Mathematical Programming 59 (1993) 117-132
North-Holland



# A strongly polynomial algorithm for minimum convex separable quadratic cost flow problems on two-terminal series-parallel networks

Arie Tamir

*New York University, New York, USA, and Tel Aviv University, Tel Aviv, Israel*

Received 25 September 1989
Revised manuscript received 27 June 1991

We present strongly polynomial algorithms to find rational and integer flow vectors that minimize a convex separable quadratic cost function on two-terminal series-parallel graphs.

*Key words*: Quadratic cost flow problems, strongly polynomial algorithms, series-parallel graphs.

## 1. Introduction

This paper was motivated by the following model of location of undesirable or obnoxious facilities. Given a general network the objective is to select $p$ points on the network so that the sum of the distances between all pairs of points is maximized. It was shown in [6, 17] that this problem and some of its variants are NP-hard. However, when the network is a tree, we formulated the model as a minimum separable convex quadratic cost flow problem with integer flow variables [17]. As such the model was solvable in polynomial time by the algorithm presented in [15]. Nevertheless, the latter algorithm was not satisfactory for our purposes. Our goal was to obtain a combinatorial strongly polynomial algorithm for the above location model. The algorithm in [15], as well as the one in [14] which solves the real relaxation of the above cost flow problem, are both polynomial but not strongly polynomial.

In this paper we deal with series-parallel networks and present strongly polynomial procedures for solving the integer and the rational convex quadratic separable cost flow problems with one source and one sink. The integer flow problem induced by the above obnoxious location model is defined on the following special series-parallel network: Add a node to the node set of the underlying tree and connect it to all the tips (leaves) of the tree [17].

*Correspondence to*: Prof. Arie Tamir, Department of Statistics, Tel Aviv University, Ramat Aviv, 69978 Tel Aviv, Israel.

A directed (multi) graph $G=(V, E)$ is given by its node set $V=\{1, \ldots, n\}$, its arc set $E$ and two mappings $h, t: E \to V$ which assign to each arc $e \in E$ the head $h(e)$ and tail $t(e)$ of $e$, respectively. $h(e)$ is the successor of $t(e)$ and $t(e)$ is called the predecessor of $h(e)$. Two arcs $e$ and $e'$ in $E$ are called parallel if $h(e)=h(e')$ and $t(e)=t(e')$. Let $m=|E|$.

Each arc $e \in E$ is associated with a nonnegative flow capacity bound $u_e$. Also, let $x_e$, $e \in E$, indicate the flow on $e$. Given a positive real $q$ suppose that $q$ units of flow are to be transferred from a given source node, say node 1, to a given sink, say node $n$. Let $D$ be an $m \times m$ symmetric positive semi-definite matrix, and let $c$ be an $m$-component vector. The minimum convex quadratic cost flow problem on $G$ is to compute $f(q)$,

$$
\begin{aligned}
f(q) = \text{minimum} \quad & x'Dx + c'x \\
\text{subject to} \quad & (Ax)_i = \begin{cases} q, & i = 1, \\ -q, & i = n, \\ 0, & i \neq 1, n, \end{cases} \qquad (1.1) \\
& 0 \leq x_e \leq u_e, \quad e \in E,
\end{aligned}
$$

where $x=(x_e)$, $e \in E$, is the $m$-component flow vector and $A$ is the node arc incidence matrix of $G$. By the parametric version of the model we refer to the problem of computing and representing the function $f(q)$ for all $q \geq 0$ such that (1.1) is feasible. The discrete (integer) minimum quadratic cost flow problem is the problem obtained from (1.1) when we add the requirement that all flow variables $x_e$, $e \in E$, must be integers.

Throughout the paper we assume that all numerical input is rational. Also, when we refer to (1.1) for a given $q \geq 0$, we assume that it is feasible. Under these assumptions (1.1) has an optimal solution which is rational, and it can be found in polynomial time by an ellipsoid method, [11]. Thus, (1.1) is referred to as the rational problem. The repertoire of numerical operations used by the algorithm in [11] consists only of comparisons and the four elementary operations, $\{+, -, /, *\}$. The total number of operations performed by the procedure in [11] depends polynomially on the capacity bounds $\{u_e\}$, $e \in E$, the vector $c$ and the matrix $D$.

More recently it was demonstrated in [2, 4] that (1.1) can be solved in polynomial time where the total number of operations depends only on the matrix $D$. The results in [2, 4] generalize the seminal work of Tardos [18], who was the first to prove that the linear case, $D=0$, is solvable in strongly polynomial time.

For the linear case the discrete model coincides with the rational one when $q$ and $\{u_e\}$, $e \in E$, are all integer. This is not necessarily the case when $D$ is nonzero. In fact, the discrete version of (1.1) is NP-hard. The problem of finding a 0-1 solution to a single equation can be reduced to (1.1) by squaring the equation.

In this study we focus on the case when $D$ is diagonal, i.e., the objective in (1.1) is separable and convex. Motivated by the location model mentioned above, we restrict ourselves to series-parallel graphs.

The separable convex case for general graphs was considered by Minoux in [14, 15]. Using a scaling technique he showed in [15] that the discrete model can be solved in polynomial time, where the complexity bound depends only on the size of $G$, and the capacity bounds $\{u_e\}$, $e \in E$. A solution to the rational model is derived in [14]. The polynomial complexity bound there depends also on the matrix $D$.

To the best of our knowledge no strongly polynomial algorithm is known to solve either problem (1.1) or its discrete version even when $D$ is diagonal. In our context a strongly polynomial time implies a complexity bound which depends (polynomially) only on the size of $G$, i.e., on $n$ and $m$. The reader is referred to [8] for related open questions.

In the next section we use the proximity results in [5] to show how to obtain a discrete optimal solution from a rational optimal solution for a general graph. Section 3 is devoted to series-parallel graphs. We show that the real parametric function $f(q)$, $q \geq 0$, is convex piecewise quadratic with at most $2m$ breakpoints. We present an $O(mn + m \log m)$ algorithm to generate and represent this parametric function. Given the representation and a specific rational value of $q$, a rational optimal solution will be computed in $O(\log m)$ time. In Section 4 we briefly discuss several extensions and state some open problems.

