

**A strongly polynomial
Contraction-Expansion algorithm
for network flow problems**

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

G-2016-18

March 2016

Cette version est mise à votre disposition conformément à la politique de libre accès aux publications des organismes subventionnaires canadiens et québécois.

Avant de citer ce rapport, veuillez visiter notre site Web (<https://www.gerad.ca/fr/papers/G-2016-18>) afin de mettre à jour vos données de référence, s'il a été publié dans une revue scientifique.

This version is available to you under the open access policy of Canadian and Quebec funding agencies.

Before citing this report, please visit our website (<https://www.gerad.ca/en/papers/G-2016-18>) to update your reference data, if it has been published in a scientific journal.

Les textes publiés dans la série des rapports de recherche *Les Cahiers du GERAD* n'engagent que la responsabilité de leurs auteurs.

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2016
– Bibliothèque et Archives Canada, 2016

The authors are exclusively responsible for the content of their research papers published in the series *Les Cahiers du GERAD*.

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2016
– Library and Archives Canada, 2016

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.desrosiers@gerad.ca
marco.luebbecke@rwth-aachen.de

March 2016

**Les Cahiers du GERAD
G-2016-18**

Copyright © 2016 GERAD

Abstract: This paper addresses the resolution of the capacitated minimum cost flow problem on a network containing n nodes and m arcs. Verifying 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. 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 minimum mean cycle-canceling and primal network simplex algorithms. By guiding the resolution using a particular expansion scheme, we are able to recuperate theoretical results from the minimum mean cycle-canceling algorithm. As such, we obtain a strongly polynomial Contraction-Expansion algorithm which runs in $O(m^3n^2)$ time. The algorithmic choices stick to very practical observations of the minimum mean cycle-canceling algorithm's resolution behavior yet are not exploited by latter, namely that of phases and jumps on the optimality parameter. The resolution time is ultimately significantly reduced, even more so as the size of the instance increases.

Key Words: 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 tiny fraction of iterations induce nondegenerate 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 resolution of network flow problems also known as 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 one such alternative. Introduced by Goldberg and Tarjan (1989), this algorithm copes with degeneracy at the expense of a more involved pricing problem able to only identify improving directions with 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 roads bring another improvement due to Gauthier et al. (2015) which combines phases with Cancel-and-Tighten (CT) presented along side MMCC in the initial 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. Furthermore, we adopt the definitions and nomenclature of Ahuja et al. (1993) in which the reader may find several network flow properties.

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 resolution process is incidentally 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.

2 Network problem

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

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

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., arcs sharing the same head and tail.

When right-hand side $\mathbf{b} := [b_i]_{i \in N}$ is the null vector, the formulation (1) is called a *circulation* problem. The circulation problem is a cornerstone of network flow problems. Case in point, it is brought to light within the minimum mean cycle-canceling algorithm where it justifies the validity of the latter. Indeed, its core component, namely the so-called residual network is a circulation problem for which upper bounds are eventually neglected.

For the sake of formality, let us recall the path and cycle definitions used by Ahuja et al. (1993). A *path* in a directed graph $G = (N, A)$ is a sequence without repetition of nodes and arcs $i_1 - a_1 - i_2 - a_2 - \dots - i_{r-1} -$

$a_{r-1} - i_r$, satisfying the property that for all $i_k, 1 \leq k \leq r$, either $a_k = (i_k, i_{k+1}) \in A$ or $a_k = (i_{k+1}, i_k) \in A$. The sequence is typically given using nodes only. A *directed path* is an *oriented* version of a path in the sense that for any two consecutive nodes i_k and i_{k+1} on the path, the arc $(i_k, i_{k+1}) \in A$. A *cycle* is a path together with the arc (i_r, i_1) or (i_1, i_r) whereas a *directed cycle* is a directed path together with the arc (i_r, i_1) . The *cost* of a path or a cycle is the total unit cost of traveling on the forward arcs minus that on the backward arcs. Note 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 $\bar{c}_{ij} := c_{ij} - \pi_i + \pi_j$. Although Ahuja et al. (1993) limit the following fundamental property to directed cycles, the same telescopic summation argument proves to be enough regardless of the directed nature.

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

In the following subsections, we define the traditional residual network which exposes the improving disposition of the current solution $\mathbf{x}^0 := [x_{ij}^0]_{(i,j) \in A}$. An optimality certificate is then provided which indeed amounts to the statement of *cycle-canceling* algorithms. We finally move on to a *contraction* gymnastic which induces a so-called contracted network.

2.1 Residual network

The residual network is a marginal construction of the flow that may traverse the network aside the current flow \mathbf{x}^0 . The principle is simple, the combination of \mathbf{x}^0 along with the optimal marginal flow computed on the residual network is optimal for the original formulation. As eloquently resumed in Figure 1, each arc $(i, j) \in A$ can be replaced by two arcs representing upwards and downwards possible flow variations:

- arc (i, j) with cost $d_{ij} := c_{ij}$ and residual flow $0 \leq y_{ij} \leq r_{ij}^0 := u_{ij} - x_{ij}^0$;
- arc (j, i) with cost $d_{ji} := -c_{ij}$ and residual flow $0 \leq y_{ji} \leq r_{ji}^0 := x_{ij}^0 - \ell_{ij}$.

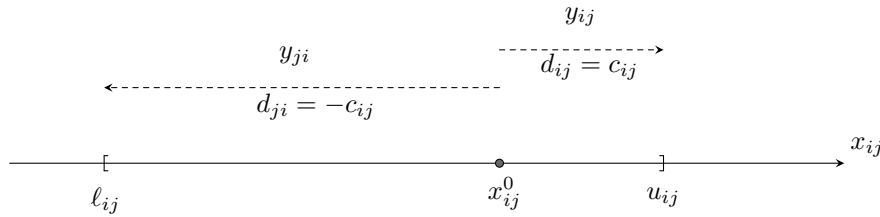


Figure 1: A change of variables

Denoted $G(\mathbf{x}^0) := (N, A(\mathbf{x}^0))$, the residual network with respect to \mathbf{x}^0 corresponds to the change of variables $y_{ij} - y_{ji} := x_{ij} - x_{ij}^0, \forall (i, j) \in A$. 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 positive residual capacities are of interest, i.e.,

$$A(\mathbf{x}^0) := \{(i, j) \in A' \mid r_{ij}^0 > 0\}. \quad (2)$$

An equivalent formulation of (1) states as

$$\begin{aligned} z^* &= z^0 + \min \sum_{(i,j) \in A(\mathbf{x}^0)} d_{ij} y_{ij} \\ \text{s.t.} \quad &\sum_{j:(i,j) \in A(\mathbf{x}^0)} y_{ij} - \sum_{j:(j,i) \in A(\mathbf{x}^0)} y_{ji} = 0, \quad [\pi_i], \quad \forall i \in N, \\ &0 \leq y_{ij} \leq r_{ij}^0, \quad \forall (i, j) \in A(\mathbf{x}^0), \end{aligned} \quad (3)$$

where $z^0 := \mathbf{c}^\top \mathbf{x}^0$ is the objective function evaluation of solution \mathbf{x}^0 . Furthermore, observe that traveling in both directions would be counterproductive and can be simplified to the net flow in a single direction. This means that the marginal flow must be such that $y_{ij} y_{ji} = 0, \forall (i, j) \in A$.

Take a look at Figure 2 which exhibits the construction of the residual network $G(\mathbf{x}^0)$. Figure 2a contains arc flow variables $x_{ij}^0, (i, j) \in A$ at different values. Each of these can be attributed a status depending on its actual value: *lower* when $x_{ij}^0 = \ell_{ij}$, *upper* when $x_{ij}^0 = u_{ij}$, or *free* when $\ell_{ij} < x_{ij}^0 < u_{ij}$. When a variable is free, the flow can be carried in either direction thus meaning the presence of two residual arcs. However, when a variable is lower (resp. upper), this induces only one arc oriented in the forward (resp. backward) direction. Handled in a very similar fashion, arcs at their lower or upper bound are also said to be *restricted*, a generic term which simplifies the text when the differentiation makes no difference. Let the disjoint sets $L(\mathbf{x}^0) \cup U(\mathbf{x}^0) \cup F(\mathbf{x}^0) = A$ respectively echo this status partition of the original set of arcs at \mathbf{x}^0 . In order to simplify the presentation, this partition is denoted as $\{L^k, U^k, F^k\}$, where $k \geq 0$ refers to the solution \mathbf{x}^k .

