
CE - minimal expansion	87.59	7 256	151	5.46	1.30	0.32
CE - cycle expansion	63.98	4 686	144	5.19	1.70	0.47
CE - cycle expansion/loops	51.84	3 449/9 499	142	7.40/3.32	1.61/2.23	0.58
CT	0.82	7 294	357	21.62	1.37	–
gridgen_sr_13a						
MMCC	15 629.40	20 893	289	72.85	7.45	–
CE - minimal expansion	6 709.96	25 291	301	5.93	9.70	0.56
CE - cycle expansion	2 435.97	9 202	236	4.70	19.18	0.62
CE - cycle expansion/loops	1 906.97	6 034/23 429	224	7.13/2.58	18.53/36.01	0.70
CT	36.17	27 473	1 482	49.80	7.28	–
road_paths_05_WI_a						
MMCC	9 552.52	1 119	300	82.27	1.00	–
CE - cycle expansion	10 355.30	1 122	348	82.20	1.00	1.00
CT	2 389.51	3 248	18 647	34.85	1.00	–

First of all, there is no denying that Cancel-and-Tighten is orders of magnitude faster than MMCC. The proposed Contraction-Expansion algorithm sits somewhere in between although it is clear that the contraction boosts the speed of fetching negative cycles by a significant amount even more so as the size of the instance gets larger. The same feel is palpable across other benchmark instances such as the significantly larger problem gridgen_sr_13a from the `GRIDGEN` family which contains 8 191 nodes and 745 381 arcs. It is important to understand that the benefit of the contraction comes from the degeneracy observed in encountered solutions. For instance, the problem road_paths_05_WI_a ($n = 519\,157$ and $m = 1\,266\,876$) from the `ROAD` family is structured with uniform capacities at 1 such that there are never any free variables, hence no contraction. We therefore just get penalized with the contraction computational overhead.

The remainder of this section looks at the version `A` of the `GRIDGRAPH`, `NETGEN`, and `GRIDGEN` families, in the process, bringing further testimonies to the previous claims. The computational results obtained on the first family are presented in more details as follows. For each of the three categories of problems (**long**, **square**, **wide**), the number of arcs is roughly $m \approx 2n$ whereas the node size n ranges from 2^8 to 2^{17}, where the power is given by the instance difficulty index i. Table 2 displays the average contraction level and CPU time CPU_{CE} for CE. As expected the latter increases with the instance size, yet the average contraction level of about 60% appears stable across this family. The middle columns indicate the relative CPU time calculated with respect to MMCC's as $\beta := CPU_{CE}/CPU_{MMCC}$. For instance, CE requires 2 619 seconds to solve grid_square_16 meaning that MMCC took $2\,619/0.21 \approx 12\,471$ seconds to terminate. The relative

computing times β are plotted in Figure 18 with respect to the instance difficulty index i. Once again, the relative performance of CE increases with the instance size.

Table 2: Computational results for CE on the GRIDGRAPH family, version A

	long			**square**			**wide**		
i	CPU_{CE} (sec)	β	$\overline{\alpha}$	CPU_{CE} (sec)	β	$\overline{\alpha}$	CPU_{CE} (sec)	β	$\overline{\alpha}$
8	0.01	0.65	0.64	0.01	0.60	0.64	0.01	0.60	0.64
9	0.02	0.44	0.59	0.04	0.43	0.56	0.05	0.50	0.59
10	0.08	0.31	0.56	0.16	0.36	0.58	0.21	0.41	0.61
11	0.18	0.26	0.61	0.64	0.30	0.56	0.93	0.44	0.64
12	0.59	0.19	0.56	2.95	0.29	0.58	4.43	0.38	0.64
13	1.51	0.12	0.50	11.96	0.26	0.57	19.72	0.39	0.64
14	5.45	0.16	0.45	59.18	0.26	0.58	89.83	0.41	0.63
15	8.51	0.08	0.51	325.44	0.22	0.60	476.19	0.38	0.63
16	23.15	0.04	0.45	2 619.07	0.21	0.60	2 911.39	0.31	0.63
17	402.98	0.13	0.45	17 890.40	0.27	0.61	15 944.70	0.34	0.63