## 2. Finding a discrete solution from a rational solution

In this section we use the proximity results in Granot and Skorin-Kapov [5]. Specifically we show that if we add to our repertoire of permissible operations the "floor" operation, $\lfloor \cdot \rfloor$, then an optimal solution to the discrete model of (1.1) with a diagonal $D$ can be obtained from any optimal rational solution to (1.1) in strongly polynomial time. ($\lfloor a \rfloor$ is the largest integer smaller than or equal to $a$. $\lceil a \rceil$ is the smallest integer larger than or equal to $a$.)

First we motivate our approach. As mentioned above, a solution to the discrete version of (1.1) can be obtained in time which is polynomial in the flow capacity bounds, (e.g., [15]). Therefore, to solve the discrete problem in strongly polynomial time it will suffice to exhibit a strongly polynomial transformation of the discrete version of (1.1) into an instance of the same model where the capacity bounds are polynomial functions of $m$, the number of arcs of $G$. We show now that such a transformation exists whenever a solution to the rational model (1.1) is available.

Let $x^*$ be an optimal solution to the rational model (1.1). It follows from [5], that there exists an optimal solution to the discrete model, say $z^*$, such that

$$|z_e^* - x_e^*| \leq m \quad \text{for all } e \in E. \tag{2.1}$$

By computing $\lfloor x_e^* \rfloor$ for all $e \in E$, we can obtain an integer flow vector $y^* = (y_e^*)$,

$e \in E$, satisfying

$$|z_e^* - y_e^*| \leq m \quad \text{for all } e \in E,$$

$$0 \leq y_e^* \leq u_e \quad \text{for all } e \in E,$$

and

$$(Ay^*)_i = \begin{cases} q, & i = 1, \\ -q, & i = n, \\ 0, & i \neq 1, n. \end{cases} \tag{2.2}$$

$y^*$ is any integer feasible solution to (1.1) with the additional constraints $\lfloor x_e^* \rfloor \leq y_e^* \leq \lceil x_e^* \rceil$, $e \in E$. $y^*$ can be computed in strongly polynomial time by using any of the strongly polynomial algorithms for the classical maximum flow - minimum cut problem.

Next we find an integer nonnegative flow $f^* = (f_e^*)$, $e \in E$, such that

$$y_e^* - m^2 \leq f_e^* \leq z_e^* \leq y_e^* + m,$$

$$(Af^*)_i = \begin{cases} F^{**}, & i = 1, \\ -F^{**}, & i = n, \\ 0, & i \neq 1, n, \end{cases} \tag{2.3}$$

with $q - m^2 \leq F^{**} \leq q$.

Consider the integer flow $y^*$ in (2.2). This flow constitutes a circulation in the graph $G^1$ obtained from $G$ by adding a directed arc from node $n$ to node 1 with a capacity bound of value $q$. We subtract a unit flow along at most $m^2$ simple cycles in $G^1$ to obtain a nonnegative integer circulation $(f^*, F^{**})$, such that

$$0 \leq f_e^* \leq y_e^* - m \quad \text{for all } e \text{ in } E \text{ with } y_e^* \geq m$$

and

$$f_e^* = 0 \quad \text{if } y_e^* < m.$$

(Decreasing a flow along an arc by one unit is done by finding a cycle containing that arc and consisting of arcs with positive flow, and decreasing the flow on the cycle by one unit.)

Let $F^{**}$ be the value of the flow along the added arc from $n$ to 1. Therefore $f^*$ satisfies (2.3) and

$$y_e^* - m^2 \leq f_e^* \leq y_e^* - m \leq z_e^* \leq y_e^* + m \quad \text{for all } e \text{ with } y_e^* \geq m,$$

$$0 = f_e^* \leq z_e^* \leq y_e^* + m \quad \text{for all } e \text{ with } y_e^* < m, \tag{2.4}$$

$$q - m^2 \leq F^{**} \leq q.$$

Finally, define $v_e = z_e^* - f_e^*$ for all $e$ in $E$. Let $v = (v_e)$, $e \in E$. Therefore, by using

(2.4), the integer version of (1.1) with a diagonal $D$ is reduced to

$$\begin{aligned}
&\text{minimize} && ((v+f^*)'D(v+f^*)+c'(v+f^*)) \\
&\text{subject to} && (Av)_i=\begin{cases} q-F^{**}, & \text{if } i=1, \\ -(q-F^{**}), & \text{if } i=n, \\ 0, & \text{otherwise,} \end{cases} \\
& && 0\leqslant v_e=(z_e^*-f_e^*)\leqslant m^2+m, \quad \text{if } y_e^*\geqslant m, \\
& && 0\leqslant v_e=(z_e^*-f_e^*)=z_e^*\leqslant 2m, \quad \text{if } y_e^*<m, \\
& && v_e\leqslant u_e-f_e^* \quad \text{and } v_e \text{ integer for all } e\in E.
\end{aligned} \tag{2.5}$$

Since the arc capacities in (2.5) are bounded by a polynomial in $m$, and $q-F^{**}\leqslant m^2$, an optimal integer solution can be obtained in strongly polynomial time by the algorithm in [15] mentioned above. (2.5) can also be converted to the linear case and solved by the algorithm in [18] in strongly polynomial time.

We refer the reader to [9] where the use of proximity results and polynomial algorithms is discussed in the context of general convex separable nonlinear objective functions.

**Remark 2.1.** We have shown above that the integer solution can be derived from a fractional solution in strongly polynomial time using the operations $\{+, -, /, *, \lfloor\cdot\rfloor\}$. In a certain respect the inclusion of $\lfloor\cdot\rfloor$ is essential to obtain a strongly polynomial algorithm. Given two positive integers $a$ and $b$, the results in [16] imply that using the algebraic decision tree model with $\{+, -, /, *\}$, there is no strongly polynomial algorithm to compute $\lfloor b/a\rfloor$. Therefore the necessity of the $\lfloor\cdot\rfloor$ operation is implied by the following observation.

Let $(x_1^*, x_2^*)$ be an optimal integer solution to the problem:

$$\begin{aligned}
&\text{minimize} && \{\tfrac{1}{2}x_1^2+\tfrac{1}{2}(a-1)x_2^2\} \\
&\text{subject to} && x_1+x_2=b, \\
& && x_1 \text{ and } x_2 \text{ are nonnegative integer.}
\end{aligned}$$

Solving the rational model and using the above proximity result on the rational and the discrete solutions we note that $x_2^*-1\leqslant\lfloor b/a\rfloor\leqslant x_2^*$. Therefore, $\lfloor b/a\rfloor=x_2^*$ if $x_2^*\leqslant b/a$, and $\lfloor b/a\rfloor=x_2^*-1$ otherwise. Thus, if we could obtain $x_2^*$ in strongly polynomial time using only the operations in $\{+, -, /, *\}$, we would compute $\lfloor b/a\rfloor$ in strongly polynomial time using the same set of operations.