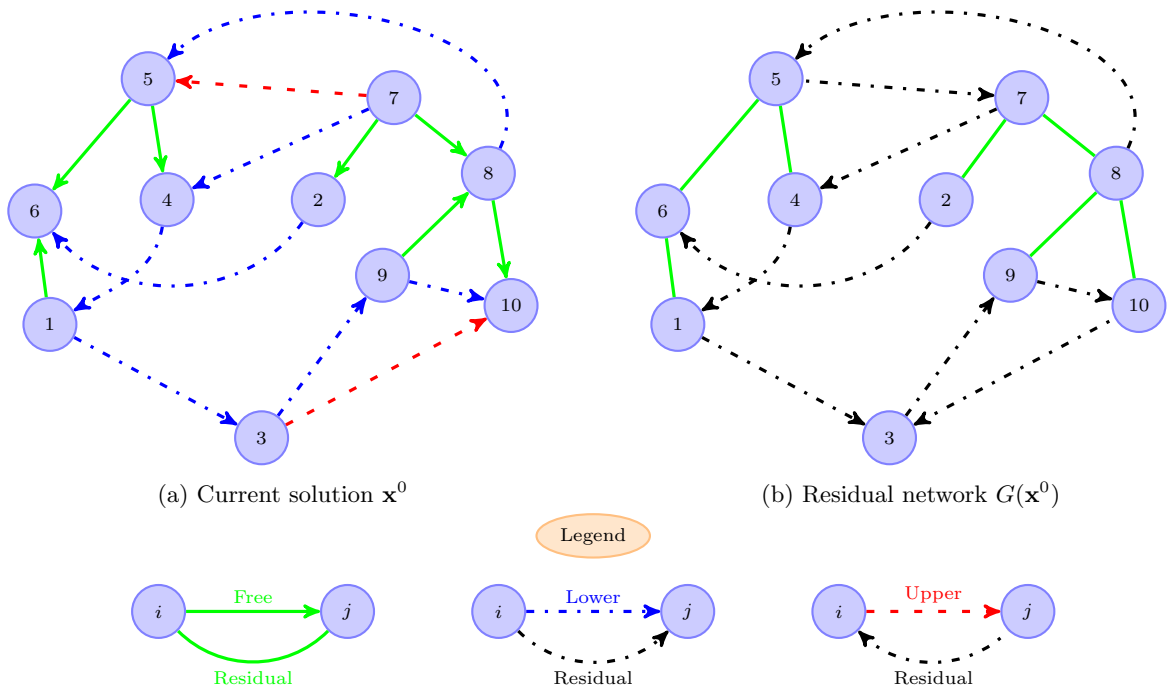


Figure 2: Residual network construction

2.2 Optimality conditions

It can be shown 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 3.8). By iteratively canceling a negative cycle and updating the residual network accordingly, one obtains a generic cycle-canceling algorithm which terminates when there remains no negative cycle (Klein 1967).

For the record, 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 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 $\rho := \min_{(i,j) \in W} r_{ij}^0$. A directed cycle is *anceled* when the step size is equal to ρ such that at least one of the residual capacity $r_{ij}^0, (i, j) \in W$ is saturated.

Since negative cycles are canceled sequentially, a cycle-canceling algorithm is basically defined by its oracle. 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 average cost. Technical

aspects of this algorithm are presented in Section 4.2 where we build upon these results to show the runtime complexity of the proposed 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

The definition of a *cycle free* solution can be read in Ahuja et al. (1993) as a solution \mathbf{x}^0 which contains no cycle of free arcs. In linear programming terminology, such and only such solutions are basic. By extension, a linearly independent set of arcs cannot form a cycle. Also recall that a *tree* T is a connected graph that contains no cycle. With this in mind, 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.

Consider the following construction. By definition, each of these trees encapsulates a nonoverlapping set of nodes connected using a subset of arcs in F^0 . Let each tree be rooted by an arbitrary node $\mathcal{R}(i) = i \in N$ such that any two nodes $i \neq j$ in N belonging to the same tree must have the same root node $\mathcal{R}(i) = \mathcal{R}(j)$. As depicted in Figure 3a, root nodes 1, 2 and 3 have been chosen such that $\mathcal{R}(1) = \mathcal{R}(4) = \mathcal{R}(5) = \mathcal{R}(6) = 1$. Let $G(F^0, \mathbf{x}^0)$ be the residual network on \mathbf{x}^0 where the *tree-layer* defined with respect to the set of free arcs F^0 is superposed. Take notice that the tree-layer appears only as a visual aid such that for all intents and purposes, $G(F^0, \mathbf{x}^0) \equiv G(\mathbf{x}^0)$. 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 . In performing the *contraction* of every tree to its root node, null variables y_{ij} corresponding to the respective $x_{ij}^0, (i, j) \in \{L^0, U^0\}$ are also caught in the crossfire. Each arc (i, j) is rerouted directly to the root nodes associated with its tail and head, that is, $\mathcal{R}(i)$ and $\mathcal{R}(j)$ respectively. Contrary to the initial assumption, this contraction gymnastic 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 costs computation have 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 concerns only efficiency matters.

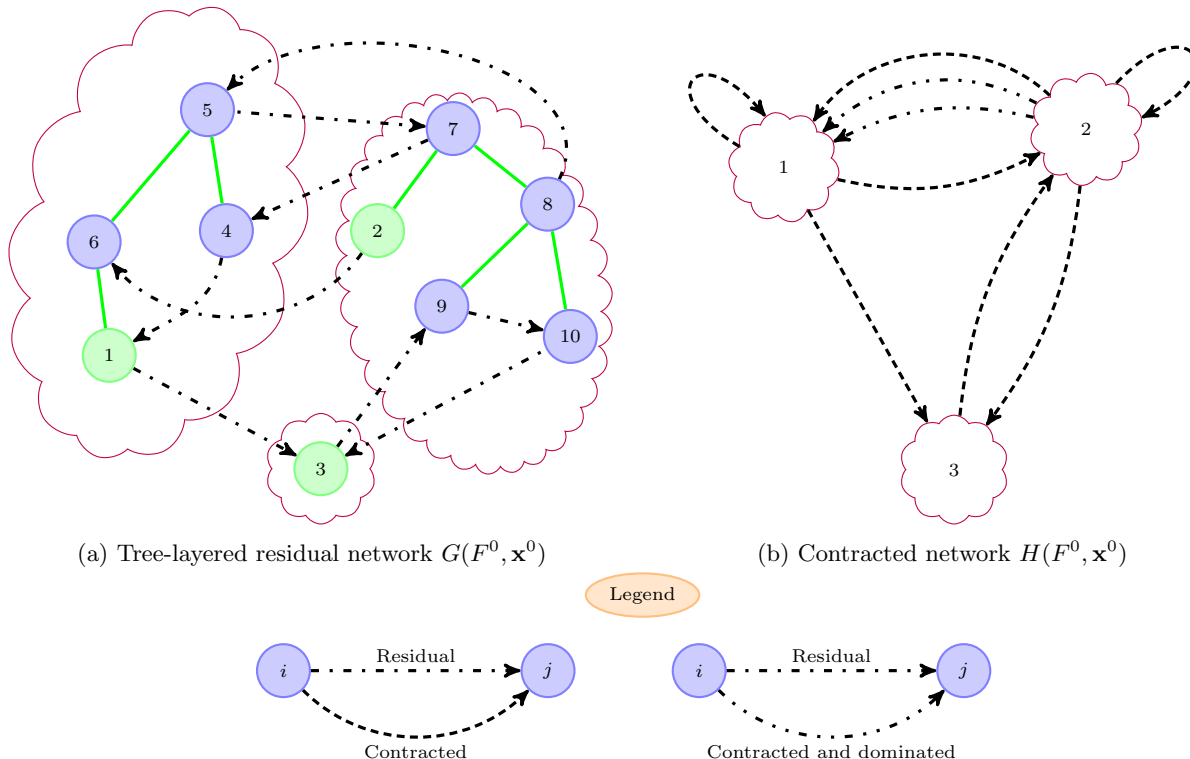


Figure 3: Contracted network based on the set of free arcs

Let us formalize the construction of the contracted network using a general tree-layer definition. Indeed, 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* in the contracted network. These sets are formed by

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

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

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

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

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

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

Furthermore, the arcs of the contracted network could have also been defined with respect to said root nodes, say with the set $A_P(\mathbf{x}^0) := \{(\mathcal{R}(i), \mathcal{R}(j)) \mid (i,j) \in V_P(\mathbf{x}^0)\}$. Moreover, it is possible to maintain a bijection between the sets $V_P(\mathbf{x}^0)$ and $A_P(\mathbf{x}^0)$ which might be more perceivable when omitting dominance.