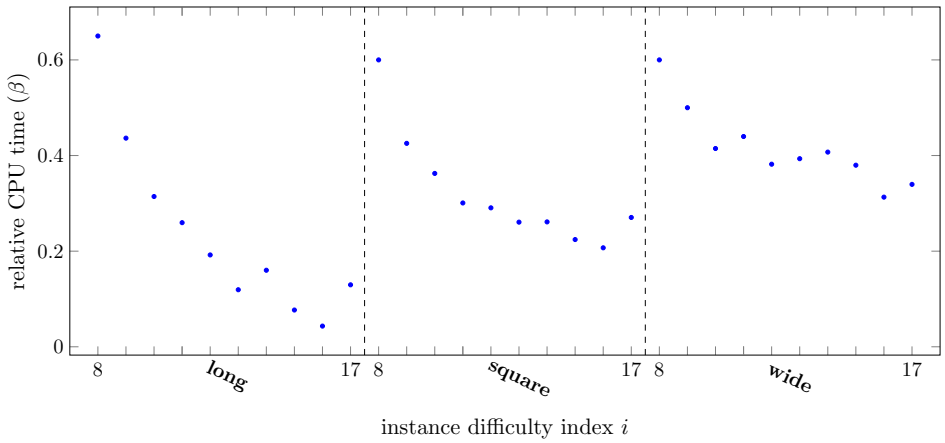


Figure 18: Relative CPU time for CE when solving the GRIDGRAPH family, version A

The NETGEN family is broken down into fives categories (**8**, **deg**, **lo_8**, **lo_sr**, **sr**). The category **deg** is stable in node size at $n = 4\,096$ although it increases exponentially in terms of arcs $m = n2^i$. All other problems range from sizes 2^8 to 2^{19} in terms of nodes n, and in terms of arcs either $m = 8n$ for **8** and **lo_8** or $m = n2^{i/2}$ for **lo_sr** and **sr** problems. The mean of the average contraction levels $\overline{\alpha}$ is about 93% although it is significantly closer to 100% for problems in the **lo_8** and **lo_sr** categories. Said lack of contraction is apparent in Figure 19 where β is mostly present above 1 represented by the horizontal dotted line.

The GRIDGEN family is partitioned into three categories of problems (**8**, **deg**, **sr**). The category **deg** is once again stable at $n = 4\,097$ and increases exponentially in terms of arcs $m = n2^i$. The two other categories range in node sizes n from 2^8 to 2^{18} whereas the number of arcs is given by $m = 8n$ for the former category and $m = n2^{i/2}$ for the latter. The mean of the average contraction levels $\overline{\alpha}$ is about 90% for these problems. Figure 20 displays the experiment outcome on this family and still features increasing relative performance as the instances grow in difficulty.

While we certainly did not cover the complete benchmark suite, this paper does not pretend to propose a competitive algorithm just yet. As a case in point, even larger instances up to 2^{22} nodes are available but MMCC's solution proved to be far too demanding. We also believe the omitted instance versions and/or more results from other families would not significantly contribute to this framework. Finally, we recall that PS is a top performer for the solution of minimum cost flow problems despite the fact that the vast majority of pivots performed, read over 70%, are degenerate (Ahuja et al. 1993, Figure 18.7). With this in mind, the goal of this paper is rather to understand better the mechanics of algorithms with strictly positive step size improvements.

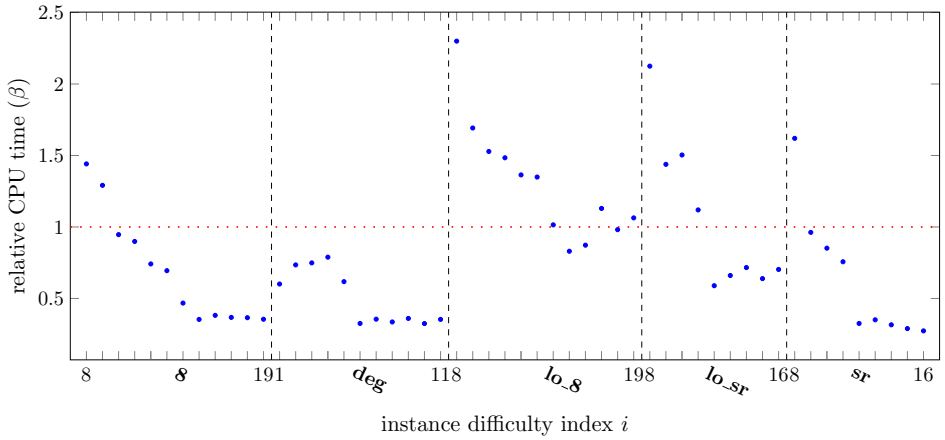


Figure 19: Relative CPU time for CE when solving the `NETGEN` family, version `A`

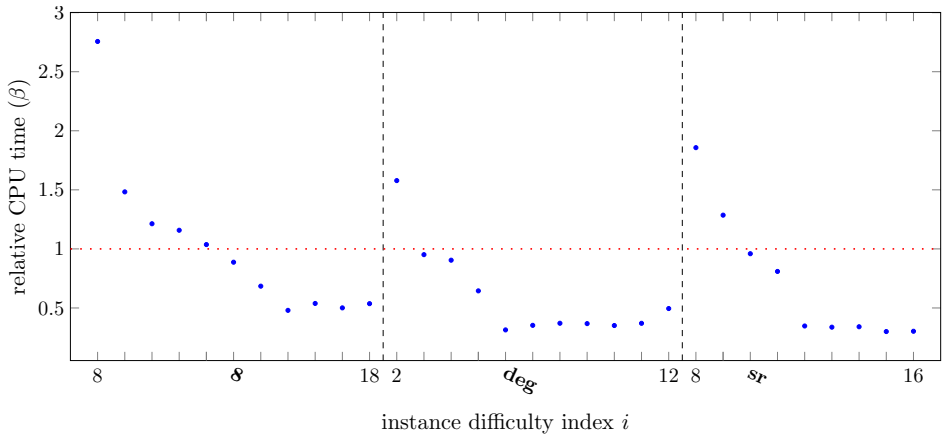


Figure 20: Relative CPU time for CE when solving the `GRIDGEN` family, version `A`