## 3. Solving the parametric quadratic cost flow problem on series–parallel graphs

In this section we present an $O(mn+m\log m)$ algorithm to generate the parametric separable convex quadratic cost flow function for a series-parallel graph with $m$ arcs.

We need the following definitions:

A (two terminal) series-parallel graph is a multi-graph with exactly one source and

one sink, which is defined recursively as follows:

(i) A single arc $e$ together with $t(e)$ and $h(e)$ is a series-parallel graph. $t(e)$ is the source terminal and $h(e)$ is the sink terminal.

(ii) If $G_1$ *and* $G_2$ are series-parallel graphs, so is the multi-graph obtained by either one of the following operations:

(a) Parallel composition: identify the source of $G_1$ with the source of $G_2$, and the sink of $G_1$ with the sink of $G_2$. The common source is the source of the composition and the common sink is its sink.

(b) Series composition: identify the sink of $G_1$ with the source of $G_2$. The source of $G_1$ is the source of the composition. The sink of the composition is the sink of $G_2$.

Let $G=(V, E)$ be a series-parallel multi-graph, with $V=\{1, \ldots, n\}$. Suppose that nodes 1 and $n$ denote the source and sink respectively. The parametric quadratic cost flow problem on $G$ is to determine the function $f(q)$ in (1.1) for all $q \geq 0$.

It follows from quadratic programming that $f(q)$ is convex and piecewise quadratic over its domain. For general graphs the number of pieces is known to be exponential in the number of arcs even when the objective in (1.1) is linear, [20].

Focusing on the series-parallel case we will show that for a separable quadratic objective, $f(q)$ has at most $2m$ quadratic pieces. Moreover, $2m$ is a sharp bound.

The linear case, $D=0$ in (1.1), for series-parallel graphs was discussed by Bein, Brucker and Tamir [1]. It is shown there that $f(q)$ has at most $m$ pieces, breakpoints. Furthermore, the function $f(q)$ can be generated efficiently by a greedy algorithm.

It is quite common to represent the recursive construction process of a series-parallel graph $G$ by a binary tree called a decomposition tree, [19]. The arcs of $G$ are represented by the leaves of the decomposition tree. Nodes of this tree, which are not leaves, are labelled by $S$, indicating a series composition, or $P$ indicating a parallel composition.

Our algorithm generates the envelope $f(q)$, $q \geq 0$, by following the decomposition tree. We start with the leaves of the tree. Each leaf represents a series-parallel graph consisting of a single arc, say $e$. Therefore, its parametric cost function is a quadratic function over its domain, $0 \leq q \leq u_e$. In the general step we consider two series-parallel subgraphs, $G_1$ and $G_2$, with their respective parametric cost functions $f_1(q_1)$ and $f_2(q_2)$.

If $G$ is obtained by a series composition of $G_1$ and $G_2$, then $f(q)$, the parametric cost function of $G$, is given by

$$f(q)=f_1(q)+f_2(q).$$

We say that $f$ is a *series composition* of $f_1$ and $f_2$. If $G$ is a parallel composition of $G_1$ and $G_2$ then $f(q)$ is defined by:

$$\begin{aligned} f(q) = \text{minimum} \quad & \{f_1(q_1)+f_2(q_2)\} \\ \text{subject to} \quad & q_1+q_2=q, \\ & q_1, q_2 \geq 0. \end{aligned}$$

In this case we say that $f$ is the *parallel composition* of $f_1$ and $f_2$.

To generate the parametric function for a series-parallel graph we will follow its decomposition tree and recursively perform the necessary series and parallel compositions of the respective cost functions. The total number of compositions is $m-1$. Before we describe the core of the algorithm, the parallel composition, we introduce the following terminology which is quite common for convex functions.

Let $g$ be a real continuous convex function from some closed interval $[0, \alpha]$ in $\mathbb{R}^1_+$ to $(-\infty, \infty)$. We extend $g$ to $\mathbb{R}^1_+$, by defining $g(x)=\infty$ for $x>\alpha$. $g$ is called an *extended convex* function on $\mathbb{R}^1_+$.

Let $g$ be an extended convex function on $\mathbb{R}^1_+$. $g$ is piecewise quadratic, if there exist $0=t_1<t_2<\cdots<t_k<t_{k+1}=\infty$, such that for each $j$, $1\leq j\leq k$, the reduction of $g$ to $[t_j; t_{j+1})$ is a quadratic function, say $g^j$, defined on $\mathbb{R}^1$, and for $1\leq j\leq k$, $g^j$ and $g^{j+1}$ are not identical. The points $\{t_1, \ldots, t_k\}$ are called the *breakpoints* of $g$, and the quadratics $\{g^1, \ldots, g^k\}$ are its *components*. $g$ is differentiable everywhere but possibly at its breakpoints. If $g(t_k+)=\infty$, then $g(x)=\infty$ for all $x>t_k$, and $t_k$ is the only discontinuity point of $g$. If $g(x)<\infty$ for all $x\geq 0$, $g$ is said to be *proper convex*. The derivative function, $g'$, is piecewise linear and nondecreasing with possible discontinuities at the breakpoints. $t_j$, $1\leq j\leq k$, is a discontinuity point of $g'$ if $g'(t_j-)<g'(t_j+)$. For convenience we set $g'(0-)=g'(0+)$. Also if $g(t_k+)=\infty$, we define $g'(t_k+)=\infty$.

An extended piecewise quadratic function $g$ will be represented by the sorted sequence of its breakpoints $\{t_1<t_2<\cdots<t_k\}$, and the respective quadratic components $\{g^1, \ldots, g^k\}$. Next we define a complexity measure for $g$. First define a set BP($g$) by

$$\begin{aligned}\mathrm{BP}(g)=&\{t_j-\,|\,t_j,\ 1\leq j\leq k,\ \text{is a breakpoint of } g\}\\ &\cup\{t_j+\,|\,t_j,\ 1\leq j\leq k,\ \text{is a breakpoint of } g\\ &\qquad\qquad \text{and } g'(t_j-)<g'(t_j+)<\infty\}.\end{aligned}$$

The complexity of $g$, CMPLX($g$), is defined by $\mathrm{CMPLX}(g)=|\mathrm{BP}(g)|$, i.e., CMPLX($g$) is the sum of the number of finite breakpoints of $g$ and the number of discontinuity points of $g'$.