We present two alternative contraction examples which morally 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 sent to the visible set $V_P(\mathbf{x}^0)$. We refer to this

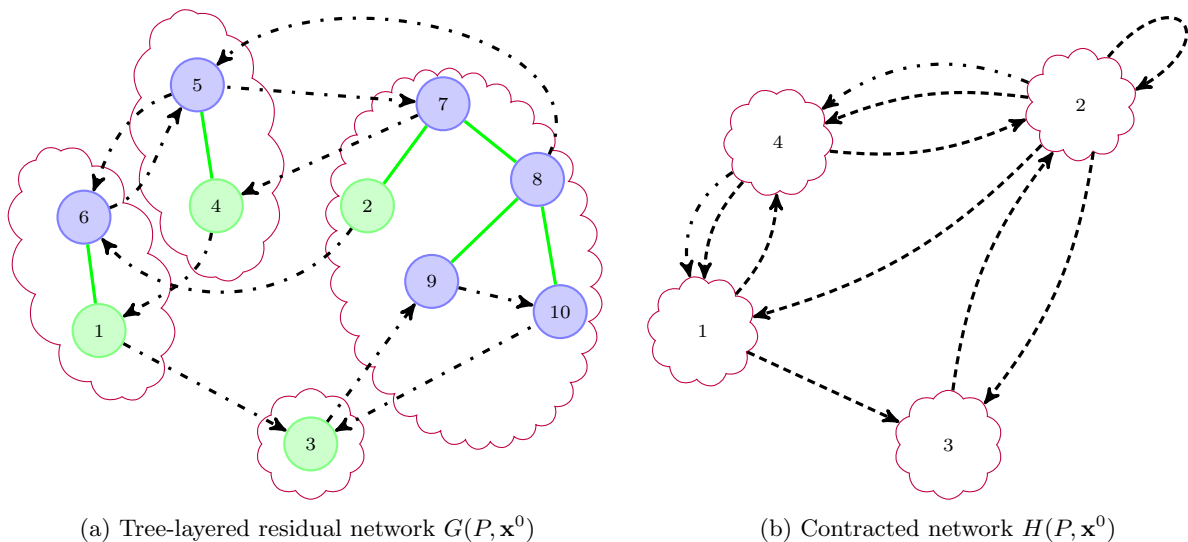


Figure 4: Contracted network with coerced degeneracy on the free arc (5,6)

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.

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 $P = F^0 \cup \{(7, 4), (3, 9)\}$ consists of the union of the free arcs along with two additional independent arcs at their bound. Without loss of generality, assume these two arcs are basic degenerate in the 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 and where each nonbasic arc becomes a loop, indeed, a directed cycle on $H(B^0, \mathbf{x}^0)$.

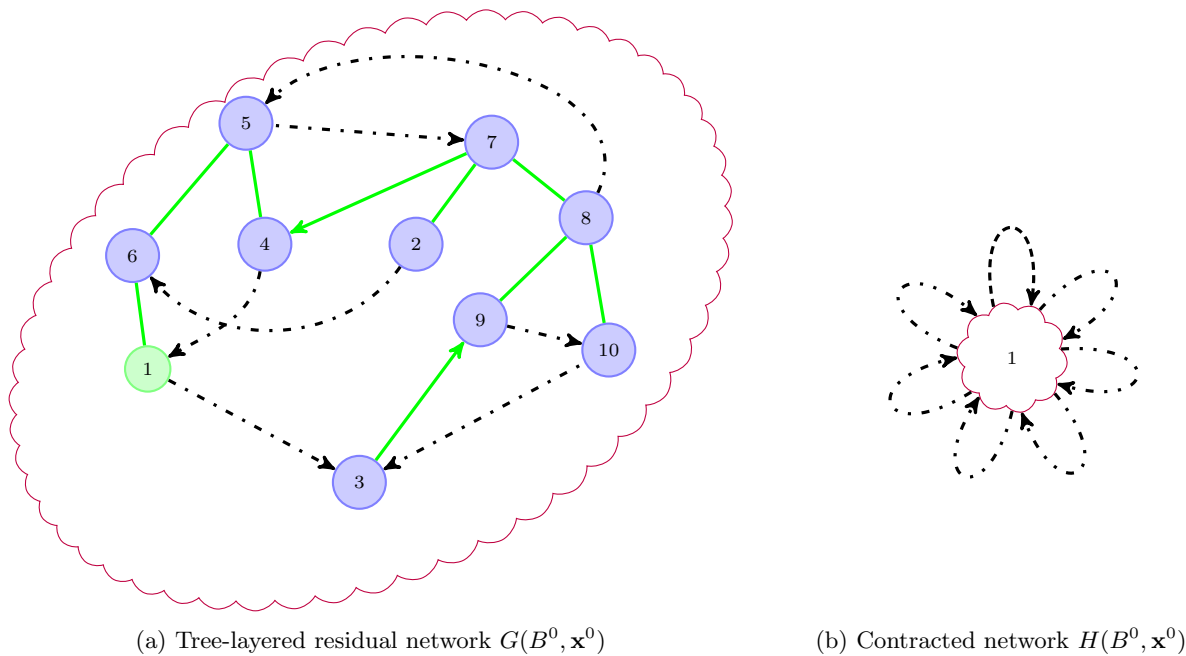


Figure 5: Contracted network with a set of basic arcs (primal network simplex algorithm)

When all is said and done, only two cases are possible for an arc contained in the selected set P : it is either free or it is not. Bare with us, 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 verify.

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 nonbasic nature of the current solution \mathbf{x}^0 . 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 costs 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

When defined with respect to the set F^0 which in turn trivially verifies 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 detected in G . 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 .

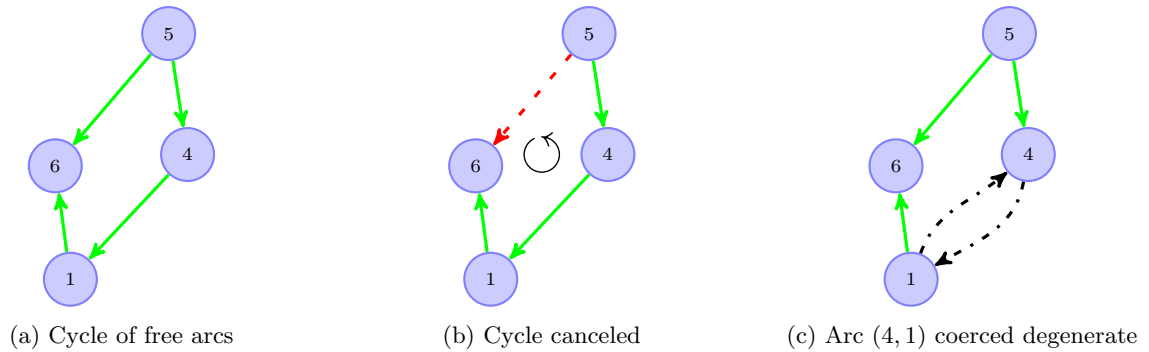


Figure 6: Working with a nonbasic feasible solution on network G

Once one lets $f := |F^0| \leq m$, Proposition 2 asserts that it is easy to maintain a working tree-layer regardless of whether \mathbf{x}^0 is basic or not.

Proposition 2 *A linear independent set P can be derived from F^0 by either removing any cycle of free arcs using at most f cycle cancellations or applying a trivial coerce degeneracy scheme 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-basics 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 $\mathcal{P}(i), i \in N$ be 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) .

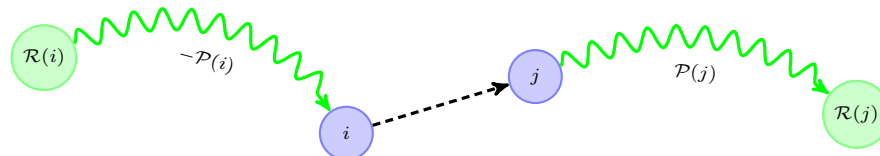


Figure 7: Rooted path

Several remarks about the rooted path are noteworthy. The two root nodes can eventually 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 null length 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. Given the bijection between $V_P(\mathbf{x}^0)$ and $A_P(\mathbf{x}^0)$, the same description holds for any contracted arc $(\mathcal{R}(i), \mathcal{R}(j)) \in A_P(\mathbf{x}^0)$.