6 Conclusion

We start with a note addressed to users of the primal network simplex algorithm. Observe that the spirit of the minimum mean cycle-canceling algorithm is tangibly similar to that of PS. The pricing step is home to the optimality certificate whereby the latter is acquired unless an improving direction is identified. It turns out this is not all that surprising since it has been shown that PS and MMCC belong to (even constitute extreme cases of) a more generic contraction algorithm.

A variety of special cases inducing strictly positive step sizes is also identified. By combining these with results from MMCC, a strongly polynomial Contraction-Expansion algorithm which behaves much better than the former, especially as the instance's size increases, comes to life. The reader is carried around this birth process by opposing the behavior of MMCC and Cancel-and-Tighten in a computational study. Both iteration and phase bases are illustrated, although a strong emphasis on the latter concept is systematically done thus providing an alternative way of showing strongly polynomial runtime.

Such a property can also be observed in this framework when using partial contraction. The latter is obtained by modifying the choice of hidden arcs as the algorithm progresses. The selection is made in such a way that it actually corresponds to an *expansion* of the contracted network. Furthermore, the proposed Contraction-Expansion algorithm sticks to practical observations otherwise overlooked in MMCC. As such, phase markers are verified algorithmically rather than just existing for theoretical purposes. It even recuperates the idea that not all jumps are created equal thus underlining the important aspect of Type 2

cycles, namely the measurable jump. The Type 3 cycle is born. The strongly polynomial argument uses both phases and Type 3 cycles on top of which the convergence of the original *optimality parameter* is neglected in favor of a lower bound.

Furthermore, while it is true that the time complexity is not improved with respect to MMCC, we believe it is more interesting to note that the same time complexity is achieved despite a more complicated algorithm. Indeed, strongly polynomial algorithms might benefit from another look that aims to combine their time complexity with practical observations that make them behave more efficiently.

There seems to be some arbitrage to be done between trying to meet optimality conditions in a more aggressive manner and the work required to do so. By contenting itself with a sufficient condition, a significant proportion of cancellations performed in PS are degenerate whereas MMCC uses a rule whose computational footprint is too high. That being said, the contracted network is closer to the spirit of an oracle than is the residual network. By this, we mean that it matters not to see all directed negative cost cycles so long as at least one can be detected thus allowing to improve and repeat.

References

Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Upper Saddle River, NJ, USA, 1993.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms. The MIT Press, Cambridge, MA, USA and London, England, 3rd edition, 2009.

George B. Dantzig and Mukund N. Thapa. Linear Programming 2: Theory and Extensions. Springer Series in Operations Research and Financial Engineering (Book 2). Springer, New York, NY, USA, 2003.

DIMACS. Network Flows and Matching: First DIMACS Implementation Challenge, 1990–1991. ftp://dimacs.rutgers.edu/pub/netflow.

Jean Bertrand Gauthier, Jacques Desrosiers, and Marco E. Lübbecke. About the minimum mean cycle-canceling algorithm. Discrete Applied Mathematics, 196:115–134, 2015. doi:10.1016/j.dam.2014.07.005. Advances in Combinatorial Optimization.

Andrew V. Goldberg and Robert Endre Tarjan. Finding minimum-cost circulations by canceling negative cycles. Journal of the ACM, 36(4):873–886, 1989. doi:10.1145/76359.76368.

Richard Manning Karp. A characterization of the minimum cycle mean in a digraph. Discrete Mathematics, 23(3): 309–311, 1978. doi:10.1016/0012-365X(78)90011-0.

Morton Klein. A primal method for minimal cost flows with applications to the assignment and transportation problems. Management Science, 14(3):205–220, November 1967.

Péter Kovács. Minimum-cost flow algorithms: an experimental evaluation. Optimization Methods and Software, 30 (1):94–127, 2015. doi:10.1080/10556788.2014.895828.

Roy E. Marsten, Matthew J. Saltzman, David F. Shanno, George S. Pierce, and J. F. Ballintijn. Implementation of a dual affine interior point algorithm for linear programming. ORSA Journal on Computing, 1989. doi:10.1287/ijoc.1.4.287.

Tomasz Radzik and Andrew V. Goldberg. Tight bounds on the number of minimum-mean cycle cancellations and related results. Algorithmica, 11(3):226–242, 1994. doi:10.1007/BF01240734.