It follows from convexity theory that both series and parallel compositions preserve convexity. If $g_1$ and $g_2$ are both piecewise quadratic then so is their series composition, $g$, and

$$\mathrm{CMPLX}(g)\leq \mathrm{CMPLX}(g_1)+\mathrm{CMPLX}(g_2).$$

Furthermore, $g$ can easily be generated from $g_1$ and $g_2$ in $O(\mathrm{CMPLX}(g_1)+\mathrm{CMPLX}(g_2))$ time by merging the two sequences of breakpoints.

We will next focus on the parallel composition and show that it preserves the above properties as well.

To facilitate the discussion, let $g_1$ and $g_2$ be a pair of convex piecewise quadratic functions. Let $0=t_{1,1}<t_{1,2}<\cdots<t_{1,k(1)}<t_{1,k(1)+1}=\infty$ and $0=t_{2,1}<t_{2,2}<\cdots<t_{2,k(2)}<t_{2,k(2)+1}=\infty$ indicate their respective sequences of breakpoints. Consider $g$,

the parallel composition of $g_1$ and $g_2$:

$$\begin{aligned} g(q) = \text{minimum} \quad & \{g_1(q_1) + g_2(q_2)\} \\ \text{subject to} \quad & q_1 + q_2 = q, \\ & q_1, q_2 \geqslant 0. \end{aligned} \tag{3.1}$$

$g(q)$ is the solution value for the classical continuous effort allocation problem with two variables [10]. The solution to this problem can be obtained by modifying the incremental algorithm of Luss and Gupta [12], which is originally designed for the case where both $g_1$ and $g_2$ are differentiable strictly convex functions. Basically, the incremental algorithm to generate $g(q)$ for all $q \geqslant 0$, amounts to a (continuous) sorting of the two derivative functions, $g_1'$ and $g_2'$. The precise meaning of this approach will be clear after we describe the algorithm. But an intuitive explanation is provided when we look at the discrete version of (3.1) where $q_1$ and $q_2$ are restricted to integral values. In this case, the solution to the problem is obtained by merging the two discrete derivatives, i.e., the two monotone sequences: $\{g_1(j+1) - g_1(j)\}$, $j = 0, 1, 2, \ldots$, and $\{g_2(j+1) - g_2(j)\}$, $j = 0, 1, 2, \ldots$. For any integral $q \geqslant 0$, $g(q)$ is obtained by selecting the $q$th smallest element of the merged sequence, [10].

The main idea in the solution of (3.1) is quite simple. For each value of the argument $q$ we look for values of $q_1$ and $q_2$ that sum up to $q$, and such that the deravitive functions $g_1'$ and $g_2'$, evaluated at $q_1$ and $q_2$ respectively, are equal. However, the derivative functions can possess discontinuities and therefore exact equality may not be feasible. Hence, we have to consider and carefully examine the breakpoints of $g_1'$ and $g_2'$, while parametrically varying $q$. This is the essence of the detailed technical discussion of parallel composition.

We now describe the algorithm to compute the function $g(q)$, $q \geqslant 0$, defined as the parallel composition of $g_1$ and $g_2$.

**Algorithm 1.**

*Step 0.* Initially start with $q^1 = q^2 = 0$. $q^1 \in [t_{1,1}; t_{1,2})$ and $q^2 \in [t_{2,1}; t_{2,2})$.

*Step 1.* Let $q^1 \in [t_{1,j(1)}; t_{1,j(1)+1})$, for some $1 \leqslant j(1) \leqslant k(1)$, and $q^2 \in [t_{2,j(2)}; t_{2,j(2)+1})$, for some $1 \leqslant j(2) \leqslant k(2)$. ($g(q)$ has already been defined and represented for all $0 \leqslant q \leqslant q^1 + q^2$).

Consider $g_1'(q^1+)$ and $g_2'(q^2+)$, and the respective intervals $(q^1; t_{1,j(1)+1})$ and $(q^2; t_{2,j(2)+1})$. $g_1'$ and $g_2'$ are linear when restricted to the above intervals.

If $g_1'(q^1+) < g_2'(q^2+)$ go to Step 2. If $g_1'(q^1+) > g_2'(q^2+)$ go to Step 3. If $g_1'(q^1+) = g_2'(q^2+)$ proceed to Step 4.

*Step 2.* ($g_1'(q^1+) < g_2'(q^2+)$). Define $\bar{q}$ to be the largest real $\bar{q} \leqslant t_{1,j(1)+1}$ such that $g_1'(\bar{q}-) \leqslant g_2'(q^2+)$. For $0 \leqslant q \leqslant \bar{q} - q^1$, define

$$g(q^1 + q^2 + q) = g_1(q^1 + q) + g_2(q^2).$$

If $\bar{q} = \infty$ stop. Otherwise set $q^1 \leftarrow \bar{q}$ and go to Step 1.

*Step 3.* $(g_1'(q^1+) > g_2'(q^2+))$. Define $\bar{q}$ to be the largest real $\bar{q} \leq t_{2,j(2)+1}$ such that $g_2'(\bar{q}-) \leq g_1'(q^1+)$. For $0 \leq q \leq \bar{q} - q^2$ define

$$g(q^1+q^2+q) = g_1(q^1) + g_2(q^2+q).$$

If $\bar{q} = \infty$ stop. Otherwise set $q^2 \leftarrow \bar{q}$ and go to Step 1.

*Step 4.* $(g_1'(q^1+) = g_2'(q^2+))$.

If $g_1$ is linear over $(q^1; t_{1,j(1)+1})$ go to Step 4.1. If $g_2$ is linear over $(q^2; t_{2,j(2)+1})$ go to Step 4.2. Otherwise go to Step 4.3.

*Step 4.1.* Let $\bar{q} = t_{1,j(1)+1}$. For $0 \leq q \leq \bar{q} - q^1$ define

$$g(q^1+q^2+q) = g_1(q^1+q) + g_2(q^2).$$

If $\bar{q} = \infty$ stop. Otherwise set $q^1 \leftarrow \bar{q}$ and go to Step 1.

*Step 4.2.* Let $\bar{q} = t_{2,j(2)+1}$. For $0 \leq q \leq \bar{q} - q^2$ define

$$g(q^1+q^2+q) = g_1(q^1) + g_2(q^2+q).$$

If $\bar{q} = \infty$ stop. Otherwise set $q^2 \leftarrow \bar{q}$ and go to Step 1.