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 part of the bijection are used to produce contracted cycles, the uniqueness of rooted paths guarantees 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. \square

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

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

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., $\mathcal{R}_s := \mathcal{R}(j_s) = \mathcal{R}(i_{s+1}), \forall s \in \{1, \dots, |W_H|\}$, 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 combination of rooted paths. Although, while the rooted paths simplify the work of extracting the extended cycle via concatenation, we underscore that the latter eventually requires a quick fix whose goal is to provide an elementary path between j_s and i_{s+1} .

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 $(i_{s+1}, j_{s+1}), s \in \{1, \dots, |W_H|\}$. Notice that both paths $\mathcal{P}(j_s)$ and $-\mathcal{P}(i_{s+1})$ have at least one node in common which is incidentally 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 $j_s \rightsquigarrow u_s$ and then $u_s \rightsquigarrow \mathcal{R}_s$ while $-\mathcal{P}(i_{s+1})$ travels from $\mathcal{R}_s \rightsquigarrow u_s$ followed by $u_s \rightsquigarrow i_{s+1}$. Since all nodes from u_s to \mathcal{R}_s 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

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

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

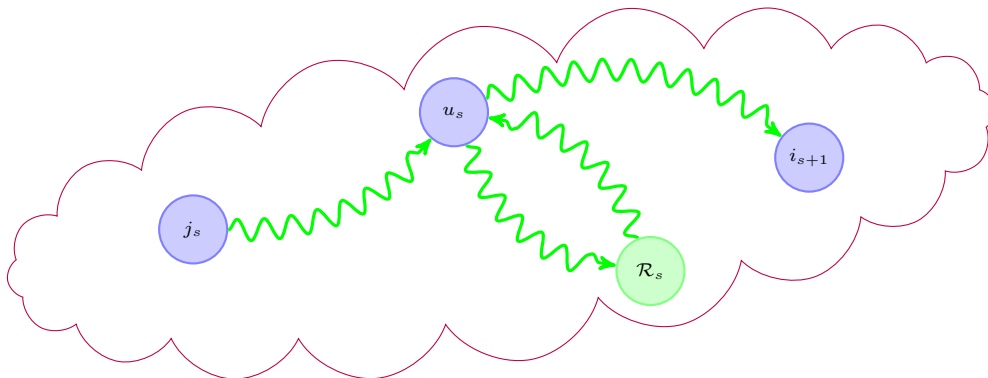


Figure 8: Elementary tree path detection

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. While there are probably many ways to ensure this, one straightforward way is to manipulate the dual variables such that Proposition 4 is true. Recall that 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 $\bar{c}_{ij} := c_{ij} - \pi_i + \pi_j$. 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.

Proposition 4 *The reduced cost of the contracted cycle W_H is equal to the cost of its extended counterpart $W_{H:G}$.*

Proof. For $i \in N$, let π_i be the cost of the path $\mathcal{P}(i)$, that is, from i to $\mathcal{R}(i)$ in the tree-layered residual network. All root nodes are consequently assigned a null dual variable value and moreover the reduced cost of any arc in the forest is zero, i.e., $\bar{d}_{ij} = 0, \forall (i, j) \in H_P(\mathbf{x}^0)$, indeed those arcs that are hidden in the contracted network. 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, we have the requested result. \square

3.4 Contraction algorithm

As all relevant parameters have been established, we are ready to present a contraction algorithm. Alike the pricing problem of the minimum mean cycle-canceling algorithm, residual capacities are omitted (not that we would have a meaningful way to attribute these values to contracted arcs anyway) and we aim to detect the presence of a negative cycle using the contracted network $H(P, \mathbf{x}^0) = (N_P(\mathbf{x}^0), V_P(\mathbf{x}^0))$ as well as a convexity constraint. Proposition 5 states the kind of cycles that are identified using the following pricing problem:

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

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

Proposition 5 *The pricing problem (9) finds a minimum average cost directed cycle on $H(P, \mathbf{x}^0)$, averaged over the number of arcs it contains.*

Proof. Charnes and Cooper (1962) show that linear fractional functionals can be equivalently solved using linear programming under mild conditions, namely a nonempty and bounded feasible region such that the denominator is either always positive or always negative. Assume the following linear fractional functional program $\{\min \mathbf{d}^\top \mathbf{x} / \mathbf{1}^\top \mathbf{x} \mid \mathbf{A} \mathbf{x} = \mathbf{0}, \mathbf{x} \geq \mathbf{0}, \mathbf{1}^\top \mathbf{x} > 0\}$ satisfies all conditions such that an equivalent linear program writes as $\{\min \mathbf{d}^\top \mathbf{y} \mid \mathbf{A} \mathbf{y} = \mathbf{0}, \mathbf{1}^\top \mathbf{y} = 1, \mathbf{y} \geq \mathbf{0}\}$ where the change of variables $\mathbf{y} := \mathbf{x} / \mathbf{1}^\top \mathbf{x}$ is performed. Since the convexity constraint $\mathbf{1}^\top \mathbf{y} = 1$ implies $\mathbf{1}^\top \mathbf{x} > 0$ the transformation holds. Flow conservation implies that the solution vector \mathbf{y}^* sports only two values: 0 or $1/w$, where $w = |W_H|$ when solving (9). Assuming $\mathbf{x} = w \mathbf{y}$, the change of variables is obviously verified since $\mathbf{1}^\top \mathbf{x} = w$. Finally, the pricing problem (9) optimizes over variable coefficients $\bar{\mathbf{d}}^\top = [d_{ij} - \pi_i + \pi_j]_{(i,j) \in V_P(\mathbf{x}^0)}$. The straightforward telescopic sum idea used to show the cost and reduced cost of a cycle are the same applies in the same fashion. \square

Recall that by using the arc cost transfer policy described in Section 3.3, the dual variables associated with the root nodes are null. Solving (9) not only provides an optimal cycle W_H^0 of minimum average reduced cost μ_H^0 on the contracted network but also a new set of dual variable values for these root nodes. The following is a direct adaptation of Gauthier et al. (2015, Proposition 3):

Proposition 6 *When solving the pricing problem (9) at iteration $k \geq 0$, there exists some dual variable values $\pi_\ell^k, \ell \in N_P(\mathbf{x}^k)$ such that $\bar{d}_{ij} - \pi_{\mathcal{R}(i)}^k + \pi_{\mathcal{R}(j)}^k \geq \mu_H^k, \forall (i, j) \in V_P(\mathbf{x}^k)$. By complementary slackness conditions, the latter in fact holds at equality for all positive variables, i.e., $\forall (i, j) \in W_H^k$.*

If the residual network contains a negative cycle, then by Proposition 4, so does the pricing problem (9). By construction, the current solution $\mathbf{x}^k, k \geq 0$ is then optimal when $\mu_H^k \geq 0$, that is, when the contracted network does not contain any negative cycle. Otherwise, given the identified contracted cycle W_H^k , one computes the nonnegative step size

$$\rho^k := \min_{(i,j) \in W_{H:G}^k} r_{ij}^k \geq 0, \quad (10)$$

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

$$\begin{aligned} x_{ij}^{k+1} &:= \begin{cases} x_{ij}^k + \rho^k, & \forall (i, j) \in A \mid (i, j) \in W_{H:G}^k \\ x_{ij}^k - \rho^k, & \forall (i, j) \in A \mid (j, i) \in W_{H:G}^k \\ x_{ij}^k, & \text{otherwise} \end{cases} \\ z^{k+1} &:= z^k + \rho^k |W_H^k| \mu_H^k. \end{aligned} \quad (11)$$

Observe that when $\rho^k = 0$, the set P^{k+1} must be different from P^k otherwise a dead-end is encountered and the algorithm does not terminate. Degeneracy and cycling phenomena from the primal simplex come to mind. Indeed, unless rules for null 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 $P^k \subset A$;

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

Figure 9: Generic contraction algorithm for network flow problems

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

3.5 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 take place on the extended cycle $W_{H:C}$ is trivial since all arcs unaccounted for can be used in either directions. 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 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 node 1. The corresponding extended cycle is defined by $\{(2, 6), \{\}, \{(6, 5), (5, 4), (4, 7), (7, 2)\}\}$. Unfortunately, the arc $(4, 7)$ is not a residual arc and reduces the possible flow on the identified extended cycle to zero. In simplex terms, pivoting on arc variable x_{26} induces a degenerate pivot.

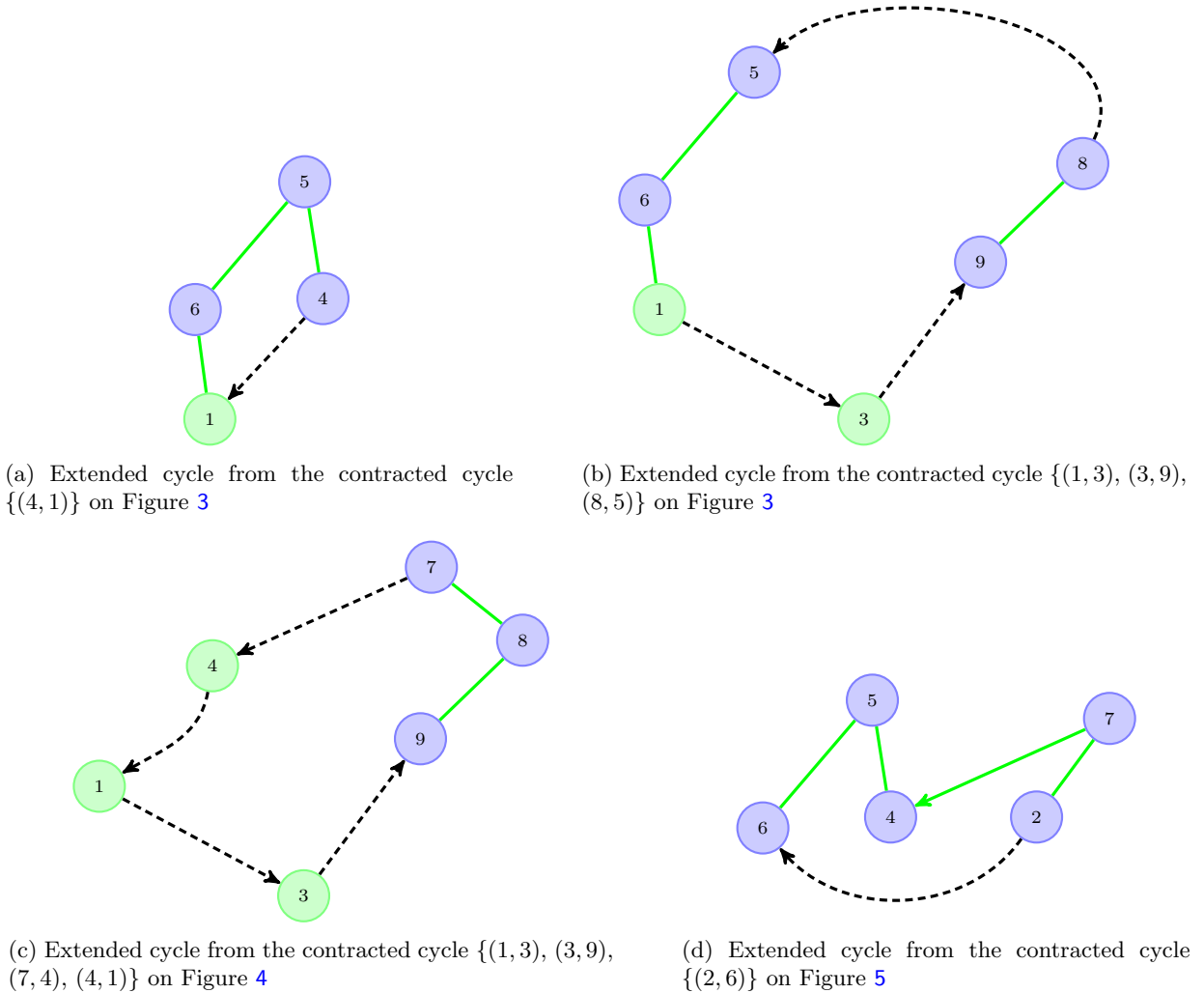


Figure 10: Extended cycle extractions

The contracted network $H(P, \mathbf{x}^0)$ is guaranteed to identify a directed extended cycle when $P \subseteq F^0$, that is, when the hidden arcs are free, any contracted cycle has a positive step size on $G(\mathbf{x}^0)$. On the other hand, when $P \not\subseteq F^0$ there is no such guarantee since the extended cycle could use an arc with a null residual capacity. Such is the proof of Proposition 7.

Proposition 7 *Given the set $P \subset A$ of linearly independent arcs, if*

- $P \subseteq F^0$, the extended cycle $W_{H:G}$ is directed on $G(\mathbf{x}^0)$ and the step size is positive. Therefore, the oracle (9) provides necessary and sufficient optimality conditions for (1).
- $P \not\subseteq F^0$, the extended cycle $W_{H:G}$ may be undirected on $G(\mathbf{x}^0)$ and the step size is greater than or equal to zero. In that case, the oracle (9) provides sufficient optimality conditions for (1).

Regardless of the choice of the set P , a noteworthy observation is that 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., $\mathcal{R}(i) \neq \mathcal{R}(j)$. The former class is said to be *compatible* with P .

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.*

Examples of compatible arcs can be seen in Figure 3a as arcs (4, 1) and (9, 10) or in Figure 3b as contracted arc loops on root nodes 1 and 2. Compatible arcs are sometimes favored because these loops are easy to identify on the contracted network thus forming directed cycles on their own.

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 associated cost is therefore the actual reduced cost of the arc. 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 costs $\bar{d}_{ij}, \forall (i, j) \in V_P(\mathbf{x}^0)$ as *rooted costs*. Also recall that the proposed arc cost transfer strategy is indeed used in the primal network simplex algorithm where a spanning tree and only one root node are used. The reader may want to reread the leading statement of this section in light of the compatibility concept and the primal network simplex reconciliation.

3.6 Extremal point solution space

It is straightforward to verify that both 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. As each contracted arc is compatible, applying cost dominance trivially results in a single contender for the pricing problem to identify. Recall that rooted costs are equal to reduced costs for all compatible arcs such that the entering variable identified by Dantzig's pivot rule would be the same.

Let \mathcal{C}_P be the set of all directed cycles obtainable as 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 size.

Proposition 8 *The cardinality of the extremal solution space $|\mathcal{C}_P|$ of the pricing problem (9) on the contracted network $H(P, \mathbf{x}^0)$ is $O\left(2^{(n-|P|+1)^2}\right)$.*

Proof. For the sake of simplicity, the argument is carried with a dominance scheme applied on the contracted 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 contracted 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 $|\mathcal{C}_P| \equiv O\left(2^{(n-|P|+1)^2}\right)$. \square

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 bigger one. Consider for instance Figures 10b and 10c where the cycle uses the nondominated 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 null step size. 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 (5b) has $|\mathcal{C}_B| = 1$ whereas many more cycles are present in the contracted network using all free variables (3b), a number that increases exponentially as the contracted network gets larger. We are thus expecting that the resolution of the pricing problem (9) be harder as $|P|$ gets smaller.

3.7 Root selection

Since the selection of root nodes is arbitrary, the attentive 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 9, 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 9 *Root selection has no influence on the compatible set, the extreme point solution set of the pricing problem (9), the average cost evaluation or the nature of the optimality certificate.*

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 contracted 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 contracted arcs is unaltered. Furthermore, by Proposition 4 the original cost on each cycle is maintained regardless of the root selection. As such, the average 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. \square

Furthermore, the proof of Proposition 4 makes the same assumption as in Ahuja et al. (Chapter 11.4 1993), that is, a root node dual variable value is arbitrarily fixed to 0 since any constant θ can be added to the vector of dual variables without modifying the reduced costs. More generally, each root node dual variable value can be fixed to a different constant without any impact on the outcome of the pricing problem (9).