*Step 4.3.* ($g_1$ and $g_2$ are proper quadratic over their respective intervals). Define $q_1 = q_1(q)$ and $q_2 = q_2(q)$ by the unique solution to the system

$$g_1'(q^1+q_1) = g_2'(q^2+q_2),$$

$$q_1 + q_2 = q - q^1 - q^2.$$

($q_1(q)$ and $q_2(q)$ are both increasing functions of $q$.) Let $\bar{q}$ be the largest real such that

$$q^1 + q_1(\bar{q}) \leq t_{1,j(1)+1},$$

$$q^2 + q_2(\bar{q}) \leq t_{2,j(2)+1}.$$

For $0 \leq q \leq \bar{q} \leq q^1 - q^2$ define

$$g(q^1+q^2+q) = g_1(q^1+q_1(q)) + g_2(q^2+q_2(q)).$$

If $\bar{q} = \infty$ stop. Otherwise, set $q^1 \leftarrow q^1 + q_1(\bar{q})$, $q^2 \leftarrow q^2 + q_2(\bar{q})$, and go to Step 1.

The validity of the above algorithm to compute $g(q)$, $q \geq 0$, follows from the validity of the incremental algorithm in [10].

For every $q \geq 0$, let $q_1^*(q)$ and $q_2^*(q)$ denote the solution to the problem (3.1) as defined by Algorithm 1. It follows that $q_1^*(q)$ and $q_2^*(q)$ are continuous, piecewise linear and nondecreasing functions of $q$. Therefore, $g(q)$ is a continuous, extended convex piecewise quadratic function on $\mathbb{R}_+^1$. The breakpoints of $g(q)$ are sequentially generated by the above algorithm. Starting with $q = 0$, each iteration computes the next breakpoint of $g$. It is given by $q = q^1 + q^2$, where $q^1$ and $q^2$ are defined in Step 1. The computational effort to compute $g(q)$, $q \geq 0$, is certainly linear in $\text{CMPLX}(g_1) + \text{CMPLX}(g_2) + \text{CMPLX}(g)$.

**Remark 3.1.** The above definition of a breakpoint of $g$ requires that the two quadratics defining $g$ on both sides of the breakpoint are not identical. Due to some degeneracy this property might not hold for some pairs $(q^1, q^2)$, generated in Step 1. However, we will ignore this aspect since the proper breakpoints can easily be detected when Algorithm 1 terminates.

The validity of Algorithm 1 implies the following properties that we state without a proof.

**Proposition 3.2.** *Let $q$ be a breakpoint of the parallel composition function $g$. Then there exist $q^1$ and $q^2$, defined in Step 1 at some iteration of Algorithm 1, such that $q = q^1 + q^2$ and,*

(1) *Either $q^1$ is a breakpoint of $g_1$ or $q^2$ is a breakpoint of $g_2$.*

(2) $$g'(q+) = \min\{g_1'(q^1+), g_2'(q^2+)\},$$
$$g'(q-) = g_1'(q^1-) \quad \text{if } q^1 = q \text{ and } q^2 = 0,$$
$$g'(q-) = g_2'(q^2-) \quad \text{if } q^1 = 0 \text{ and } q^2 = q,$$
$$g'(q-) = \max\{g_1'(q^1-), g_2'(q^2-)\} \quad \text{if } q^1, q^2 > 0.$$

(3) *For every $i = 1, 2$ and $q^* < q^i$, $g_i'(q^*-) \leq g_i'(q^*+) \leq g'(q-)$.* □

**Theorem 3.3.** *Let $g$ be the parallel composition of $g_1$ and $g_2$ defined by Algorithm 1. Then*

$$\text{CMPLX}(g) \leq \text{CMPLX}(g_1) + \text{CMPLX}(g_2). \tag{3.2}$$

**Proof.** By definition, CMPLX($g$) is the sum of the breakpoints of $g$ and the number of discontinuity points of $g'$. To prove (3.2) we use Algorithm 1 to identify the discontinuity points and define a one to one mapping from BP($g$) to BP($g_1$) ∪ BP($g_2$).

Using the above notation, for $i = 1, 2$

$$\text{BP}(g_i) = \{t_{i,j} - \mid t_{i,j},\ 1 \leq j \leq k(i), \text{ is a breakpoint of } g_i\}$$
$$\cup \{t_{i,j} + \mid t_{i,j},\ 1 \leq j \leq k(i), \text{ is a breakpoint of } g_i \text{ and } g_i'(t_{i,j}-) < g_i'(t_{i,j}+) < \infty\}.$$

Starting with $t = 0$, Algorithm 1 defines all the breakpoints of $g$ sequentially in Step 1 above. Let $0 = t_1 < t_2 < \cdots < t_r < t_{r+1} = \infty$ be these breakpoints. (See Remark 3.1 above.)

Let $t_j = q^1 + q^2$ be such a breakpoint and suppose, inductively, that $t_j-$ has already been mapped into some element in BP($g_1$) ∪ BP($g_2$). The basic property of this mapping is that it maintains the same derivative value. For example if $t_j-$ is mapped into $t_{1,p}-$ then $g'(t_j-) = g_1'(t_{1,p}-)$.

From Step 1 we enter into one of the other following steps with a pair $(q^1, q^2)$.

Suppose first that we enter Step 2. Let $t_j = q^1 + q^2$. $g'((q^1+q^2)+) = g_1'(q^1+)$. If $g'((q^1+q^2)-) = g_1'(q^1+)$, then $t_j$ is not a discontinuity of $g'$, and therefore $t_j+$ is not in BP($g$). Otherwise, $g'((q^1+q^2)-) < g_1'(q^1+)$. If $q^1 = 0$, then $q^1 = t_{1,1} = 0$ and

$t_{1,1}-$ in BP($g_1$) has not been yet mapped onto. Add $t_j+$ to BP($g$) and map it onto $t_{1,1}-$. If $q^1>0$ we have from Proposition 3.2,

$$g_1'(q^1-) \leqslant g'((q^1+q^2)-) < g_1'(q^1+),$$

and $q^1$ is a discontinuity point of $g_1'$. Specifically, $q^1 = t_{1,j(1)}$ from some $j(1)$ in Step 1. Moreover, since $g_1'(q^1+) > g'((q^1+q^2)-)$, the element $t_{1,j(1)}+$ in BP($g_1$) has not been mapped onto by an element in BP($g$). Thus, we map $t_j+$ onto $t_{1,j(1)}+$ in BP($g_1$). Before we return to Step 1 we define $t_{j+1}$ and map $t_{j+1}-$.