Proposition 10 *The dual variable of each root node $\pi_i, i \in N_P(\mathbf{x}^0)$, can be initialized to an arbitrary but fixed value. The remaining dual variable values $\pi_i, i \in N \setminus N_P(\mathbf{x}^0)$ are then the cost of the path $\mathcal{P}(i)$ plus the root node dual variable value $\pi_{\mathcal{R}(i)}$.*

4 Behavioral study

Supported with numerical results from a minimum cost flow problem containing 1,025 nodes and 92,550 arcs referred to as Instance 1, Section 4.1 analyzes the behavior of a specific variant of the contraction algorithm opposite MMCC's established behavior. This behavioral 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 , 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

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

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

Proposition 11 *Let \mathbf{x}^k be a nonoptimal solution. Given the set $P \subseteq F^k$, the following ordering of the minimum mean costs always holds:*

$$\mu_H^k \leq \mu_G^k \leq \mu_{H:G}^k. \quad (13)$$

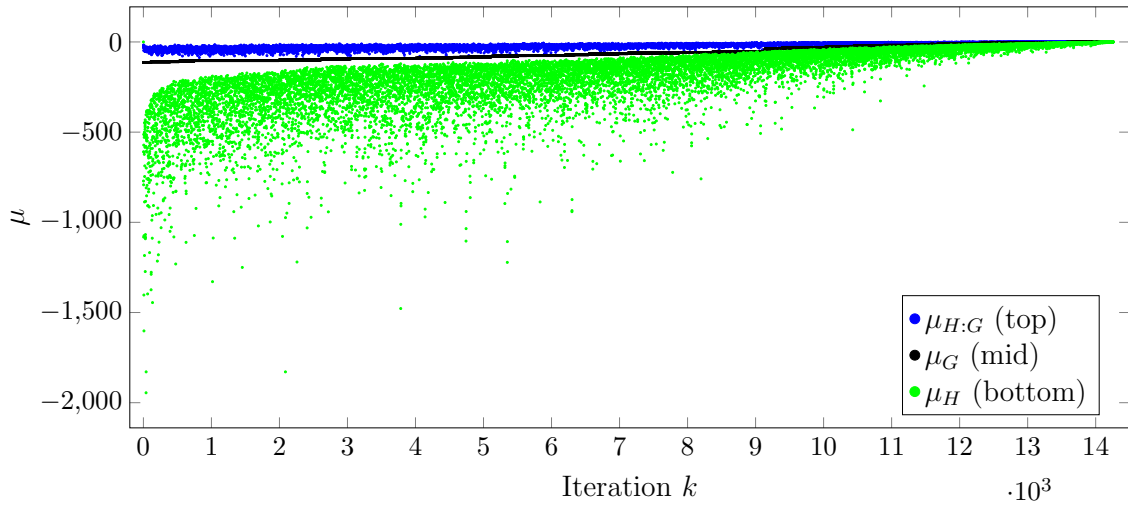
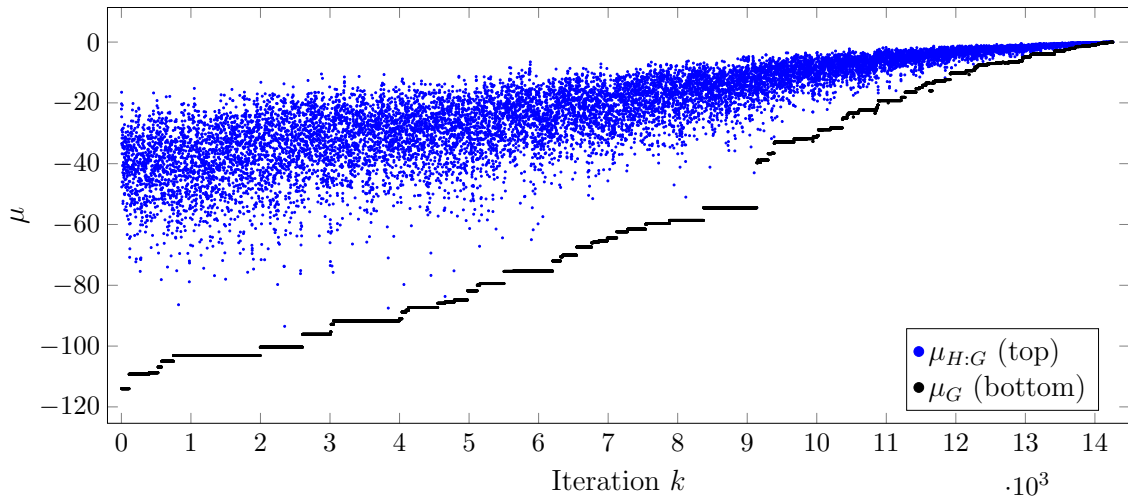
Proof. The cycle $W_{H:G}^k$ is visible as is on the residual network $G(\mathbf{x}^k)$ which means $\mu_G^k \leq \mu_{H:G}^k$ trivially holds by the nature of minimizing the mean cost in $G(\mathbf{x}^k)$. Furthermore, although contracted, any cycle W_G^k that also appears in $H(P, \mathbf{x}^k)$ is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

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 can be interpreted in a straightforward manner: The contracted cycle appears to provide a lower minimum mean cost μ_H^k yet once the hidden arcs are accounted for, this illusion disappears and indeed shows that μ_G^k is overestimated by $\mu_{H:G}^k$. 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 notion of estimation might be better understood by observing the evolution of the various μ during the resolution 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.

As expected, the plot verifies the aforementioned ordering of Proposition 11. 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 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 and applies the minimum mean cycle at each iteration yielding an algorithm which features a nondecreasing 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 10,000th 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

Figure 11: Comparison of μ_H^k , μ_G^k and $\mu_{H:G}^k$ Figure 12: Zoom on comparison of μ_G^k and $\mu_{H:G}^k$

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.

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 average cost to that of the minimum mean cycle cost is an enlightening exercise. Indeed, the chaotic 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 properties are 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 bounded number of operations measured solely by the size of its input (see Cormen et al. (2009) for further inquiries). For network flow

problems, this bound 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 the *minimum mean cost directed cycle* in $O(mn)$ time. Second, an optimal solution is reached by canceling at most $O(m^2n \log n)$ such negative cycles. The first part can be seen as the inner loop while the second 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 $O(m^2n)$. 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.

Type 2 (negative) cycles. Even though the nondecreasing 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 nonnegative 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

$$\mu_G^l \geq (1 - 1/|W^l|) \mu_G^h \geq (1 - 1/n) \mu_G^h \quad (14)$$

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

At the end of the day, since each phase 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), respectively Theorem 2 using the minimal increasing factor $(1 - 1/n)$ and Theorem 3 rather exploiting the stronger factor $(1 - 1/|W^l|)$. Truth be told, 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 resolution process cares not about these cycle Types. The enlightenment comes from the inclusion of Cancel-and-Tighten (CT) 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 the minimum mean cycle 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 chaotic behavior of μ_{CT}^k can clearly be observed throughout the resolution, 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 4,900th 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 properties despite the usage of Type 1 cycles going against the nondecreasing 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).

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 12, 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 $\hat{\mu}_{CT}^h$ are shown to mimic the behavior of the true value μ_G^h . 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_G^h instead.

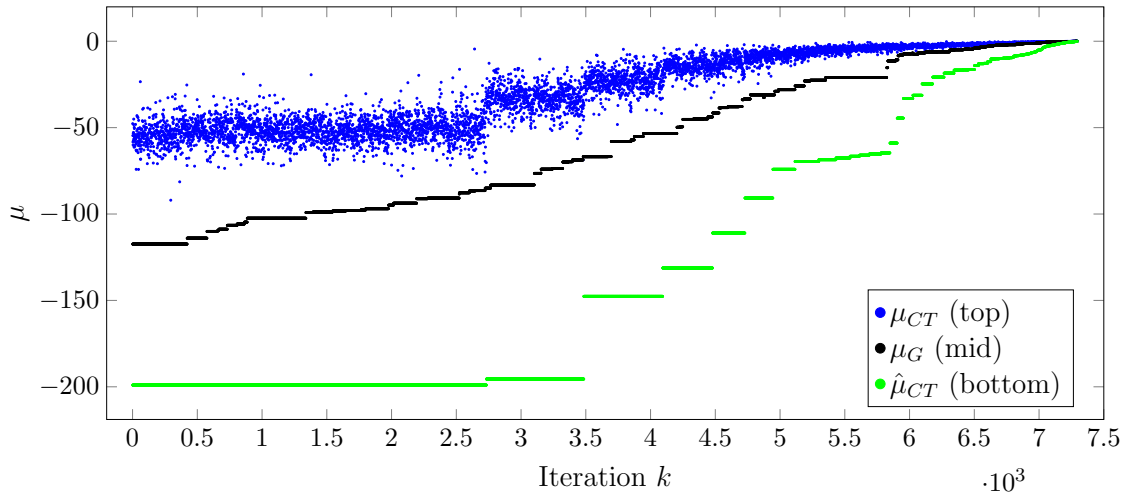


Figure 13: Comparison of μ_{CT}^k , μ_G^k and $\hat{\mu}_{CT}^k$

Proposition 12 Assuming $\hat{\mu}^h$ is a lower bound for μ_G^h at the beginning of phase h .

- Arc fixing for arc $(i, j) \in A$ occurs or has already occurred if $|\bar{c}_{ij}^h| \geq -2n\hat{\mu}^h$.
- 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.

Optimality parameter $\hat{\mu}^h$. It is at this point important to create separation between μ_G^k at iteration k and strongly polynomial properties. In MMCC, μ_G^k is coined as the *optimality parameter*: 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 Cancel-and-Tighten strategy in the above example. Indeed, whether one looks at μ_H^k or μ_{CT}^k or any other cycle-canceling scheme, the minimum mean cycle μ_G^k can always be fetched as secondary information (recall that $\mu_H^k \leq \mu_G^k \leq \mu_{H:G}^k$ and $\hat{\mu}_{CT}^k \leq \mu_G^k \leq \mu_{CT}^k$). If strongly polynomial properties are to be established on oscillating values from one iteration to the next, it appears mandatory to divert the analysis to a lower bound $\hat{\mu}^h$ over the phases instead. In other words, while the portion above μ_G^k (blue) may be unpredictable, it is the portion below μ_G^k (green) that should be well-behaved. We are ready to propose an adaptation of the contraction algorithm.

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_G^h 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 13.

Definition 4 Let \mathbf{x}^h be a nonoptimal 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 $\hat{\mu}^t \leq \mu_G^t \leq \mu^t$) produces the measurable jump

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

Proposition 13 In MMCC, a Type 3 cycle is identified within at most m cancellations.

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

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 12, that is,

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

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.

Let \mathbf{x}^0 be a nonoptimal solution at iteration $k = 0$. Note that \mathbf{x}^0 is also a nonoptimal phase solution ensuring a lower bound $\hat{\mu}^0 \leq \mu_G^0$. Let the set $P^0 = F^0$. By Proposition 7, the extended cycle $W_{H:G}^0$ is directed on the residual network $G(\mathbf{x}^0)$ and the step size is 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 nondecreasing 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 reapply the full contraction, it is worthwhile to eliminate any cycle of free arcs from the phase solution to maximize the contraction benefit.

```

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

```

Figure 14: Contraction-Expansion algorithm

5.3 Complexity analysis

The complexity analysis of the Contraction-Expansion algorithm revolves around bringing the resolution 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 resolution of Instance 1. 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 14 *The Contraction-Expansion algorithm completes a phase in $O(m^2n)$ time.*

Proof. The proof is threefold with respect to the expansion scheme portion of the algorithm. We prove that 1) the nondecreasing 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 average reduced cost cycle on the contracted network requires $O(mn)$ time.

Gauthier et al. (2015, Proposition 4) states 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 average reduced cost μ_H^k along with an optimal set of dual variables $\pi_\ell^k, \ell \in N_P(\mathbf{x}^k)$ on the root nodes (Proposition 6). From there, fix the root nodes dual variables to these new values and update the remaining dual variables accordingly (Proposition 10). The residual network $G(\mathbf{x}^k)$ hence satisfies $\bar{d}_{ij} - \pi_{\mathcal{R}(i)}^k + \pi_{\mathcal{R}(j)}^k \geq \mu_H^k, \forall (i, j) \in A(\mathbf{x}^k)$ and in particular $\bar{d}_{ij} = 0, \forall (i, j) \in H_P(\mathbf{x}^k)$. Upon canceling the

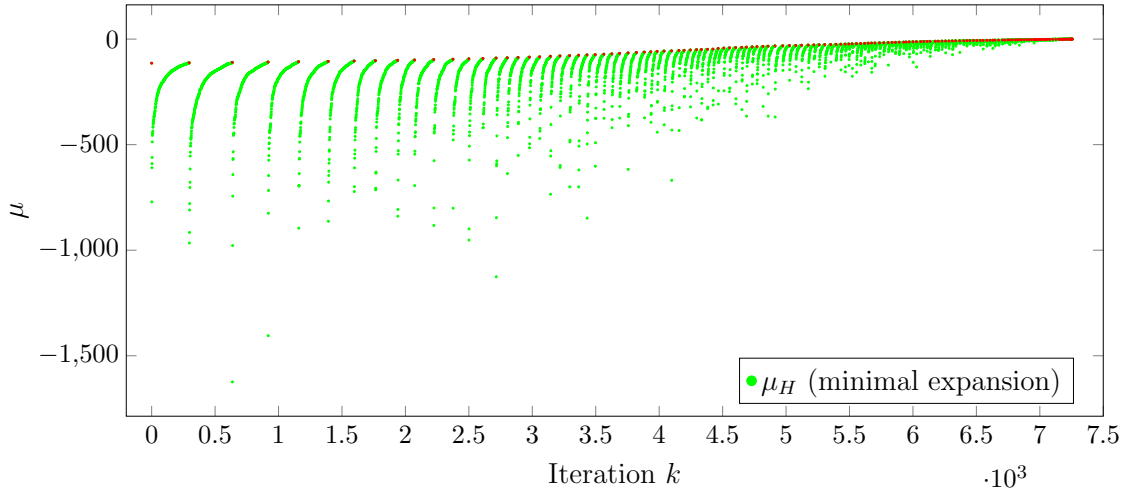


Figure 15: Evolution of μ_H^k with the Contraction-Expansion algorithm

expanded cycle $W_{H:G}^k$ and obtaining \mathbf{x}^{k+1} , every new residual arcs in $G(\mathbf{x}^{k+1})$ either has a reduced cost of 0 or $-\mu_H^k$, i.e., $\bar{d}_{ij} - \pi_{\mathcal{R}(i)}^k + \pi_{\mathcal{R}(j)}^k \geq \mu_H^k, \forall (i, j) \in A(\mathbf{x}^{k+1})$. 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_H^k . The average cost of the next contracted cycle is then at least μ_H^k .

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 $|F^0| \equiv O(n)$ 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 some restricted variables changed from one bound to another. Unfortunately, this kind of pathological phenomenon is what nightmarish instances are made of. 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 allows 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 13). To sum up, at most $n + m \equiv O(m)$ 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 $O(m^2n)$. \square

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

Proof. We show that at most $O(mn)$ phases can occur in accordance with Gauthier et al. (2015, Theorem 3). Since $\mu_H^k \leq \mu_G^k$ is indeed a lower bound for the true minimum mean cycle cost value by Proposition 11, this is also true for any phase solution. As soon as $\mu_H^k \geq \hat{\mu}^h (1 - 1/|W_{H:G}^k|)$, the phase h ends and the lower bound is increased to $\hat{\mu}^{h+1} := \mu_H^k$. By Proposition 12, arc fixing occurs on the lower bound value $\hat{\mu}^h$ as well such that the phase time complexities are valid. By Proposition 14, since each phase runs in $O(m^2n)$, the compound time is obtained. An initial valid lower bound for μ_G^0 can be trivially obtained with $\hat{\mu}^0 := -\max_{(i,j) \in A} |c_{ij}|$. \square

The proposed phase definition along with the Type 3 cycle not only satisfy theoretical properties of the strongly polynomial complexity of MMCC, they also express very practical observations. The hope is that not only the Type 3 cycle occurs much faster than 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_H^k 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

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

such that all arcs of the expanded cycle $W_{H,G}^k$ 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.