Consider the updating of $q^1$ by $q^1 \leftarrow \bar{q}$ where $\bar{q}$ is defined in Step 2. If $\bar{q} = t_{1,j(1)+1}$, augment $t_{j+1}-$ to BP($g$) and map it onto $t_{1,j(1)+1}-$. Otherwise, $\bar{q} < t_{1,j(1)+1}$, and is defined by $g_1'(\bar{q}-) = g_2'(q^2+)$. If $q^2 = 0$ then $q^2 = t_{2,1} = 0$ and $t_{2,1}-$ in BP($g_1$) has not yet been mapped onto. We add $t_{j+1}-$ to BP($g$) and map it onto $t_{2,1}-$. If $q^2>0$, we claim that $q^2$ is a discontinuity point of $g_2$. Indeed, using Proposition 3.2 we have

$$g_2'(q^2-) \leqslant g'((q^1+q^2)-) \leqslant g'((q^1+q^2)+) = g_1'(q^1+) < g_2'(q^2+).$$

Therefore $q^2 = t_{2,j(2)}$ for some $j(2)$ in Step 1, and $t_{2,j(2)}+$ is in BP($g_2$). We augment $t_{j+1}-$ to BP($g$) and map it onto $t_{2,j(2)}+$. (We note in passing that in this case $t_{j+1}$ is not a discontinuity point of $g'$ since $\bar{q}$ is not a breakpoint of $g_1$ and therefore $g'(t_{j+1}+) = g_1'(\bar{q}-) = g'(t_{j+1}-)$.)

The case where we enter Step 3 from Step 1 is totally symmetric to the case where we enter Step 2. Thus we skip the details.

Suppose that Step 1 directs to Step 4.1, and let $t_j = q^1+q^2$. As above $g'((q^1+q^2)+) = g_1'(q^1+)$. If $g'((q^1+q^2)-) = g'(q^1+)$ then, $t_j+$ is not in BP($g$). If $g'((q^1+q^2)-) < g_1'(q^1+)$, then add $t_j+$ to BP($g$) and map it as in Step 2 above.

The next breakpoint of $g$, $t_{j+1}$, is defined in Step 4.1 by setting $q^1 \leftarrow t_{1,j(1)+1}$. Thus, add $t_{j+1}-$ to BP($g$) and map it onto $t_{1,j(1)+1}-$ in BP($g_1$).

The case where we enter Step 4.2 is totally symmetric and therefore the details are omitted.

Finally, suppose we enter Step 4.3. Let $t_j = q^1+q^2$. Again if $g'((q^1+q^2)-) = g_1'(q^1+)$ $(= g_2'(q^2+))$, $t_j+$ is not in BP($g$). Let $g'((q^1+q^2)-) < g_1'(q^1+)(= g_2'(q^2+))$. From Property 2 of Proposition 3.2, assume without loss of generality that

$$g_1'(q^1-) = g'((q^1+q^2)-) < g_1'(q^1+),$$

and $q^1$ is a discontinuity point of $g_1'$. $q^1 = t_{1,j(1)}$ for some $j(1)$ in Step 1. Augment $t_j+$ to BP($g$) and assign it the element $t_{1,j(1)}+$ in BP($g_1$). The new breakpoint of $g$, $t_{j+1}$, is defined in Step 4.3 by $t_{j+1} = \bar{q}$, where $q_1+q_1(\bar{q}) = t_{1,j(1)+1}$ or $q_2+q_2(\bar{q}) = t_{2,j(2)+1}$. In the former case map $t_{j+1}-$ in BP($g$) onto $t_{1,j(1)+1}-$ in BP($g_1$) while if the latter holds $t_{j+1}-$ is mapped onto $t_{2,j(2)+1}$ in BP($g_2$).

To conclude the proof of (3.2) we note that we have constructed a one to one map from BP($g$) to BP($g_1$) $\cup$ BP($g_2$). □

We are now ready to describe the algorithm to compute $f(q)$, the parametric separable cost flow function in (1.1), for a series-parallel graph $G$ with $m$ arcs and $n$ nodes. Let $T$ denote the decomposition tree representing $G$. Each leaf of $T$

represents some arc of $G$. Thus $T$ has $m$ leaves and $m-1$ other nodes which describe the sequence of series and parallel compositions.

Starting with an arc $e$ of $G$, its respective parametric function is quadratic in the interval $[0; u_e]$, where $u_e$, defined in (1.1), is the capacity bound on the flow on $e$. Viewed as an extended convex quadratic, this function has a single interior breakpoint at $u_e$. The algorithm proceeds recursively and performs the necessary series and parallel compositions of the respective parametric cost function. Specifically, the parametric cost function of a series composition is obtained by a simple addition of the cost functions, corresponding to the two composites. The parametric cost function of a parallel composition is computed by Algorithm 1.

The total number of series and parallel compositions performed by the algorithm is $m-1$. Let $\{v_1, \ldots, v_{2m-1}\}$ be the node set of the decomposition tree $T$, with $v_1$ denoting its root node. For each node $v_i$ let $L_i$ denote the set of leaf nodes in $T$ having $v_i$ on the unique simple path on $T$ connecting them to the root $v_1$. Let $f_i(q)$, $q \geq 0$, denote the parametric cost function associated with $v_i$ and generated by the above algorithm. In particular, if $v_i$ is a leaf $f_i$ is quadratic in the interval $[0; u_e]$, $f_i'(u_e+) = \infty$, and $\mathrm{CMPLX}(f_i) = 2$. Applying Theorem 3.3 we observe that for any node $v_i$ of $T$,

$$\mathrm{CMPLX}(f_i) \leq \sum_{v_j \in L_i} \mathrm{CMPLX}(f_j).$$

Therefore, $\mathrm{CMPLX}(f_i) \leq 2|L_i|$. $f(q)$, $q \geq 0$, the parametric cost function of $G$ coincides with $f_1(q)$, $q \geq 0$, the function associated with $v_1$, the root of $T$. Thus, $\mathrm{CMPLX}(f) \leq 2m$. Since the complexity of Algorithm 1 to compute $f_i$ is $O(|L_i|)$, it follows that the total effort to compute $f$ is $O(m^2)$.

The $O(m^2)$ complexity bound can be improved to $O(mn + m \log m)$ if we apply some preprocessing. Consider the original $n$ node $m$ arc series-parallel multi-graph $G = (V, E)$. If there are no parallel arcs $m \leq 2n-3$. However, in a multi-graph $m$ can be significantly larger than $n$. In such cases we apply the following preprocessing before we use the above algorithm. In the preprocessing each subset of parallel arcs connecting the same pair of nodes will be replaced by a single arc.

Let $i$ and $j$ be a pair of nodes in $V$, and let $E(i, j)$ be the subset of all parallel arcs having $i$ and $j$ as their head and tail respectively. Let $m(i, j) = |E(i, j)|$. Consider $G(i, j)$, the subgraph of $G$ induced by $E(i, j)$. The parametric function $f(q)$, $q \geq 0$, corresponding to $G(i, j)$ has at most $2m(i, j)$ breakpoints. To obtain the representation of $f(q)$ for $G(i, j)$ we apply Algorithm 1 recursively, using a standard divide and conquer approach: Divide $E(i, j)$ into two subsets of equal cardinality, recursively find the $f$ function of each subset, and then use Algorithm 1 to obtain $f(q)$ for $G(i, j)$. Since Algorithm 1 takes $O(m(i, j))$ time, and the recursion has $O(\log m(i, j))$ levels, the total time to obtain the representation of $f(q)$, $q \geq 0$, for $G(i, j)$ is $O(m(i, j) \log m(i, j))$.

We apply the above preprocessing phase to all pairs of adjacent nodes in $G$. The total preprocessing time is therefore $O(m \log m)$.

After the preprocessing we consider $G'$, the simple graph induced by $G$, i.e., for each pair of adjacent nodes $i$ and $j$ we eliminate all but one arc from $E(i,j)$. That arc is now associated with the function $f$ of $G(i,j)$ computed during the preprocessing. We now apply our above algorithm, using the decomposition tree of $G'$. Since $G'$ has only $O(n)$ arcs, and the total number of breakpoints associated with all the graphs $G(i,j)$ is $O(m)$, the total time for processing $G'$ to obtain its parametric function $f$ is $O(mn)$. The total time including the preprocessing is therefore $O(mn + m \log m)$.

The above is summarized in the following:

**Corollary 3.4.** *Let $G$ be a two-terminal series-parallel graph with $n$ nodes and $m$ arcs. Let $f(q)$, $q \geq 0$, be the parametric separable convex quadratic cost flow function defined on $G$ by (1.1) with a diagonal matrix $D$. Then $f(q)$ is a convex piecewise quadratic function with at most $2m$ breakpoints. The total effort to compute all the breakpoints of $f$ and its quadratic components is $O(mn + m \log m)$.* □

**Example 3.5.** It is shown in [1] that the parametric function $f(q)$, $q \geq 0$, has at most $m+1$ breakpoints in the linear case, i.e., $D = 0$. The following example demonstrates that the upper bound $2m$ is sharp in the quadratic case.

Consider a multi-graph consisting of $m$ parallel arcs $\{e(1), \ldots, e(m)\}$. Let the capacity bounds be defined by

$$u_{e(i)} = \begin{cases} 1, & 1 \leq i \leq m-1, \\ m, & i = m. \end{cases}$$

If $x_{e(i)}$, $1 \leq i \leq m$, is the flow on $e(i)$, then $Q(x_{e(i)})$, the corresponding cost is given by

$$Q(x_{e(i)}) = \begin{cases} ix_{e(i)}, & 1 \leq i \leq m-1, \\ \frac{1}{2}x_{e(m)}^2, & i = m. \end{cases}$$

Using the above algorithm we verify that $f(q)$, $q \geq 0$, is differentiable and has a breakpoint at each $q$ in $\{0, 1, \ldots, 2m-1\}$. Its derivative is defined by

$$f'(q) = \begin{cases} q - \frac{1}{2}i & \text{if } i \leq q \leq i+1,\ i \text{ is even and } i \leq 2m-2, \\ \frac{1}{2}(i+1) & \text{if } i \leq q \leq i+1,\ i \text{ is odd and } i \leq 2m-3. \end{cases}$$

## 4. Summary and concluding remarks

We have given an $O(mn + m \log m)$ algorithm to construct and represent the parametric cost flow function $f(q)$, $q \geq 0$, for a series-parallel graph with $m$ arcs. Given a value for $q$, say $q^*$, it takes $O(\log m)$ time to obtain $f(q^*)$ from the above representation. Using a binary search we locate the two consecutive breakpoints of $f(q)$, say $t_j$ and $t_{j+1}$, which bound $q^*$ from both sides. The restriction of $f(q)$ to

the interval $[t_j; t_{j+1}]$ is a quadratic function. $f(q^*)$ is the value of that quadratic at $q = q^*$.

To solve the discrete version of (1.1) for a given value of $q$, say $q^*$, we use the approach in Section 2. We first compute the fractional solution, and then reduce the discrete problem to the formulation (2.5). In the latter formulation all arc capacities are bounded by $m + m^2$. As mentioned in Sections 1 and 2, such a problem can be solved in strongly polynomial time even for a general graph by the algorithms in [15, 18]. For series-parallel graphs we can use an alternative algorithm. Using a standard transformation, the convex separable program in (2.5) can be converted into a linear integer flow problem on a series-parallel graph with $O(m^3)$ arcs. The latter problem can then be solved in strongly polynomial time by the greedy algorithm in [1].

Next we raise a question with respect to the discrete version of the parametric quadratic cost flow problem. For each nonnegative integer $q$, let $\bar{f}(q)$ denote the solution to the discrete version of (1.1) for the given series-parallel graph. We have demonstrated above how to compute $\bar{f}(q)$, for a given integer $q$, in strongly polynomial time.We have also represented the parametric function for the fractional case, $f(q)$, $q \geq 0$, as a piecewise quadratic with at most $2m$ quadratic components. Is there a compact and polynomial representation of $\bar{f}(q)$, $q \geq 0$ and integer, the parametric function for the discrete case?

One might hope that there is some real function $h(q)$, $q \geq 0$, such that $h(q) = \bar{f}(q)$ for every nonnegative integer and $h$ has some compact representation. For example, is $h$ representable by a polynomial number of polynomial components of a fixed bounded degree? We conjecture that this representation is infeasible in general. We were only able to demonstrate that there is no efficient representation by quadratic components.