For anyone familiar with successful divide-and-conquer methods, the resolution speed typically benefits from the decomposition at a higher rate than the cost of the latter. While postponing the end of a phase with

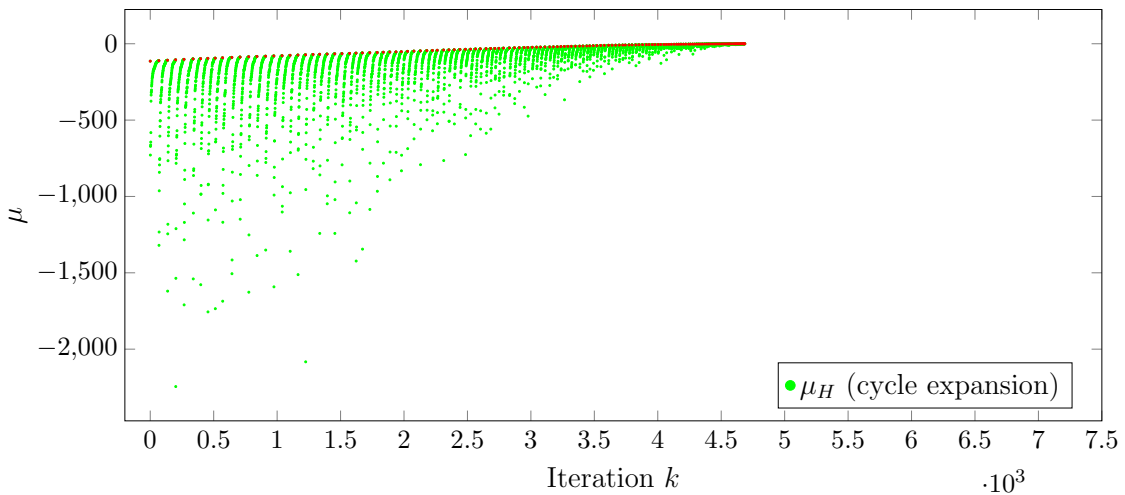


Figure 16: Evolution of μ_H^k with cycle expansion

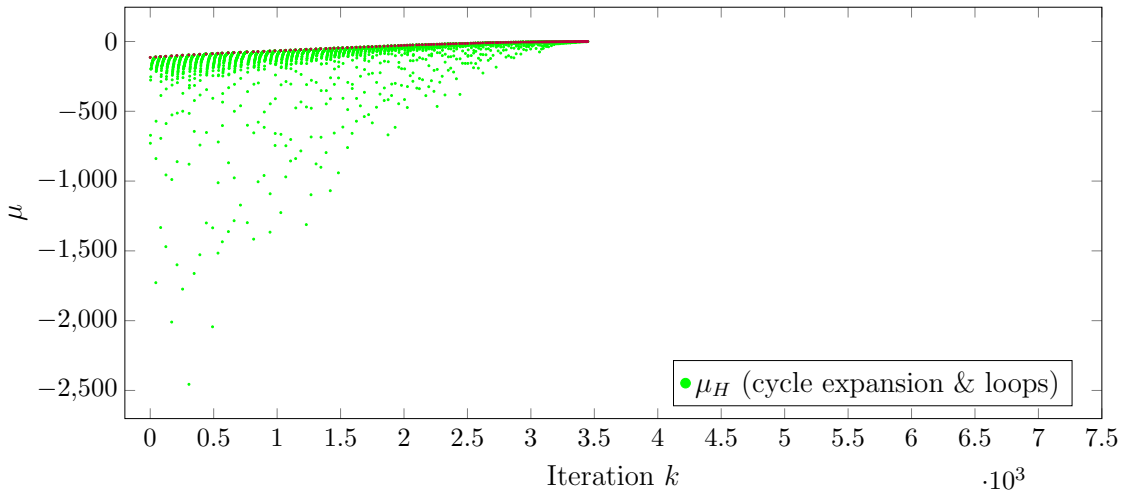


Figure 17: Evolution of μ_H^k with cycle expansion and cancellation of negative loops

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 reapplied. 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:

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

where $k \in h$ is understood as an iteration k within the phase h . While the last criteria is reminiscing of Theorem 2 which contents itself with the same jump every time thus obtaining the $O(mn \log n)$ phases, the second criteria also tracks a cycle length and compromises on the desired jump. The latter is in fact the criteria 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). For the record, these results are obtained using an Intel i7-4770K@3.50GHz with 16GB of RAM.

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 $\bar{\alpha}$ where $\alpha^k = |N_P(\mathbf{x}^k)|/n, \forall k \geq 0$.

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

Resolution	CPU (sec)	k / cycles	h	$\overline{ W }$	$\bar{\rho}$	$\bar{\alpha}$
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 **GRIDGRAPH** family and brings further testimonies to the previous claims. For each of the three categories of problems (**long**, **square**, **wide**) ranging from sizes 2^8 to 2^{17} , 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. Once again, the relative performance of CE increases with the instance size.

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

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

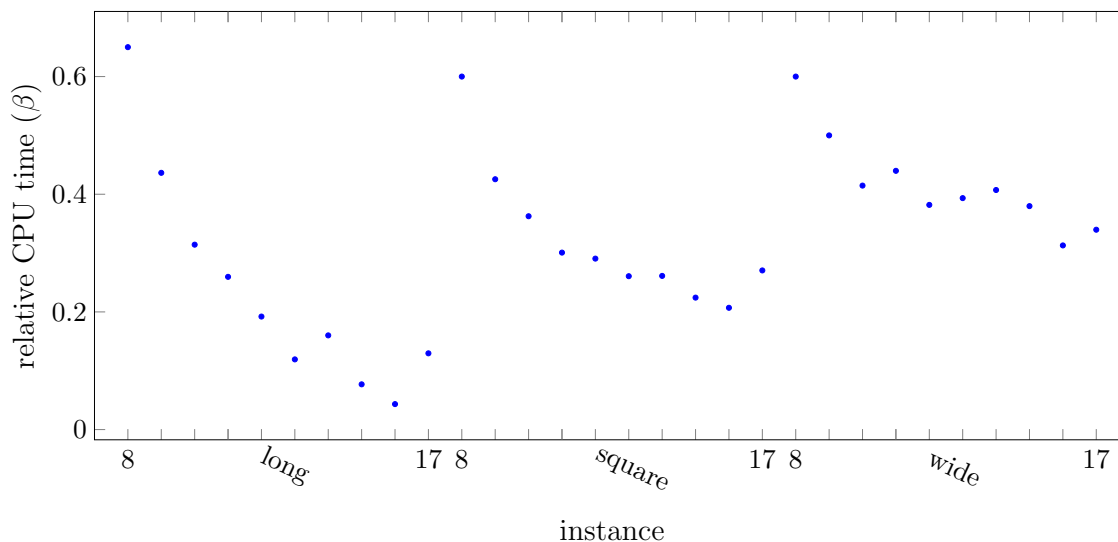


Figure 18: Relative CPU time for CE when solving the **GRIDGRAPH** family, version A

While we certainly did not cover the complete benchmark suite, this paper does not pretend to a competitive algorithm just yet. Case in point, even larger instances up to 2^{20} nodes are available but MMCC's resolution 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.

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 a more generic contraction algorithm, indeed extending to both ends of its spectrum.

A variety of special cases inducing 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 properties.

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.

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 the residual network is. 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

- R.K.A., T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, NJ, USA, 1993.
- A. Charnes and W.W. Cooper. Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9(3-4):181–186, 1962. doi:[10.1002/nav.3800090303](https://doi.org/10.1002/nav.3800090303).
- T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, USA and London, England, 3rd edition, 2009.
- DIMACS. Network Flows and Matching: First DIMACS Implementation Challenge, 1990–1991. URL <ftp://dimacs.rutgers.edu/pub/netflow>.
- J.B. Gauthier, J. Desrosiers, and M.E. Lübbecke. About the minimum mean cycle-canceling algorithm. *Discrete Applied Mathematics*, 196:115–134, 2015. doi:[10.1016/j.dam.2014.07.005](https://doi.org/10.1016/j.dam.2014.07.005). Advances in Combinatorial Optimization.
- A.V. Goldberg and R.E. Tarjan. Finding minimum-cost circulations by canceling negative cycles. *Journal of the ACM*, 36(4):873–886, 1989. doi:[10.1145/76359.76368](https://doi.org/10.1145/76359.76368).
- R.M. 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](https://doi.org/10.1016/0012-365X(78)90011-0).
- M. Klein. A primal method for minimal cost flows with applications to the assignment and transportation problems. *Management Science*, 14(3):205–220, November 1967.
- P. Kovács. Minimum-cost flow algorithms: an experimental evaluation. *Optimization Methods and Software*, 30(1):94–127, 2015. doi:[10.1080/10556788.2014.895828](https://doi.org/10.1080/10556788.2014.895828).
- R.E. Marsten, M.J. Saltzman, D.F. Shanno, G.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](https://doi.org/10.1287/ijoc.1.4.287).
- T. Radzik and A.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](https://doi.org/10.1007/BF01240734).