Finally we comment on the extension of (1.1) for separable convex cost functions defined by higher degree polynomials. Consider, for example, the case where for each arc $e$ the cost of flowing $x_e$ units on $e$ is a convex cubic function with rational coefficients, say $P_e(x_e)$. Given a rational $q \geq 0$ the objective is to minimize $\sum_{e \in E} P_e(x_e)$ subject to the constraints in (1.1). Let $f^*(q)$ denote the optimal objective value. Unlike the quadratic case, $x^*$, the optimal solution, is not necessarily rational. It is algebraic, i.e., for each arc $e$, $x_e^*$ is a root of some minimal one-dimensional polynomial with integer coefficients; its characteristic polynomial. The degree and the height of this characteristic polynomial are not even known to be of polynomial size.

To demonstrate the latter point consider the following special case of a cubic flow problem on a tree graph.

$$\begin{aligned} f^*(q) = \text{minimum} \quad & \sum_{i=1}^{n} \frac{x_i^3}{3a_i} \\ \text{subject to} \quad & x_1 + \cdots + x_n = q, \\ & x_i \geq 0, \quad 1 \leq i \leq n, \end{aligned} \tag{4.1}$$

where $a_i$, $i=1,\ldots,n$, is a positive integer. The objective in (4.1) is strictly convex over its domain. Using Kuhn-Tucker optimality conditions we obtain the unique optimal solution given by

$$x_i^* = \frac{\sqrt{a_i}\, q}{\sum_{j=1}^{n} \sqrt{a_j}}, \quad i=1,\ldots,n. \tag{4.2}$$

$x_i^*$, $i=1,\ldots,n$, is an algebraic number. However, the best known bound on the degree and the height of the characteristic polynomial of $x_i^*$ is exponential in $n$. (See [3] and the references cited there. The difficulty lies in determining the algebraic degree and height of the expression $\sum_{i=1}^{n} \sqrt{a_i}$.)

In the example given by (4.1), $f^*(q)$ is a convex cubic function of $q$ with algebraic coefficients. However, in general, unlike the quadratic case, $f^*(q)$ is not even piecewise polynomial. This is demonstrated by the following:

$$\begin{aligned} f^*(q) = \text{minimum} \quad & \{\tfrac{1}{3}x_1^3 + \tfrac{1}{2}x_2^2\} \\ \text{subject to} \quad & x_1 + x_2 = q, \\ & x_1, x_2 \geq 0. \end{aligned} \tag{4.3}$$

**Remark 4.1.** The above examples demonstrate some of the numerical difficulties in computing a real solution to the flow problem with a convex, separable cubic cost function. Nevertheless, note that the discrete version of this cubic model is polynomially solvable by the algorithm in [15].

Dorit Hochbaum, [7, 8], has recently proved that there is no strongly polynomial algorithm to solve the discrete cubic case while using only the operations $\{+, -, /, *\}$. The reduction in Remark 2.1 is stronger. It shows that even the quadratic case is not solvable in strongly polynomial time. In fact, using the recent results in [13] one can argue that even if the $\lfloor \cdot \rfloor$ operation is permissible, the discrete cubic model can not be solved in strongly polynomial time under a reasonable computation tree model.

## References


[1] W. Bein, P. Brucker and A. Tamir, "Minimum cost flow algorithms for series parallel networks," *Discrete Applied Mathematics* 10 (1985) 117-124.

[2] R. Chandrasekaran and S. Kabadi, "Strongly polynomial algorithm for a class of combinatorial LCPs," *Operations Research Letters* 6 (1987) 91-92.

[3] R. Chandrasekaran and A. Tamir, "Algebraic optimization: the Fermat-Weber location problem," *Mathematical Programming* 46 (1990) 219-224.

[4] F. Granot and J. Skorin-Kapov, "Towards a strongly polynomial algorithm for strictly convex quadratic programs: An extension of Tardos' algorithm," *Mathematical Programming* 46 (1990) 225-236.

[5] F. Granot and J. Skorin-Kapov, "Some proximity and sensitivity results in quadratic integer programming," *Mathematical Programming* 47 (1990) 259-268.

[6] P. Hansen and D. Moon, "Dispersing facilities on a network," RRR #52-88, RUTCOR, Rutgers University (New Brunswick, NJ, 1988).
[7] D. Hochbaum, Private communication (May 1989).
[8] D. Hochbaum, "Optimal algorithms for the allocation problem and its extensions," Technical Report, IEOR Department, University of California (Berkeley, CA, 1990).
[9] D. Hochbaum and J.G. Shanthikumar, "Convex separable optimization is not much harder than linear optimization," to appear in: *Journal of the ACM* 1990.
[10] T. Ibaraki and N. Katoh, *Resource Allocation Problems: Algorithmic Approaches* (The MIT Press, Cambridge, MA 1988).
[11] M.K. Kozlov, S.P. Tarasov and L.G. Khachian, "The polynomial solvability of convex quadratic programming," *Soviet Mathematics Doklady* 20(5) (1979) 1108-1111. [English translation.]
[12] H. Luss and S. K. Gupta, "Allocation for effort resources among competitive activities," *Operations Research* 23 (1975) 360-366.
[13] Y. Mansour, B. Schieber and P. Tiwari, "Lower bounds for computations with the floor function," *SIAM Journal on Computing* 20 (1991) 315-327.
[14] M. Minoux, "A polynomial algorithm for minimum quadratic cost flow problems," *European Journal of Operational Research* 18 (1984) 377-387.
[15] M. Minoux, "Solving integer minimum cost flows with separable convex objective polynomially," *Mathematical Programming Study* 26 (1986) 237-239.
[16] L. Stockmeyer, "Arithmetic versus boolean operations in idealized register machines," Technical Report RC 5954, IBM T.J. Watson Research Center (Yorktown Heights, NY, 1976).
[17] A. Tamir, "Obnoxious facility location on graphs," *SIAM Journal on Discrete Mathematics* 4 (1991) 550-567.
[18] E. Tardos, "A strongly polynomial minimum cost circulation algorithm," *Combinatorica* 5 (1985) 247-255.
[19] J. Valdes, R.E. Tarjan and E.L. Lawler, "The recognition of series-parallel diagraphs," *SIAM Journal on Computing* 11 (1982) 298-313.
[20] N. Zadeh, "A bad network problem for the simplex method and other minimum cost flow algorithms," *Mathematical Programming* 5 (1973) 255-266.